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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 May 2010 (20.05.2010)

PCT

(10) International Publication Number
WO 2010/056511 A2

(51) **International Patent Classification**:
G06F 9/06 (2006.01) *G06F 9/30* (2006.01)
G06F 9/22 (2006.01)

(21) **International Application Number**:
PCT/US2009/062219

(22) **International Filing Date**:
27 October 2009 (27.10.2009)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
12/290,395 30 October 2008 (30.10.2008) US

(71) **Applicant** *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95052 (US).

(72) **Inventors; and**
(75) **Inventors/Applicants** *(for US only)*: **OUZIEL, Ido** [IL/IL]; Haseora 28, Zrufa (IL). **RAPPOPORT, Lihu** [IL/IL]; 14 Vardiya st., Haifa (IL). **VALENTINE, Robert** [IL/IL]; Rechov Hadganiot, 33-5, 36054 Kiryat Tivon (IL). **GABOR, Ron** [IL/IL]; 10 Moshe Sne St., 43728 Raanana (IL). **RAGHUVANSHI, Pankaj** [IN/US]; 901 NW Brookhill St., Hillsboro, Oregon 97124 (US).

(74) **Agents**: **VINCENT, Lester, J.** et al.; Blakely Sokoloff Taylor & Zafman, 1279 Oakmead Parkway, Sunnyvale, California 94085 (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, 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, 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, PE, 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,

[Continued on next page]

(54) **Title**: TECHNIQUE FOR PROMOTING EFFICIENT INSTRUCTION FUSION

(57) **Abstract**: A technique to enable efficient instruction fusion within a computer system. In one embodiment, a processor logic delays the processing of a second instruction for a threshold amount of time if a first instruction within an instruction queue is fusible with the second instruction.

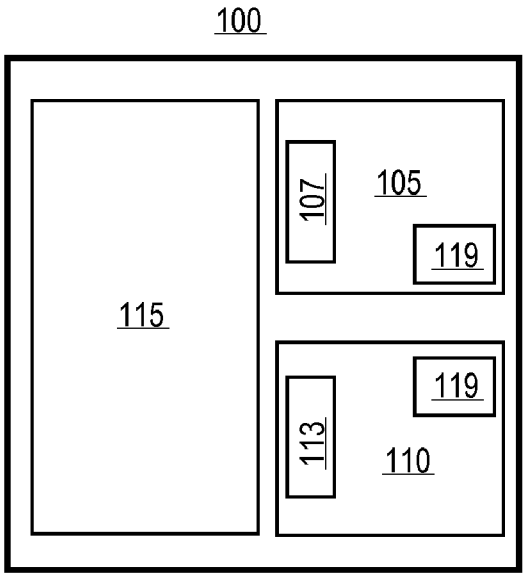

FIG. 1

WO 2010/056511 A2

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, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

TECHNIQUE FOR PROMOTING EFFICIENT INSTRUCTION FUSION

Field of the Invention

Embodiments of the invention relate generally to the field of information processing and more specifically, to the field of instruction fusion in computing systems and microprocessors.

Background

Instruction fusion is a process that combines two instructions into a single instruction which results in a one operation (or micro-operation, "uop") sequence within a processor. Instructions stored in a processor instruction queue (IQ) may be "fused" after being read out of the IQ and before being sent to instruction decoders or after being decoded by the instruction decoders. Typically, instruction fusion occuring before the instruction is decoded is referred to as "macro-fusion", whereas instruction fusion occuring after the instruction is decoded (into uops, for example) is referred to as "micro-fusion". An example of macro-fusion is the combining of a compare ("CMP") instruction or test instruction ("TEST") ("CMP/TEST") with a conditional jump ("JCC") instruction. CMP/TEST and JCC instruction pairs may occur regularly in programs at the end of loops, for example, where a comparison is made and, based on the outcome of a comparison, a branch is taken or not taken. Since macro-fusion may effectively increase instruction throughput, it may be desireable to find as many opportunities to fuse instructions as possible.

For instruction fusion opportunities to be found in some prior art processor microarchitectures, both the CMP/TEST and JCC instructions may need to reside in the IQ concurrently so that they can be fused when the instructions are read from the IQ. However, if there is a fusible CMP/TEST instruction in the IQ and no further instructions have been written to the IQ (i.e. the CMP/TEST instruction is the last instruction in the IQ), the CMP/TEST instruction may be read from the IQ and sent to the decoder without being fused, even if the next instruction in program order is a JCC instruction. An example where a missed fusion opportunity may occur is if the CMP/TEST and the JCC happen to be across a storage boundary (e.g., 16 byte boundary), causing the CMP/TEST to be written in the IQ in one cycle and the JCC to be written the following cycle. In this case, if there are no stalling conditions, the JCC will be written in the IQ at the same time or after the CMP/TEST is being read from the IQ, so a fusion opportunity will be missed,

resulting in multiple unnecessary reads of the IQ, reduced instruction throughput and excessive power consumption.

Brief Description of the Drawings

Embodiments of the invention are illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1 illustrates a block diagram of a microprocessor, in which at least one embodiment of the invention may be used;

Figure 2 illustrates a block diagram of a shared bus computer system, in which at least one embodiment of the invention may be used;

Figure 3 illustrates a block diagram a point-to-point interconnect computer system, in which at least one embodiment of the invention may be used;

Figure 4 illustrates a block diagram of a state machine, which may be used to implement at least one embodiment of the invention;

Figure 5 is a flow diagram of operations that may be used for performing at least one embodiment of the invention.

Figure 6 is a flow diagram of operations that may be performed in at least one embodiment.

Detailed Description

Embodiments of the invention may be used to improve instruction throughput in a processor and/or reduce power consumption of the processor. In one embodiment, what would be otherwise missed opportunities for instruction fusion or found and instruction fusion may occur as a result. In one embodiment, would-be missed instruction fusion opportunities are found by delaying reading of a last instruction from an instruction queue (IQ) or the issuance of the last instruction read from the IQ to a decoding phase for a threshold number of cycles, so that any subsequent fusible instructions may be fetched and stored in the IQ (or at least identified without necessarily being stored in the IQ) and subsequently fused with the last fusible instruction. In one embodiment, delaying the reading or issuance of a first fusible instruction by a threshold number of cycles may improve processor performance, since doing so may avoid two, otherwise fusible, instructions being decoded and processed separately rather than as a single instruction.

The choice of the threshold number of wait cycles may depend upon the microarchitecture in which a particular embodiment is used. For example, in one

embodiment, the threshold number of cycles may be two, whereas in other embodiments, the threshold number of cycles may be more or less than two. In one embodiment, the threshold number of wait cycles provides the maximum amount of time to wait on a subsequent fusible instruction to be stored to the IQ while maintaining an overall latency/performance advantage in waiting for the subsequent fusible instruction over processing the fusible instructions as separate instructions. In other embodiments, where power is more critical, for example, the threshold number of wait cycles could be higher in order to ensure that extra power is not used to process the two fusible instructions separately, even though the number of wait cycles may cause a decrease (albeit temporarily) in instruction throughput.

Figure 1 illustrates a microprocessor in which at least one embodiment of the invention may be used. In particular, Figure 1 illustrates microprocessor 100 having one or more processor cores 105 and 110, each having associated therewith a local cache 107 and 113, respectively. Also illustrated in Figure 1 is a shared cache memory 115 which may store versions of at least some of the information stored in each of the local caches 107 and 113. In some embodiments, microprocessor 100 may also include other logic not shown in Figure 1, such as an integrated memory controller, integrated graphics controller, as well as other logic to perform other functions within a computer system, such as I/O control. In one embodiment, each microprocessor in a multi-processor system or each processor core in a multi-core processor may include or otherwise be associated with logic 119 to enable interrupt communication techniques, in accordance with at least one embodiment. The logic may include circuits, software or both to enable more efficient fusion of instructions than in some prior art implementations.

In one embodiment, logic 119 may include logic to reduce the likelihood of missing instruction fusion opportunities. In one embodiment, logic 119 delays the reading of a first instruction (e.g., CMP) from the IQ, when there is no subsequent instruction stored in the IQ or other fetched instruction storage structure. In one embodiment, the logic 119 causes the reading or issuance of a first fusible instruction for a threshold number of cycles (e.g., two cycles) before reading the IQ or issuing the first fusible instruction to a decoder or other processing logic, such that if there is a second fusible instruction that can be fused with the first instruction not yet stored in the IQ (due, for example, to the two fusible instructions being stored in a memory or cache in different storage boundaries), the opportunity to fuse the two fusible instructions may not be

missed. In some embodiments, the threshold may be fixed, whereas in other embodiments, the threshold may be variable, modifiable by a user or according to user-independent algorithm. In one embodiment, the first fusible instruction is a CMP instruction and the second fusible instruction is a JCC instruction. In other embodiments, either or both of the first and second instruction may not be a CMP or JCC instruction, but any fusible instructions. Moreover, embodiments if the invention may be applied to fusing more than two instructions.

Figure 2, for example, illustrates a front-side-bus (FSB) computer system in which one embodiment of the invention may be used. Any processor 201, 205, 210, or 215 may access information from any local level one (L1) cache memory 220, 225, 230, 235, 240, 245, 250, 255 within or otherwise associated with one of the processor cores 223, 227, 233, 237, 243, 247, 253, 257. Furthermore, any processor 201, 205, 210, or 215 may access information from any one of the shared level two (L2) caches 203, 207, 213, 217 or from system memory 260 via chipset 265. One or more of the processors in Figure 2 may include or otherwise be associated with logic 219 to enable improved efficiency of instruction fusion, in accordance with at least one embodiment.

In addition to the FSB computer system illustrated in Figure 2, other system configurations may be used in conjunction with various embodiments of the invention, including point-to-point (P2P) interconnect systems and ring interconnect systems. The P2P system of Figure 3, for example, may include several processors, of which only two, processors 370, 380 are shown by example. Processors 370, 380 may each include a local memory controller hub (MCH) 372, 382 to connect with memory 32, 34. Processors 370, 380 may exchange data via a point-to-point (PtP) interface 350 using PtP interface circuits 378, 388. Processors 370, 380 may each exchange data with a chipset 390 via individual PtP interfaces 352, 354 using point to point interface circuits 376, 394, 386, 398. Chipset 390 may also exchange data with a high-performance graphics circuit 338 via a high-performance graphics interface 339. Embodiments of the invention may be located within any processor having any number of processing cores, or within each of the PtP bus agents of Figure 3. In one embodiment, any processor core may include or otherwise be associated with a local cache memory (not shown). Furthermore, a shared cache (not shown) may be included in either processor outside of both processors, yet connected with the processors via p2p interconnect, such that either or both processors' local cache information may be stored in the shared cache if a processor is placed into a low power

mode. One or more of the processors or cores in Figure 3 may include or otherwise be associated with logic 319 to enable improved efficiency of instruction fusion, in accordance with at least one embodiment.

In at least one embodiment, a second fusible instruction may not be stored into an IQ before some intermediate operation occurs (occuring between a first and second fusible instruction), such as an IQ clear operation, causing a missed opportunity to fuse the two otherwise fusible instructions. In one embodiment, in which a cache (or a buffer) stores related sequences of decoded instructions (after they were read from the IQ and decoded) or uops (e.g., "decoded stream buffer" or "DSB", "trace cache", or "TC") that are to be scheduled (perhaps multiple times) for execution by the processor, a first fusible uop (e.g., CMP) may be stored in the cache without a fusible second uop (e.g., JCC) within the same addressable range (e.g., same cache way). This may occur, for example, where JCC is crossing a cache line (due to a cache miss) or crossing page boundary (due to a translation look-aside buffer miss), in which case the cache may store the CMP without the JCC. Subsequently, if the processor core pipeline is cleared (due to a "clear" signal being asserted, for example) after the CMP was stored but before the JCC is stored in the cache, the cache store only the CMP in one of its ways without the JCC.

On subsequent lookups to the cache line storing the CMP, the cache may interpret the missing JCC as a missed access and the JCC may be marked as the append point for the next cache fill operation. This append point, however, may not be found since the CMP+JCC may be read as fused from the IQ. Therefore, the requested JCC may not match any uop to be filled, coming from the IQ, and thus the cache will not be able to fill the missing JCC, but may continually miss on the line in which the fused CMP+JCC is expected. Moreover, in one embodiment in which a pending fill request queue (PFRQ) is used to store uop cache fill requests, an entry that was reserved for a particular fused instruction fill may not deallocate (since the expected fused instruction fill never takes place) and may remain useless until the next clear operation. In one embodiment, a PFRQ entry lock may occur every time the missing fused instruction entry is acessed, and may therefore prevent any subsequent fills to the same location.

In order to prevent an incorrect or undesireable lock of the PFRQ entry, a state machine, in one embodiment, may be used to monitor the uops being read from the IQ to detect cases, in which a region that has a corresponding PFRQ entry (e.g., a region marked for a fill) was completely missed, due for example, to the entry's last uop being reached

without the fill start point being detected. In one embodiment, the state machine may cause the PFRQ entry to be deallocated when this condition is met. In other embodiments, an undesireable PFRQ entry lock may be avoided by not creating within the cache a fusible instruction that may be read from the IQ without both fusible instructions present. For example, if a CMP is followed by a non-JCC instruction, a fused instruction entry may be created in the cache, but only if the CMP is read out of the IQ alone (after the threshold wait time expires, for example), is the fused instruction entry not filled to the cache. In other embodiments, the number of times the state machine has detected a fill region that was skipped may be counted, a cache flush or invalidation operation may be performed after some threshold count of times the fill region was skipped. The fill region may then be removed from the cache, and the fused instruction may then be re-filled.

Figure 4 illustrates a state machine, according to one embodiment, that may be used to avoid unwanted PFRQ entry lock conditions due to a missed fusible instruction in the IQ. At state 401, in which the instructions in the IQ are not in a region marked for fill, a "fill region start" signal indicating that the IQ is about to process an instruction that is mapped to a fill-region (an instruction from the fill region according to the cache hashing) but does not start at the linear instruction pointer saved in the PFRQ ("lip") 405. this may cause the state machine to move to state 410. If the next instruction in the IQ (that will soon be decoded) ends a fill region (e.g. ends a line as hashed by the cache, or is a taken branch), then the state machine causes the deallocation 415 of the corresponding PFRQ entry and the state machine returns to state 401. If, however, the fill pointer is equal to the fill region lip 430 while either in state 401 or state 410, the state machine enters state 420, in which the access is within the fill region and after fill start point. From state 420, a last uop in the fill region indication will return 425 the state machine to state 401 without deallocation the corresponding PFRQ entry. The state machine of Figure 4 may be implemented in hardware logic, software, or some combination thereof. In other embodiments, other state machines or logic may be used.

Figure 5 illustrates a flow diagram of operations that may be used in conjunction with at least one embodiment of the invention. At operation 501, it is determined whether the currently accessed instruction in the IQ is fusible with any subsequent instruction. If not, then at operation 505, the next instruction is accessed from the IQ and the delay count is reset. If so, then at operation 510, a delay counter is incremented and at operation 515 it is determined whether the delay count threshold is reached. If it isn't, then at operation

520, no instruction fusion of the currently accessed instruction is performed. If it is, then the next instruction is accessed from the IQ and the delay count is reset at operation 505. In other embodiments, other operations may be performed to improve the efficiency of instruction fusion.

Figure 6 illustrates a flow diagram of operations that may be performed in conjunction with at least one embodiment. In order to perform one embodiment in processors having a number of decoder circuits, it may be helpful to ensure that the first fusible instruction is to be decoded on a particular decoder circuit, which is capable of decoding the fused instruction. In Figure 6, it is determined whether a particular instruction can be a first of a fused pair of instructions at operation 601. If not, then the fused instructions are issued at operation 605. If so, then it is determined whether the first fusible instruction is followed by a valid instruction in the IQ at operation 610. If so, then the fused instructions are issued at operation 610. If not, then at operation 615, it is determined whether the first fusible instruction is to be issued to a decoder capable of supporting the fused instruction. In one embodiment, decoder-0 is capable of decoding the fused instructions. If the first fusible instruction was not issued to decoder-0, then at operation 620, the first fusible instruction is moved, or "nuked", to a different decoder until it corresponds to decoder-0. At operation 625, a counter is set to an initial value, N and at operation 630, if the instruction is followed by a valid instruction or the counter is zero, then the fused instructions are issued at operation 635. Otherwise, at operation 640, the counter is decremented and the invalid instruction is nuked. In other embodiments, the counter may increment to a final value. In other embodiments, other operations, besides a nuke operation may clear the invalid instruction.

One or more aspects of at least one embodiment may be implemented by representative data stored on a machine-readable medium which represents various logic within the processor, which when read by a machine causes the machine to fabricate logic to perform the techniques described herein. Such representations, known as "IP cores" may be stored on a tangible, machine readable medium ("tape") and supplied to various customers or manufacturing facilities to load into the fabrication machines that actually make the logic or processor.

Thus, a method and apparatus for directing micro-architectural memory region accesses has been described. It is to be understood that the above description is intended to be illustrative and not restrictive. Many other embodiments will be apparent to those of

skill in the art upon reading and understanding the above description. The scope of the invention should, therefore, be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

CLAIMS

What is claimed is:

1. An apparatus comprising:
   an instruction queue (IQ);
   logic to delay processing of a first fusible instruction for a threshold amount of time, such that a second fusible instruction, fusible with the first fusible instruction, may be fused with the first fusible instruction if the second fusible instruction is stored within the IQ within the threshold amount of time.
2. The apparatus of claim 1, wherein the first fusible and the second fusible instructions are stored across a fetch boundary prior to being stored in the IQ.
3. The apparatus of claim 1, wherein the logic is to delay processing of the first fusible instruction only if the first fusible instruction is the last instruction stored in the IQ.
4. The apparatus of claim 1, wherein the logic includes a counter to be incremented once for each cycle after the first fusible instruction is stored in the IQ and is the last instruction in the IQ until a threshold number of cycles corresponding to the threshold amount of time is reached.
5. The apparatus of claim 1, further comprising a state machine to prevent a fill buffer request queue (FBRQ) from locking an entry corresponding to the first and second fusible instructions, if an intermediate instruction is performed between the first fusible instruction being stored in the IQ and the second fusible instruction being stored in the IQ.
6. The apparatus of claim 5, wherein the intermediate instruction is to cause the IQ to be cleared.
7. A method comprising:
   determining whether the currently accessed instruction within an instruction queue (IQ) is fusible with any subsequent instruction to be stored in the IQ;
   accessing a next instruction from the IQ and resetting the delay counter if the currently accessed instruction is not fusible with a subsequent instruction to be stored in the IQ;
   incrementing the delay counter if a currently accessed instruction is fusible and is the last instruction in the IQ.

8. The method of claim 7 further comprising fusing the currently accessed instruction with the subsequent instruction if the first and second instructions are fusible and the delay counter has not reached a threshold value.
9. The method of claim 8, further comprising processing the currently accessed instruction separately from the subsequent instruction if the first and second instructions are not fusible.
10. The method of claim 8, further comprising processing the currently accessed instruction separately from the subsequent instruction the delay counter has reached the threshold value.
11. The method of claim 7, further comprising preventing a fill buffer request queue (FBRQ) from locking an entry corresponding to the currently accessed instruction and the subsequent instructions if they are fusible and an intermediate event is performed before subsequent instruction is stored in a cache and after the currently accessed instruction is stored in the cache.
12. A system comprising:
    a storage to store a first and second fusible instruction within a first and second access boundary, respectively;
    a processor having fetch logic to fetch the first and second fusible instruction into an instruction queue (IQ);
    delay logic to delay reading of the first fusible instruction from the IQ for a threshold amount of cycles;
    instruction fusion logic to fuse the first and second fusible instructions if the second fusible instruction is stored in the IQ after the first fusible instruction and before the threshold amount of cycles has been reached.
13. The system of claim 12, further comprising a counter to increment if the first fusible instruction is the only instruction in the IQ and to stop counting when the threshold amount of cycles has been reached.
14. The system of claim 13, wherein the counter is to reset if the second fusible instruction is stored in the IQ before the threshold amount of cycles has been reached.
15. The system of claim 13, wherein the counter is to reset if the second fusible instruction is stored in the IQ before the threshold amount of cycles has been reached.
16. The system of claim 12, wherein the storage includes an instruction cache and the first and second boundary sizes are each 64 bytes.

17. The system of claim 12, wherein the storage includes a dynamic random-access memory and the first and second boundary sizes are each 4096 bytes.
18. The system of claim 12, wherein the first fusible instruction is a CMP/TEST instruction and the second fusible instruction is a JCC instruction.
19. The system of claim 18, wherein the threshold number of cycles is two.
20. The system of claim 12, further including a state machine to prevent a fill buffer request queue (FBRQ) from locking an entry corresponding to the first and second fusible instructions, if an intermediate event is performed between the first fusible instruction being stored in the cache and the second fusible instruction being stored in the cache.

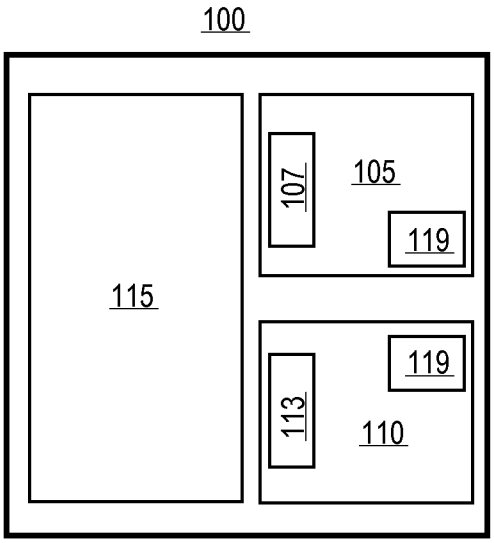
100
115
105
107
119
119
113
110

FIG. 1

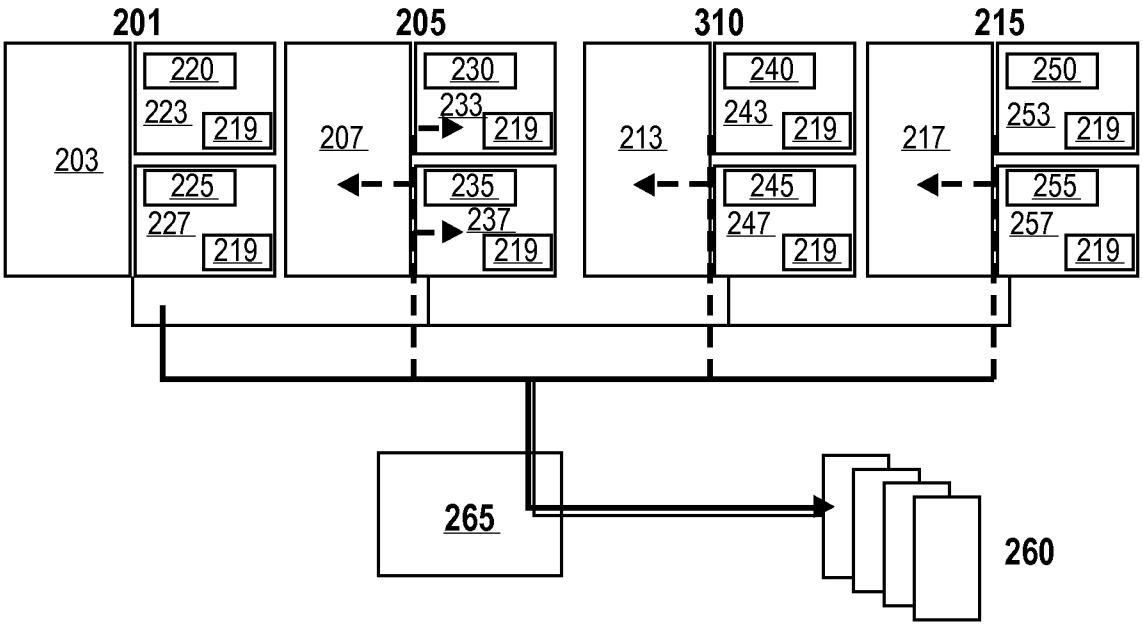
201
205
310
215
203
220
223
219
225
227
219
207
230
233
219
235
237
219
213
240
243
219
245
247
219
217
250
253
219
255
257
219
265
260

FIG. 2

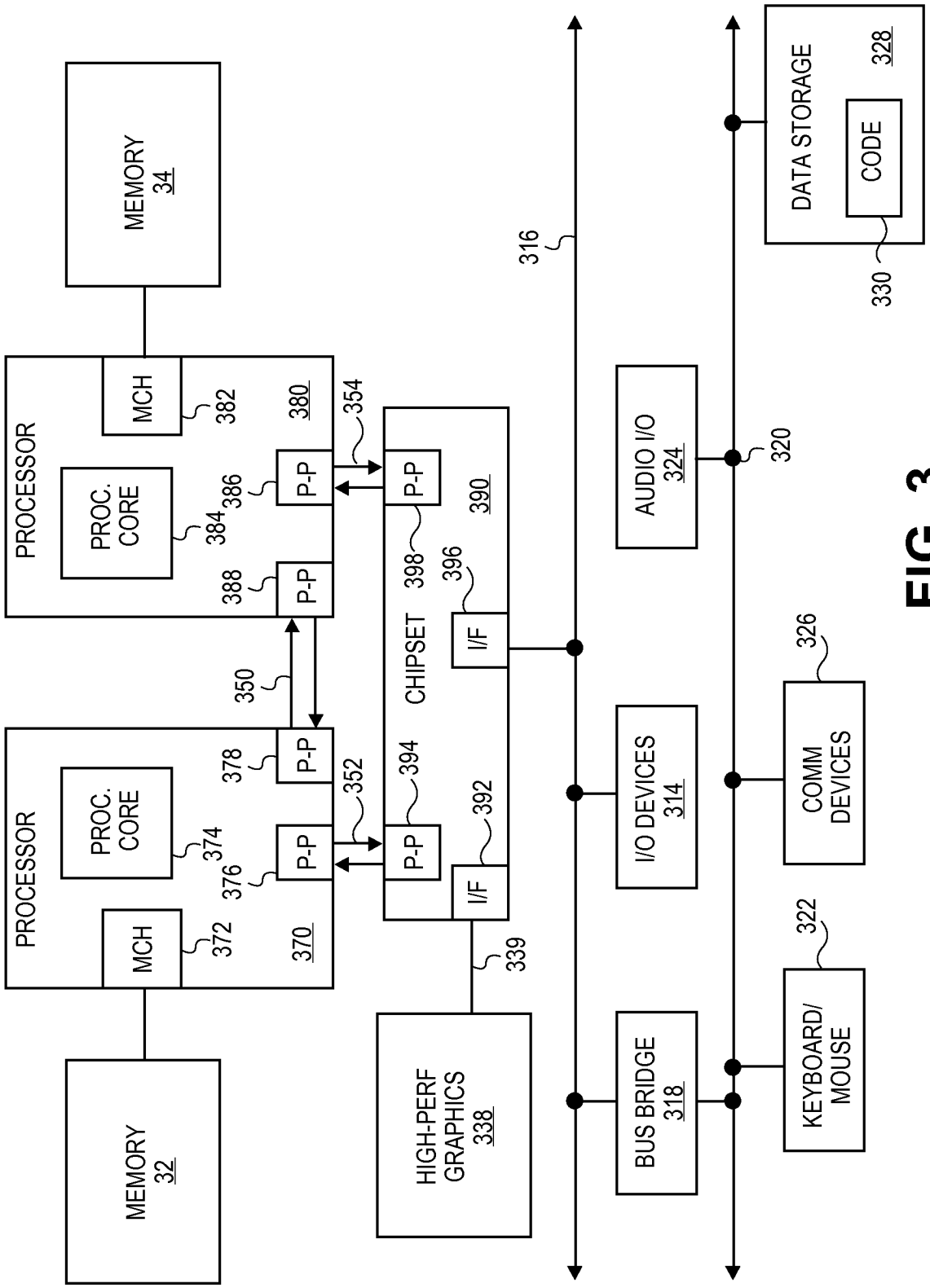
MEMORY 34
MCH
382
PROCESSOR
PROC. CORE
384
386
P-P
380
354
388
P-P
P-P
350
CHIPSET
398
396
I/F
390
316
DATA STORAGE
CODE
328
330
AUDIO I/O 324
320
378
P-P
352
394
PROCESSOR
PROC. CORE
374
376
P-P
P-P
I/F
392
370
MCH
372
339
I/O DEVICES 314
326
COMM DEVICES
HIGH-PERF GRAPHICS 338
BUS BRIDGE 318
322
KEYBOARD/ MOUSE
MEMORY 32

FIG. 3

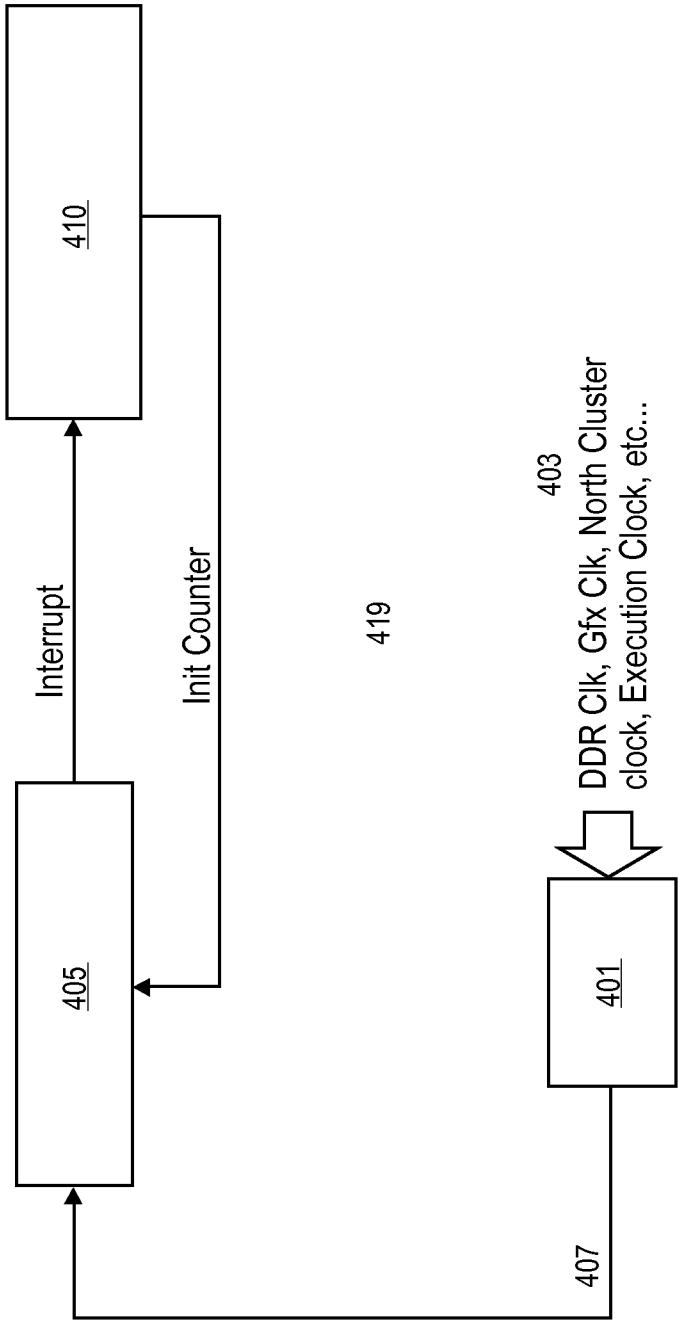
410
Interrupt
Init Counter
405
419
403
DDR Clk, Gfx Clk, North Cluster clock, Execution Clock, etc...
401
407

FIG. 4

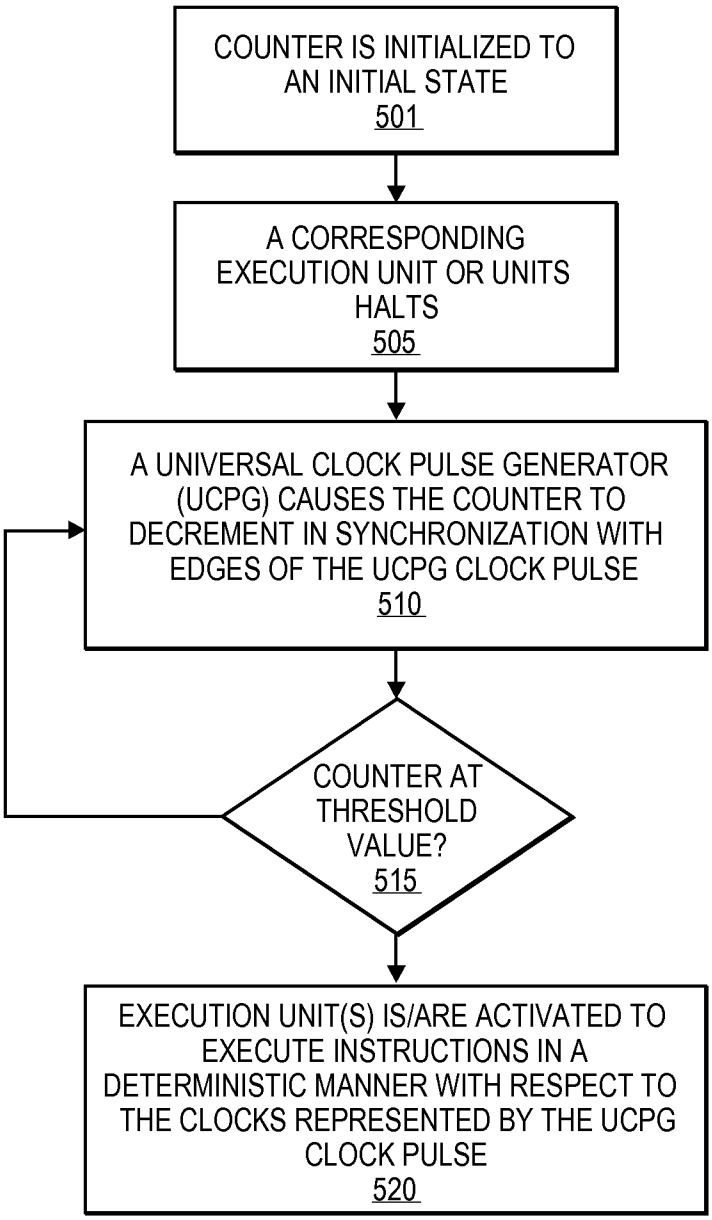
COUNTER IS INITIALIZED TO AN INITIAL STATE
501
A CORRESPONDING EXECUTION UNIT OR UNITS HALTS
505
A UNIVERSAL CLOCK PULSE GENERATOR (UCPG) CAUSES THE COUNTER TO DECREMENT IN SYNCHRONIZATION WITH EDGES OF THE UCPG CLOCK PULSE
510
COUNTER AT THRESHOLD VALUE?
515
EXECUTION UNIT(S) IS/ARE ACTIVATED TO EXECUTE INSTRUCTIONS IN A DETERMINISTIC MANNER WITH RESPECT TO THE CLOCKS REPRESENTED BY THE UCPG CLOCK PULSE
520

FIG. 5