(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 February 2017 (02.02.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/019042 A1

(51) International Patent Classification:
G06F 12/08 (2006.01) *G06F 12/02* (2006.01)

(21) International Application Number:
PCT/US2015/042500

(22) International Filing Date:
28 July 2015 (28.07.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **SHERLOCK, Derek Alan**; 3404 East Harmony Road, Fort Collins, Colorado 80528-9544 (US). **WALKER, Shawn**; 3404 East Harmony Road, Fort Collins, Colorado 80528-9544 (US).

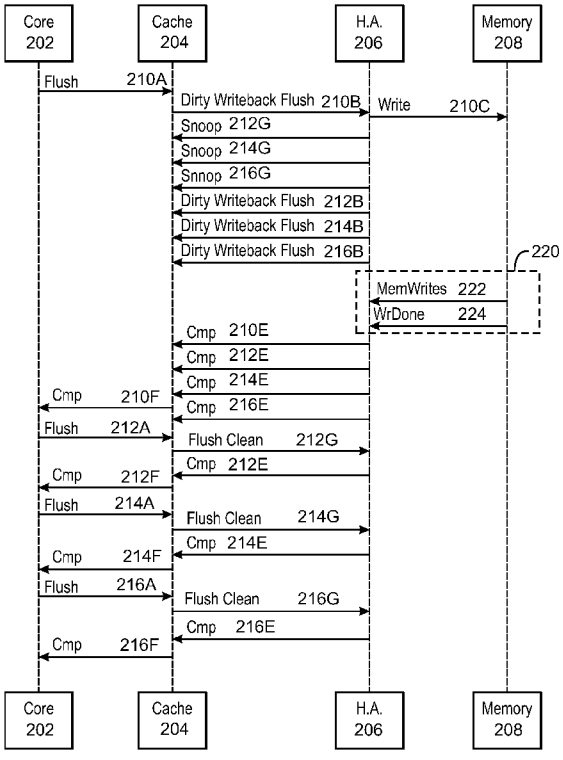
(74) Agents: **FEBBO, Michael A.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (US).

(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, BN, BR, BW, BY, BZ, CA, CH, CL, 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, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, 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, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE,

[Continued on next page]

(54) Title: HARDWARE FLUSH ASSIST



200B

FIG. 2B

(57) Abstract: An apparatus for assisting a flush of a cache is described herein. The apparatus comprises processing element. The processing element is to probe a cache line at an offset address and write the cache line at the offset address to a non-volatile memory in response to a flush instruction at a first address.

WO 2017/019042 A1

DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, 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))*

Published:

— *with international search report (Art. 21(3))*

HARDWARE FLUSH ASSIST

BACKGROUND

[0001] The advent of cloud computing and handheld devices has increased the demand for faster and more reliable access to data. Database performance relies heavily on the average access time of storage by a processor. Systems heretofore utilize various techniques to enhance the performance of data transfers between processors and storage.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The advantages of the present examples are better understood by referring to the following detailed description and the attached drawings, in which:

[0003] Fig. 1 is a block diagram of a computing device that may be used to assist a flush of a cache;

[0004] Fig. 2A is a diagram of a system with data transfers in a load-store model as described herein;

[0005] Fig. 2B is a diagram of a system with data transfers with a hardware cache assist in a load-store model as described herein; and

[0006] Fig. 3 is a process flow diagram of a method for a hardware flush assist.

[0007] The same numbers are used throughout the disclosure and the figures to reference like components and features. Numbers in the 100 series refer to features originally found in Fig. 1; numbers in the 200 series refer to features originally found in Fig. 2; and so on.

DETAILED DESCRIPTION

[0008] Most central processing unit (CPU) architectures enable cache management instructions which explicitly flush modified regions of the cache to memory under programmatic control. The region of the cache may be a cache line. Additionally, this is type of instruction is often termed a flush operation when the cache line is also invalidated, or a clean operation when the cache line remains valid. As used herein, a flush, flush operation, or flush instruction is the transfer of data from a cache to another level of memory. In some embodiments, the flush

operation is performed from the cache to a non-volatile memory. For ease of description, the present techniques are described using a flush, flush operation, or flush instruction. However, the present techniques can be applied to any instruction where data is transferred from a first regions to another location.

[0009] Synchronization barrier instructions to enable notifications to software when all previously-issued flush operations have completed. These barrier instructions prevent race condition hazards that might otherwise occur if, for example, a stored data record were marked as being valid before the content of the record was in fact valid. By flushing the record contents and then using a synchronization barrier to ascertain that the flush was complete, the software can delay marking the record as valid until its content is known to already be written to persistent memory.

[0010] In CPU architectures that include a blocking direct memory access (DMA) model for non-volatile memory access and a load-store model for volatile memory access, the instructions for flushing and for synchronization barriers are intended for relatively low-volume tasks, such as synchronizing data structures shared between caches lacking hardware coherency. For example, the flushing and synchronization barriers are used to synchronize data and instruction caches in self-modifying code, or to synchronize page table updates with translation caches. As a result, the flush instructions and synchronization barriers may not be well-suited to high volume use. For example, these instructions may lead to poor system performance when applied to high volume usage models. This poor system performance is a result of small tracker sizes, lack of pipelining and concurrency optimization, etc.

[0011] Applying a load-store model to non-volatile memory places demands upon these instructions that can result in high volume use. For example, a database designed to operate directly within non-volatile memory may end up doing most or all of its writes as part of commit sequences, which depend upon flush instructions rather than cache evictions to write the modifications into non-volatile memory. To maintain acceptable system performance when flush operations are applied to a non-volatile memory under a load-store model, hardware may be used to assist the flushing of a cache to non-volatile memory. Embodiments described herein include a memory controller architecture that assists with the process of flushing targeted

regions of a cache to overcome performance limitations of the CPU's instruction-driven flush sequences. In some embodiments, based on an initial flush instruction, the cache is then flushed at an address that is offset from the address of the initial flush instruction.

[0012] Fig. 1 is a block diagram of a computing device 100 that may be used to assist a flush of a cache. The computing device 100 may be, for example, a laptop computer, desktop computer, ultrabook, tablet computer, mobile device, or server, among others. The computing device 100 may include a central processing unit (CPU) 102 that is configured to execute stored instructions, as well as a memory device 104 that stores instructions that are executable by the CPU 102. The computing device 100 may also include a cache 106 and non-volatile memory 108. The CPU may be coupled to the memory device 104, cache 106, and non-volatile memory 108 by a bus 110. Additionally, the CPU 102 can be a single core processor, a multi-core processor, a computing cluster, or any number of other configurations. Furthermore, the computing device 100 may include more than one CPU 102.

[0013] The memory device 104 can include random access memory (RAM), read only memory (ROM), flash memory, or any other suitable memory systems. For example, the memory device 104 may include dynamic random access memory (DRAM). The cache 106 may be a smaller, faster memory that stores a smaller subset of frequently used data for the CPU 102. In some embodiments, the cache 106 is a multi-level cache. Additionally, the cache 106 may be integrated with the memory device 104. A larger data set may be stored in a non-volatile memory 108. The non-volatile memory 108 may be a physical memory such as a hard drive, an optical drive, a thumbdrive, an array of drives, or any combinations thereof. The non-volatile memory 108 may also include remote storage drives. The amount of time for the CPU 102 to access data stored in the non-volatile memory 108 may be slower relative to the amount of time it takes for the CPU 102 to access the cache 106.

[0014] The computing device 100 may also include a graphics processing unit (GPU) 112. As shown, the CPU 102 may be coupled through the bus 110 to the GPU 112. The GPU 112 may be configured to perform any number of graphics

operations within the computing device 100. For example, the GPU 112 may be configured to render or manipulate graphics images, graphics frames, videos, or the like, to be displayed to a user of the computing device 100. The computing device 100 may also include a memory controller 114. The memory controller may include a flush assist 115, which is a set of hardware that is to provide a pre-flush of the cache when a flush instruction from the core is detected as further described below. In some embodiments, the memory controller is a home agent for the non-volatile memory 108. As used herein, a home agent is an entity that controls access to the non-volatile memory 108. The home agent can keep track of the status of data stored at the non-volatile memory 108.

[0015] The CPU 102 may also be connected through the bus 110 to an input/output (I/O) device interface 116 configured to connect the computing device 100 to one or more I/O devices 118. The I/O devices 118 may include, for example, a keyboard and a pointing device, wherein the pointing device may include a touchpad or a touchscreen, among others. The I/O devices 118 may be built-in components of the computing device 100, or may be devices that are externally connected to the computing device 100.

[0016] The CPU 102 may be linked through the bus 110 to a display interface 120 configured to connect the computing device 100 to display devices 122. The display devices 122 may include a display screen that is a built-in component of the computing device 100. The display devices 122 may also include a computer monitor, television, or projector, among others, that is externally connected to the computing device 100. The computing device 100 may also include a network interface controller (NIC) 124 may be configured to connect the computing device 100 through the bus 110 to a network 126. The network 126 may be a wide region network (WAN), local region network (LAN), or the Internet, among others.

[0017] The block diagram of Fig. 1 is not intended to indicate that the computing device 100 is to include all of the components shown in Fig. 1. Further, the computing device 100 may include any number of additional components not shown in Fig. 1, depending on the details of the specific implementation.

[0018] In examples, the non-volatile memory may be mapped for direct load-store access by the CPU. In a load-store architecture, the CPU may access a memory

address of the non-volatile memory via load and store operations. Values for an operation can be loaded from the non-volatile memory and placed in the cache. Following the operation, the result is to be stored back to the non-volatile memory. In the load-store model, CPU performance can largely on the speed at which data is transferred between the cache and non-volatile memory. Most CPU caches are designed to function with pressure-eviction as the primary method of writing modified data to memory. Thus, there is little control of the precise moment when modification to non-volatile stored data (i.e. persistence) occurs. However, data integrity concerns, such as database and file system Atomicity, Consistency, Isolation, Durability (ACID) requirements depend upon some measure of control of write commitment timing. By pre-flushing the cache, the modification of the non-volatile store data can be performed quick enough to meet the database and file system ACID requirements.

[0019] Fig. 2A is a diagram of a system 200A with data transfers in a load-store model as described herein. The system 200A includes a core 202, a cache 204, a home agent (HA) 206, and a memory 208. The core 202 may be a processing unit, such as the CPU 102 (Fig. 1) described above. Similarly, the cache may be the cache 106 (Fig. 1) described above. The home agent 206 is to control access to the memory 208. In some embodiments, the home agent 206 is a memory controller 114 (Fig. 1) as described above. Moreover, the memory 208 may be a non-volatile memory 108 (Fig. 1) as described above.

[0020] Under normal operations, the core 202 issues a flush 210A to the cache 204. In response to the flush 210A from the core 202, the cache 204 proceeds to execute the flush by issuing a dirty write back flush 210B to the home agent 206. The home agent 206 then writes the data to the memory 208 with a write 210C. Upon completion of the write 210C, the memory 208 issues a write done 210D to the home agent 206 to indicate that the write is complete. Subsequently, the home agent issues a complete 210E to the cache 204. The cache 204 then issues a complete 210F to the core 202. A flush 212A is sent to the cache 204 upon receipt of the complete 210F, which indicates that the flush 210A instruction has completed. Thus, the flush 212A waits for completion 210F of the flush 210A. Likewise, the flush 214A waits for the completion 212F of the flush 212A, and the flush 216A waits

for the completion 214F of the flush 214A. Each of the flush 210A, 212A, 214A, and 216A are illustrated as ordered and occurring one at a time for ease of description. However, the flushes may be performed in parallel. The number of flushes performed in parallel may be determined according to the capabilities of the core 202.

[0021] As illustrated, the time that it takes to complete each of the flush 210A, 212A, 214A, and 216A are dependent in part upon a dirty write back flush 210B, 212B, 214B, and 216B. A dirty area of the cache is one that has been updated by the CPU. The dirty area of the cache is incoherent with the corresponding data address in memory. A dirty write back occurs to update the corresponding data address with data from the cache, thus creating a coherent region of the cache. The core 202 issues flush instructions that walks through an area of memory, region by region, in order to create cache coherency. Writes to non-volatile memory, such as each write 210C, 212C, 214C, and 216C can consume a large amount. The write latency of non-volatile memory is several orders of magnitude higher than that of volatile memory. In a load-store architecture, the write latency that occurs with dirty write backs can cause the system to fail to meet database ACID requirements. The ACID requirements are a set of properties that guarantee that database transactions are processed reliably.

[0022] Dirty write backs are a normal part of volatile memory operations. Moreover, snoops are a normal part of volatile memory operations. Processors typically support high snoop rates along with many concurrent snoops and many concurrent resulting dirty write backs. In some embodiments, a snoop is a speculative technique where cache activity is monitored in order to locate any activity that may cause regions of the cache to become invalid according to a cache protocol. The snoops can be applied to a cache flushing to non-volatile memory, and a pre-flush can be performed based on the snoop.

[0023] Fig. 2B is a diagram of a system 200B with data transfers with a hardware flush assist in a load-store model as described herein. The system 200B is similar to Fig. 2A, and includes a core 202, a cache 204, a home agent (HA) 206, and a memory 208 as described above. The home agent 206 monitors the cache for activity that can cause incoherent regions. When the core 202 issues the flush

210A, the home agent 206 may also issue one or more snoops. As illustrated, the home agent 206 issues snoop 212G, snoop 214G, and snoop 216G. Each snoop is directed to a region of memory offset from the address of the flush 210A. In some embodiments, the snoop is offset from the address of the flush 210A such that if the core is sequentially stepping though an area of memory, region by region, where all regions are clean, it traverses the address from the flush 210A to the offset in the time it takes for a dirty flush to complete. While the snoop 212G, snoop 214G, and snoop 216G are illustrated separately, the snoops may occur in parallel or at the same time.

[0024] Accordingly, while the dirty write back flush 210B and write 210C are executed, the snoops 212G, 214G, and 216G cause a series of speculative memory writes. Reference number 220 illustrates data transfers as a result of the snoops 212G, 214G, and 216G. The dirty write back flush 212B, 214B, and 216B each occur as a result of their respective snoop. A set of memory writes 222 is to pre-flush the regions of memory that would be subsequently flushed by the flush 212A, flush 214A, and flush 216A. The memory writes 222 includes the writes 212C, 214C, and 216C. The series of memory writes 222 results in a series of write dones 224. The write dones 224 includes each write done 210D, 212D, 214D, and 216D. Since the write done 210C has issued, the home agent issues a complete 210E to the cache 204. The cache 204 then issues a complete 210F to the core 202.

[0025] A flush 212A is sent to the cache 204 upon receipt of the complete 210F, which indicates that the flush 210A instruction has completed. However, the area of the cache to be flushed according to the core 202 instruction has been previously flushed via the snoop 212G, memory writes 222, and write dones 224. A flush clean 212G instruction is sent to the home agent 206, which can immediately return the complete 212E since a series of write dones 224 has previously indicated that the instructed write to memory 208 is complete. In this manner, each flush instruction from the CPU encounters a pre-flushed region of memory that is clean. No write to non-volatile memory is performed, and the cache is made coherent in less time, without non-volatile memory latency being associated with each flush instruction on a dirty region of the cache. Note that Fig. 2B illustrates a single flush instructions for

ease of description. As noted above, several flush instructions may be performed in parallel.

[0026] Fig. 3 is a process flow diagram of a method 300 for a hardware flush assist. At block 302, a flush instruction is detected a flush instruction at address N. Flush instructions result in activity visible to the home agent, as the home agent is responsible for issuing snoops to other caches in other parts of the system. Accordingly, the flush activity is a trigger for snoops according to the present techniques. In some embodiments, a hardware engine is located in the memory controller, which is also the home agent for the memory region. The hardware engine takes advantage of the high-performance, highly concurrent snoop capabilities, to assist software flush loops, by "pre-flushing" ahead of the software using snoop flows. When the home agent receives a flush for address N (the instruction may cause a write if dirty or an invalidate if clean), it performs the flush as normal. This flush is blocking, where the underlying flush instruction blocks until persistence is achieved.

[0027] At block 304, a memory location at the address N plus an offset is snooped. When the home agent receives a flush for address N, it also snoops the cache line at another address N+OFFSET, and writes any resulting dirty data to non-volatile memory. This snoop/write back is non-blocking. It is speculative, designed to pre-clean or pre-flush the memory ahead of the future stream of software-initiated Clean or flush instructions. If the software is sequentially flushing a region of memory, by the time it reaches and attempts to flush address N+OFFSET, it is most likely to encounter an already- clean cache line, because the prior speculative snoop has was issued earlier when cache line N was flushed. Thus, for sequential flushes or cleans of large areas, software flush loops encounter mostly already-clean cache lines. Clean cache line flushes proceed faster than dirty ones, because they do not have to wait for the non-volatile memory device write latency.

[0028] The value of OFFSET is chosen such that, when a software flush loop is progressing sequentially through a memory region for which all cache lines are already clean, it will traverse the range from address N to address N+OFFSET in approximately the amount of time that it takes for a dirty flush to complete to non-volatile memory. This allows the speculative snoops to pre-flush memory just far

enough ahead of the software loop that cache lines are seldom ever hit by flush instructions while still dirty.

[0029] At block 306, a region of memory at the address N plus the offset is pre-flushed. The primary mechanism by which this invention speeds a flush operation, then, is by allowing the software to see mostly low-latency fast flushing clean cache lines, while the home-agent hardware behind the scenes is pre-flushing dirty cache lines before the software flush loop ever reaches them.

[0030] In this manner, the techniques described herein reduce the time associated with cache flushes, without affecting the native flush instructions. The hardware techniques disclosed assists with the performance of software that is coded using only the native flush instructions. In this manner, there is no disruptive recoding of software necessary. Moreover, this approach also works with native flush instructions that have been specifically tailored for load-store mapped non-volatile memory.

[0031] While the present techniques may be susceptible to various modifications and alternative forms, the embodiments discussed above have been shown only by way of example. However, it should again be understood that the techniques are not intended to be limited to the particular embodiments disclosed herein. Indeed, the present techniques include all alternatives, modifications, and equivalents falling within the true spirit and scope of the appended claims.

CLAIMS

What is claimed is:

1. An apparatus for a hardware flush assist, comprising:
   a processing element, wherein the processing element is to probe a cache line at an offset address and write the cache line at the offset address to a non-volatile memory in response to a flush instruction at a first address.

2. The apparatus of claim 1, wherein the offset address is to be calculated as an offset of an address N received for a flush.

3. The apparatus of claim 1, wherein the processing element is a hardware engine.

4. The apparatus of claim 1, wherein probing the cache line is a snoop of the cache line.

5. The apparatus of claim 1, wherein flush instructions are monitored by the processing element.

6. The apparatus of claim 1, wherein the probe and write is non-blocking.

7. A system for a hardware flush assist, comprising:
   a cache;
   a memory;
   a home agent, wherein the home agent is to detect a flush instruction on the cache at an address; and
   a memory controller, wherein the memory controller comprises a hardware engine, and the hardware engine is to snoop a region of the cache at an offset of the address, and write the region to the memory at the offset of the address.

8. The system of claim 7, wherein the memory controller is the home agent.

9. The system of claim 7, wherein the home agent is to detect a plurality of flush instructions on the cache at a plurality of addresses, and the hardware engine is to snoop a region of the cache at an offset of each of the plurality of addresses, and write the region to the memory at the offset of each of the plurality of addresses.

10. The system of claim 7, wherein the offset is a value that is traversed across a clean region of the cache from the address to the offset plus the address in an amount of time that a dirty flush is completed at the non-volatile memory.

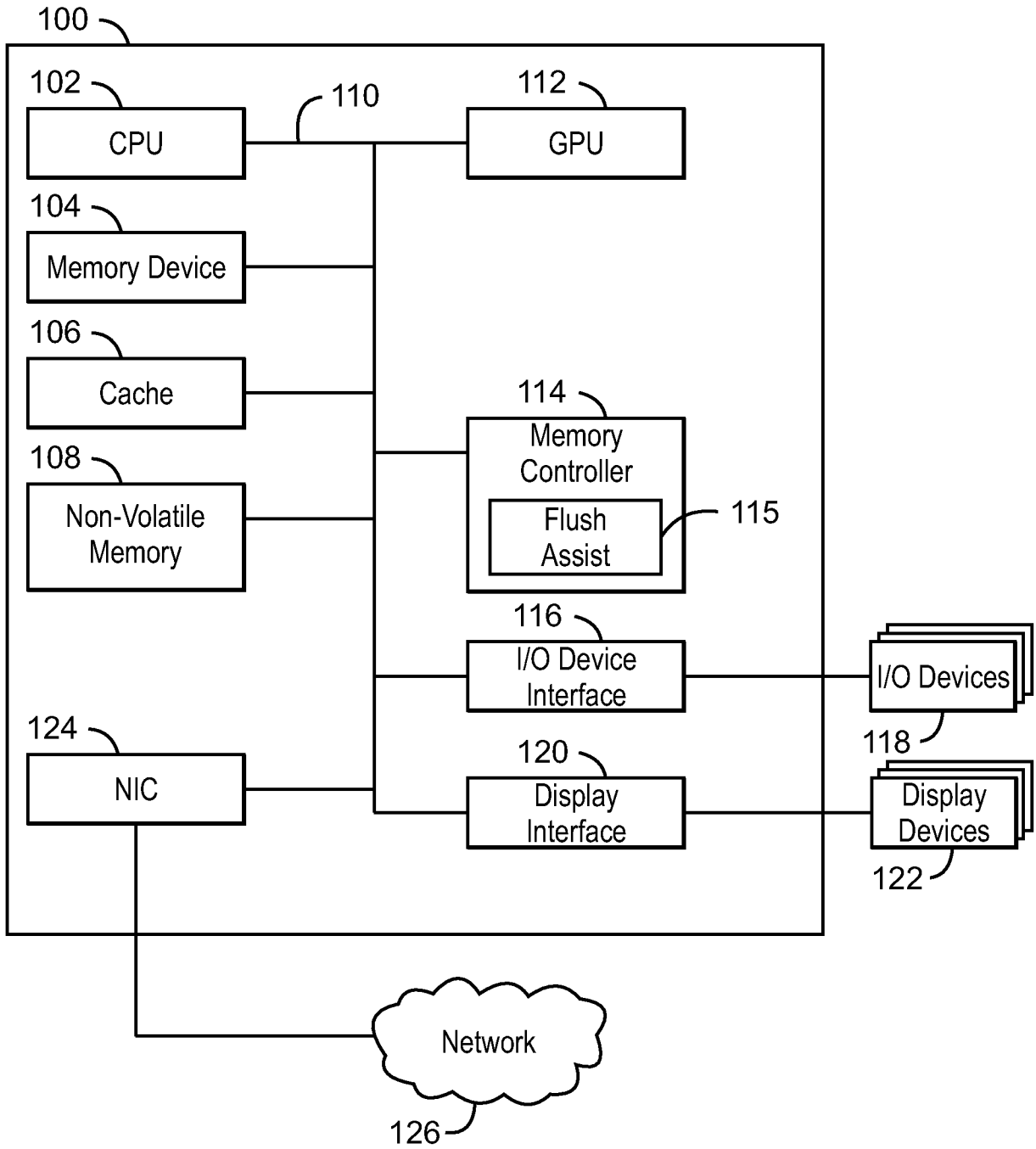
11. The system of claim 7, wherein the snoop is speculative.

12. A method for a hardware flush assist, comprising:
detecting a flush instruction at address N;
snooping a memory location at the address N plus an offset; and
pre-flushing a region of memory at the address N plus the offset.

13. The method of claim 12, wherein the offset is selected such that the flush instruction traverses a range of addresses in a clean region of memory from the address to the address plus the offset in the time that it takes a dirty flush to complete to non-volatile memory.

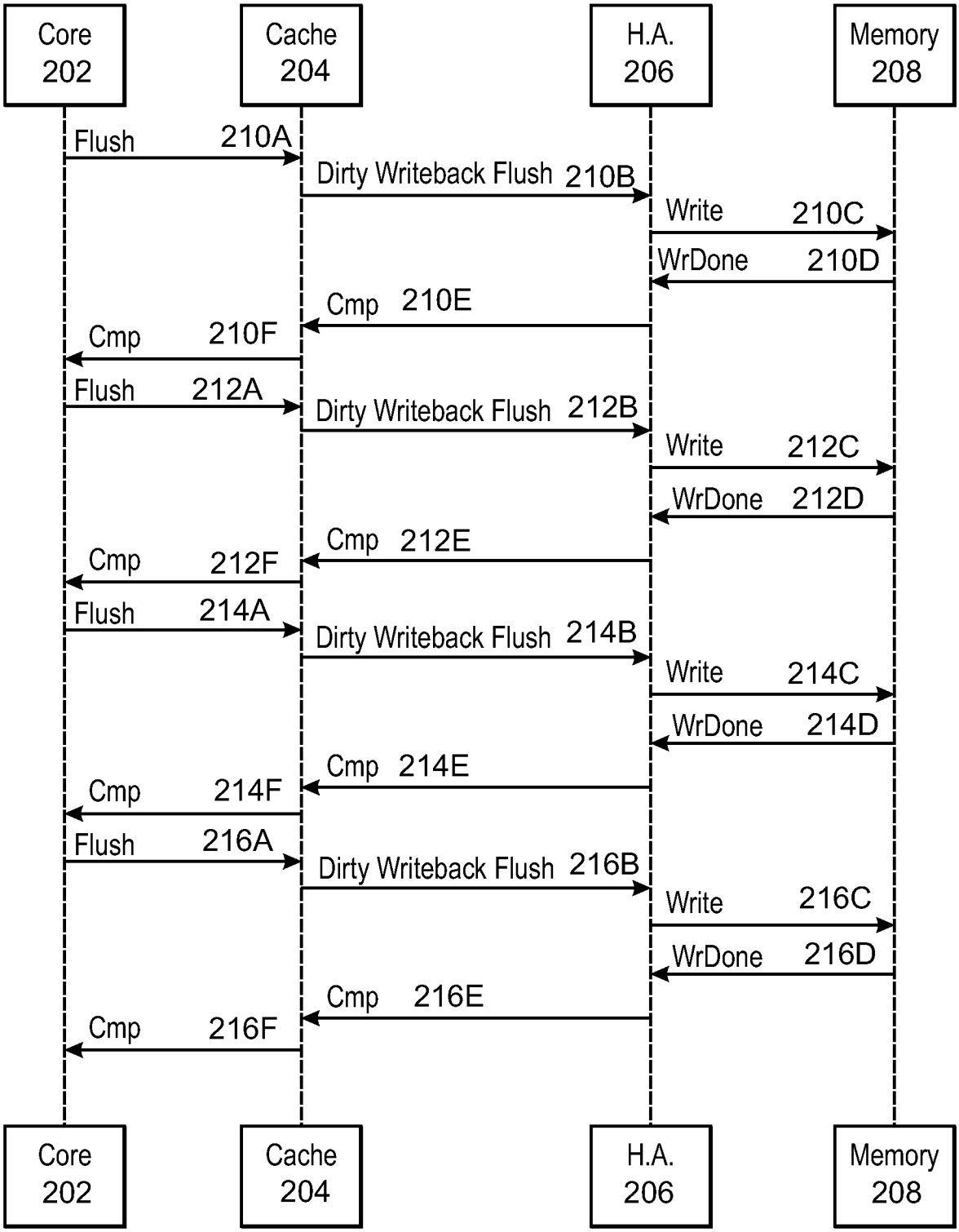
14. The method of claim 12, wherein pre-flushing the region of memory is performed prior to a next flush instruction directed the address plus the offset.

15. The method of claim 12, wherein the snooping is concurrent with the flush instruction.


100
102
CPU
110
112
GPU
104
Memory Device
106
Cache
108
Non-Volatile Memory
114
Memory Controller
Flush Assist
115
116
I/O Device Interface
I/O Devices
118
124
NIC
120
Display Interface
Display Devices
122
Network
126

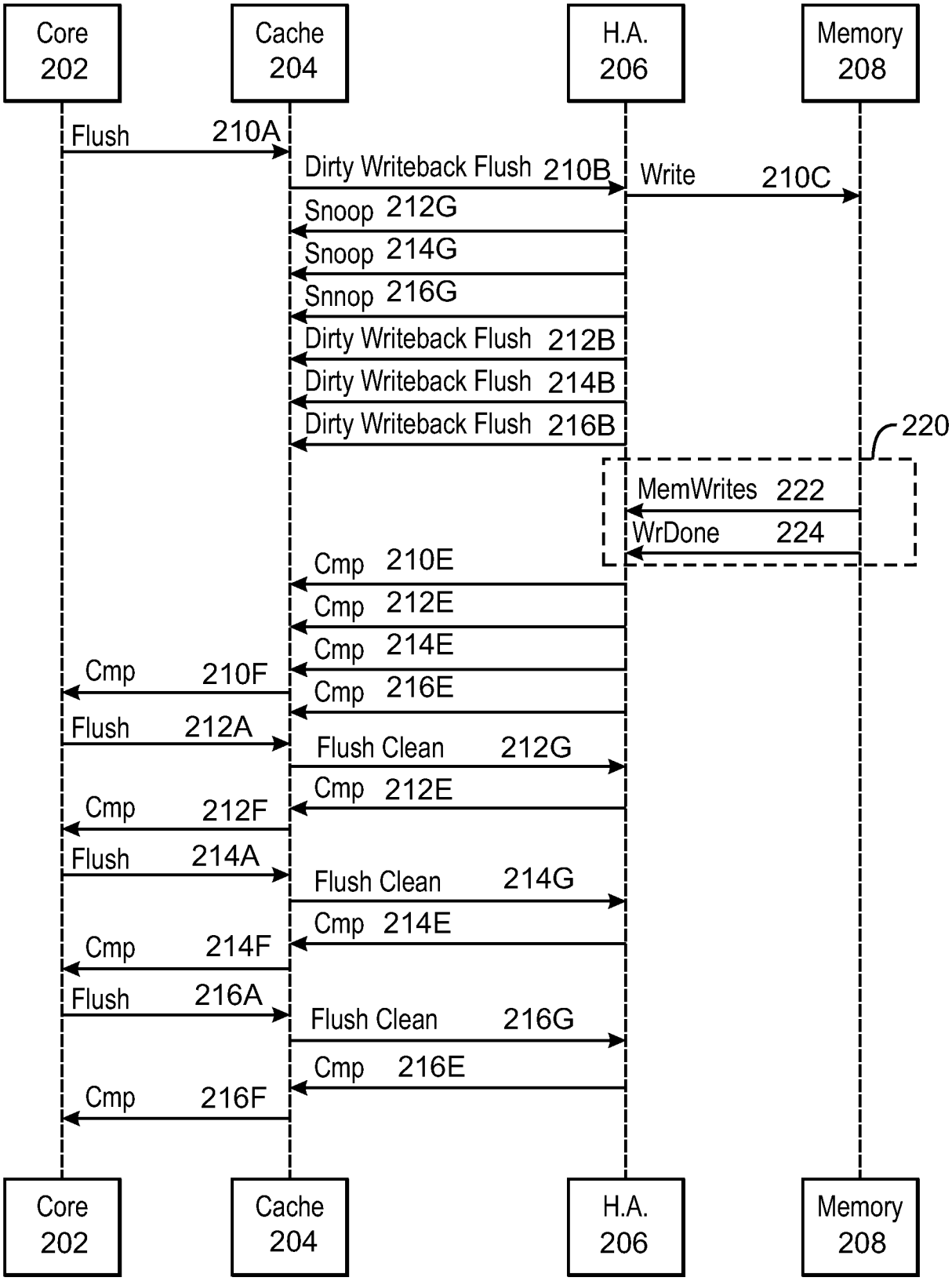
100

FIG. 1


Core 202
Cache 204
H.A. 206
Memory 208
Flush 210A
Dirty Writeback Flush 210B
Write 210C
WrDone 210D
Cmp 210E
Cmp 210F
Flush 212A
Dirty Writeback Flush 212B
Write 212C
WrDone 212D
Cmp 212E
Cmp 212F
Flush 214A
Dirty Writeback Flush 214B
Write 214C
WrDone 214D
Cmp 214E
Cmp 214F
Flush 216A
Dirty Writeback Flush 216B
Write 216C
WrDone 216D
Cmp 216E
Cmp 216F
Core 202
Cache 204
H.A. 206
Memory 208

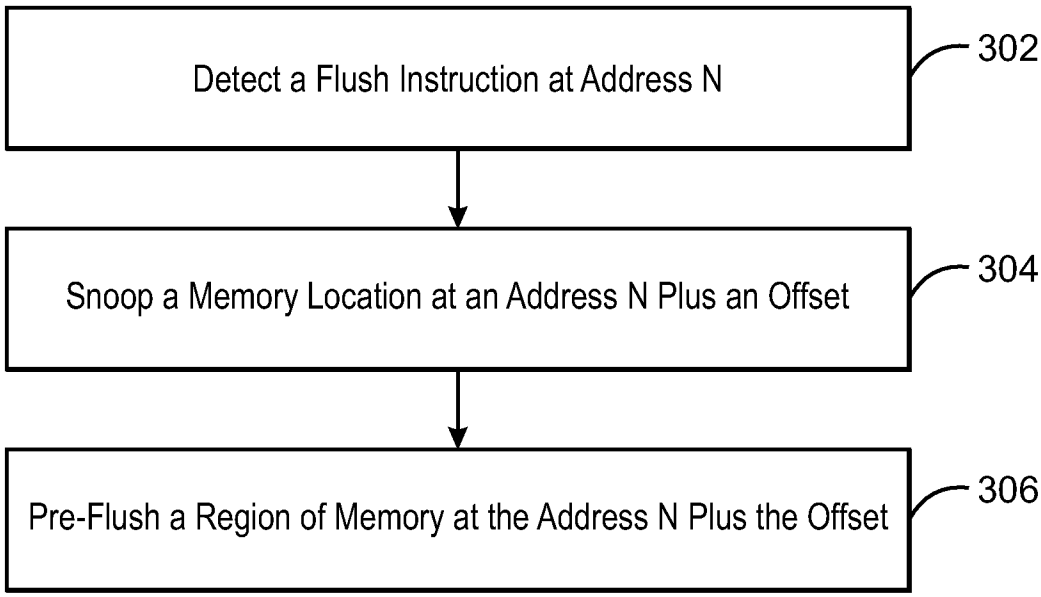
200A

FIG. 2A


Core 202
Cache 204
H.A. 206
Memory 208
Flush 210A
Dirty Writeback Flush 210B
Write 210C
Snoop 212G
Snoop 214G
Snnop 216G
Dirty Writeback Flush 212B
Dirty Writeback Flush 214B
Dirty Writeback Flush 216B
220
MemWrites 222
WrDone 224
Cmp 210E
Cmp 212E
Cmp 214E
Cmp 216E
Cmp 210F
Flush 212A
Flush Clean 212G
Cmp 212E
Cmp 212F
Flush 214A
Flush Clean 214G
Cmp 214E
Cmp 214F
Flush 216A
Flush Clean 216G
Cmp 216E
Cmp 216F
Core 202
Cache 204
H.A. 206
Memory 208

200B

FIG. 2B


Detect a Flush Instruction at Address N
302
Snoop a Memory Location at an Address N Plus an Offset
304
Pre-Flush a Region of Memory at the Address N Plus the Offset
306

300

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/042500

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/08(2006.01)i, G06F 12/02(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)
G06F 12/08; G06F 12/00; G06F 12/14; G06F 9/38; G06F 1/32; G06F 12/02

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
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: preflush, snoop, cache, address, offset memory, traverse, clean, region

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6643766 B1 (GREGG B LESARTRE et al.) 04 November 2003 See column 1 lines 15-19 and 56-59; column 3, lines 8-36; column 10, lines 55-59; claims 10-11; and figure 5.	1-15
Y	US 2004-0158681 A1 (RODNEY E. HOOKER) 12 August 2004 See paragraphs [0014], [0036]; and figure 7.	1-15
A	EP 1555611 A2 (IP-FIRST LLC) 20 July 2005 See paragraphs [0012], [0034]; and figure 2.	1-15
A	US 2014-0365736 A1 (LSI CORPORATION) 11 December 2014 See paragraphs [0004], [0031]; and figure 1.	1-15
A	US 2010-0235670 A1 (JAMES B. KELLER et al.) 16 September 2010 See paragraphs [0008], [0032]; and figure 2.	1-15

☐ 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 another 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	Date of mailing of the international search report
24 April 2016 (24.04.2016)	**25 April 2016 (25.04.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
CHIN, Sang Bum
Telephone No. +82-42-481-8398

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2015/042500

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6643766 B1	04/11/2003	FR 2808902 A1	16/11/2001
		FR 2808902 B1	13/06/2008
US 2004-0158681 A1	12/08/2004	CN 1266589 C	26/07/2006
		CN 1514352 A	21/07/2004
		CN 1514352 C	26/07/2006
		EP 1447746 A2	18/08/2004
		EP 1447746 A3	29/06/2005
		EP 1447746 B1	22/04/2009
		TW 200415508 A	16/08/2004
		TW 226575 A	11/01/2005
		TW 226575 B	11/01/2005
		TW I226575 B	11/01/2005
		US 07000081 B2	14/02/2006
EP 1555611 A2	20/07/2005	CN 1641567 A	20/07/2005
		CN 1641567 C	06/02/2008
		EP 1555611 A3	25/01/2006
		EP 1555611 B1	13/08/2008
		TW 200525354 A	01/08/2005
		TW 275938 A	11/03/2007
		TW 275938 B	11/03/2007
		TW I275938 B	11/03/2007
		US 07139876 B2	21/11/2006
		US 2004-0148467 A1	29/07/2004
US 2014-0365736 A1	11/12/2014	None	
US 2010-0235670 A1	16/09/2010	US 07752474 B2	06/07/2010
		US 08171326 B2	01/05/2012
		US 2008-0077813 A1	27/03/2008