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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 August 2014 (28.08.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/130037 A1

(51) International Patent Classification:
G06F 12/08 (2006.01) *G06F 12/14* (2006.01)

(21) International Application Number:
PCT/US2013/027118

(22) International Filing Date:
21 February 2013 (21.02.2013)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **EMPIRE TECHNOLOGY DEVELOPMENT, LLC** [US/US]; 2711 Centerville Road, Ste. 400, Wilmington, DE 19808 (US).

(72) Inventor: **SOLIHIN, Yan**; 5300 Panther Creek Drive, Raleigh, NC 27603 (US).

(74) Agent: **RUBIN, Steven, S.**; Moritt Hock & Hamroff LLP, 400 Garden City Plaza, Ste. 202, Garden City, NY 11530 (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, 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, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, 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, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

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

(54) Title: ONE-CACHEABLE MULTI-CORE ARCHITECTURE

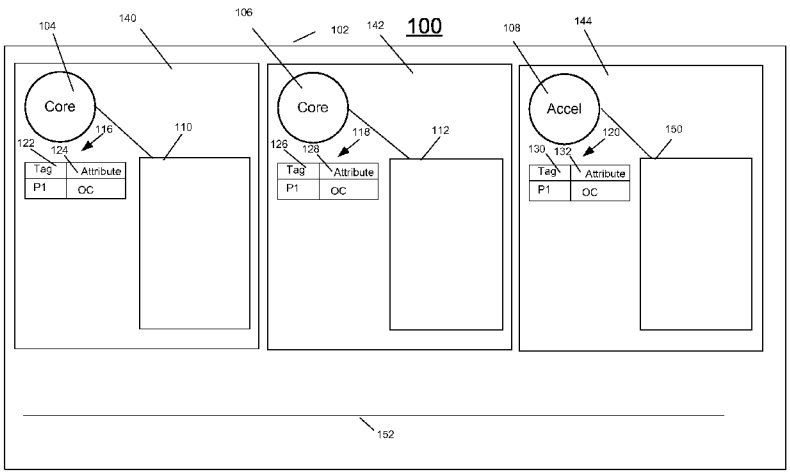


Fig. 1

(57) Abstract: Technologies are generally described for methods, systems, and devices effective to implement one-cacheable multi-core architectures. In one example, a multi-core processor that includes a first and second tile may be configured to implement a one- cacheable architecture. The second tile may be configured to generate a request for a data block. The first tile may be configured to receive the request for the data block, and determine that the requested data block is part of a group of data blocks identified as one-cacheable. The first tile may further determine that the requested data block is stored in a first cache in the first tile. The first tile may send the data block from the first cache in the first tile to the second tile, and invalidate the data blocks of the group of data blocks in the first cache in the first tile.

WO 2014/130037 A1

ONE-CACHEABLE MULTI-CORE ARCHITECTURE

BACKGROUND

[0001] Unless otherwise indicated herein, the materials described in this section are not prior art to the claims in this application and are not admitted to be prior art by inclusion in this section.

[0002] In multi-core architectures, multiple processor cores may be included in a single integrated circuit die or on multiple integrated circuit dies that are arranged in a single chip package. A cache may be used to store data for access by one or more of the processor cores. Resources in the die may be distributed across two or more tiles. Such resources may include, for example, a directory configured to maintain coherence for the caches, memory controllers, processor cores, caches, etc.

SUMMARY

[0003] In some examples, methods for processing a request for a data block in a multi-core processor are generally described. Some methods include receiving the request for the data block at a first tile that includes a first core and a first cache. The request may be received from a second tile that includes a second processor. Some methods include determining, by the first tile, that the data block is part of a group of data blocks. Some methods include determining, by the first tile that one or more data blocks of the group are stored in the first cache in the first tile. Some methods further include sending, by the first tile the data block from the first cache in the first tile to the second tile. Some methods include invalidating, by the first tile, each block in the one or more data blocks of the group of data blocks that are stored in the first cache in the first tile.

[0004] In some examples, tiles configured to process a request for a data block in a multi-core processor are generally described. A first tile may include a first cache, a first core, and a controller configured in communication with the first cache. The controller may be configured to receive the request for the data block. The request may be received from a second tile comprising a second cache and a second

core. The controller may be configured to determine that the data block is part of a group of data blocks. The controller may be configured to determine that the group is stored in one cache in the processor. The controller may be configured to determine that the group is stored in the first cache in the first tile. The controller may be configured to send the data block from the first cache in the first tile to the second tile. The controller may be configured to invalidate the group of data blocks in the first cache in the first tile.

[0005] In some examples, multi-core processors configured to process a request for a data block are generally described. The processors may include a first tile and a second tile configured in communication with the first tile. The first tile may include a first cache and a first controller configured in communication with the first cache. The second tile may be configured to generate the request for the data block, and send the request for the data block to the first tile. The first tile may be configured to receive the request for the data block. The first tile may be configured to determine that the data block is part of a group of data blocks. The first tile may be configured to determine that one or more data blocks of the group are stored in the first cache in the processor. The first tile may be configured to send the data block from the first cache in the first tile to the second tile. The first tile may be configured to invalidate each block in the one or more data blocks of the group of data blocks that are stored in the first cache in the first tile.

[0006] In some examples, multi-core processors configured to process a request for a data block are generally described. The multi-core processors may include a first tile in a die, a second tile in the die and a memory. The first tile may include a first cache and a first controller configured in communication with the first cache. The second tile may be configured in communication with the first tile. The memory may be configured in communication with the first and the second tile. The first tile may be configured to generate the request for the data block. The first tile may be configured to determine that the data block is part of a group of data blocks that is stored exclusively in one cache in the multi-core processor. The first tile may be configured to determine that the data block is not stored in the first cache. The first tile may be configured to determine that another block in the group is stored in the

first cache. The first tile may be configured to, in response to the determination that the data block is not stored in the first cache and the determination that another block in the group is stored in the first cache, send the request to the memory.

[0007] In some examples, multi-core processors configured to process a request for a data block are generally described. The multi-core processors may include a first tile, a second tile and a memory. The first tile may include a first cache and a first controller configured in communication with the first cache. The second tile may be configured in communication with the first tile. The memory may be configured in communication with the first and the second tile. The first tile may be configured to generate a request to initialize a value of the data block. The first tile may be configured to determine that the data block is part of a group of data blocks that is stored exclusively in one cache in the multi-core processor. The first tile may be configured to determine that a block in the group is stored in the first cache. The first tile may be configured to, in response to the determination that a block in the group is stored in the first cache, store the data block in the first cache and initialize the data block to the value.

[0008] The foregoing summary is illustrative only and is not intended to be in any way limiting. In addition to the illustrative aspects, embodiments, and features described above, further aspects, embodiments, and features will become apparent by reference to the drawings and the following detailed description.

BRIEF DESCRIPTION OF THE FIGURES

[0009] The foregoing and other features of this disclosure will become more fully apparent from the following description and appended claims, taken in conjunction with the accompanying drawings. Understanding that these drawings depict only several embodiments in accordance with the disclosure and are, therefore, not to be considered limiting of its scope, the disclosure will be described with additional specificity and detail through use of the accompanying drawings, in which:

Fig. 1 illustrates an example system that can be utilized to implement a one-cacheable multi-core architecture;

Fig. 2 illustrates an example system that can be utilized to implement a one-cacheable multi-core architecture;

Fig. 3 illustrates an example system that can be utilized to implement a one-cacheable multi-core architecture;

Fig. 4 illustrates an example system that can be utilized to implement a one-cacheable multi-core architecture;

Fig. 5 depicts a flow diagram for an example process for implementing a one-cacheable multi-core architecture;

Fig. 6 illustrates a computer program product that can be utilized to implement a one-cacheable multi-core architecture; and

Fig. 7 is a block diagram illustrating an example computing device that is arranged to implement a one-cacheable multi-core architecture;

all arranged according to at least some embodiments described herein.

DETAILED DESCRIPTION

[0010] In the following detailed description, reference is made to the accompanying drawings, which form a part hereof. In the drawings, similar symbols typically identify similar components, unless context dictates otherwise. The illustrative embodiments described in the detailed description, drawings, and claims are not meant to be limiting. Other embodiments may be utilized, and other changes may be made, without departing from the spirit or scope of the subject matter presented herein. It will be readily understood that the aspects of the present disclosure, as generally described herein, and illustrated in the Figures, can be arranged, substituted, combined, separated, and designed in a wide variety of different configurations, all of which are explicitly contemplated herein.

[0011] This disclosure is generally drawn, inter alia, to methods, apparatus, systems, devices, and computer program products related to one-cacheable multi-core architectures.

[0012] Briefly stated, technologies are generally described for methods, systems, and devices effective to implement one-cacheable multi-core architectures. In one example, a multi-core processor that includes a first and second tile may be

configured to implement a one-cacheable architecture. The second tile may be configured to generate a request for a data block. The first tile may be configured to receive the request for the data block, and determine that the requested data block is part of a group of data blocks identified as one-cacheable. The first tile may further determine that the requested data block is stored in a first cache in the first tile. The first tile may send the data block from the first cache in the first tile to the second tile, and invalidate the data blocks of the group of data blocks in the first cache in the first tile.

[0013] As discussed in more detail below, a system may implement a one-cacheable architecture relating to storage of groups of data blocks. An accelerator and a processor core may synchronize and communicate at task boundaries. For example, a processor core may invoke an accelerator for a processing task. The accelerator may access one or more data blocks while executing the task. Coherence for the data blocks may be simplified and may be made more efficient through implementation of a one-cacheable architecture described and explained herein.

[0014] Fig. 1 illustrates an example system that can be utilized to implement one-cacheable multi-core architectures arranged in accordance with at least some embodiments described herein. An example system 100 may include a multi-core processor 102 including two or more tiles 140, 142, and/or 144. Tiles 140, 142, 144 may be configured in communication through a bus 152. Tile 140 may include a processor core 104 and/or a cache 110. Tile 142 may similarly include a processor core 106 and/or a cache 112. Tile 144 may include an accelerator or a special purpose core 108 and a buffer 150. Accelerator 108 may be, for example, a processor core configured for a particular operation such as a compression engine, cryptographic engine, XML (extensible markup language) engine, regular expression engine, packet processing engine, graphics processing unit, etc. Tiles 140, 142 and 144 may each be associated with a respective one of translation lookaside buffers 116, 118 and 120. Each of the translation lookaside buffers may be configured to store information that is organized according to fields, such as a tag field and an attribute field. For example, translation lookaside buffer 116 may store information in a tag field 122 and an attribute field 124; translation lookaside buffer 118 may store information in a tag

field 126 and an attribute field 128; and translation lookaside buffer 120 may store information in a tag field 130 and an attribute field 132.

[0015] The data in tag fields 122, 126, 130 may identify respective groups of data blocks. The data in attribute fields 124, 128, 136 may identify a one-cacheable attribute to indicate that a block from the group of data blocks, identified by the corresponding tag field, may be stored in one tile. In some additional examples, the block may be stored exclusively in one tile. Such an attribute may simplify cache coherence for the corresponding group of data blocks. In still other examples, a coherence region can be defined by a subset of the tiles that form a virtual machine, where the block may be stored exclusively, for the region, in one tile. A one-cacheable attribute may further specify the cache hierarchy level for which the attribute applies, for example whether the one-cacheable attribute applies to the L2 cache, the L3 cache, etc.

[0016] Fig. 2 illustrates an example system that can be utilized to implement one-cacheable multi-core architectures arranged in accordance with at least some embodiments described herein. Those components in Fig. 2 that are labeled identically to components of Fig. 1 will not be described again for the purposes of clarity.

[0017] Processor 102 may be configured to execute instructions. Such instructions may be utilized to control storage and processing of data in processor 102. For example, instructions may configure processor 102 to set a value in one or more of attribute fields 124, 128, 132, each corresponding to one of the tag fields 122, 126, 130. A value in the attribute field may indicate how a corresponding group of data blocks, such as a page, may be cached. For example, a value in attribute field 124, 128, 132 may indicate that a page is "cacheable", "non-cacheable" or "one-cacheable". An attribute of "cacheable" may indicate that a block from the group of data blocks may be cached in a one or more tiles in processor 102. An attribute of "non-cacheable" may indicate that a block from the group of data blocks may not be cached in the processor. An attribute of "one-cacheable" may indicate that one or more blocks from the group of data blocks may be stored in one tile in the processor (e.g., exclusive for a defined coherence region, exclusive to all tiles, etc.). A program

may declare whether a group of data blocks, or a page, is one-cacheable - such as through use of a programming language construct, a programming language extension, a programming language directive, or an application programming interface. A library, such as a library using an accelerator, may have an entry that, when processed, may cause a determination that a group of data blocks is one-cacheable. In an example, the library may include an entry that identifies blocks in a page as one-cacheable.

[0018] Tile 144 with accelerator 108 may include a buffer 150. Accelerator 108 may be configured in communication with one or more cache controllers 158, 160 (described below) through bus or communication link 152. A main memory 154 may be external to and configured in communication with die or dies 102. Data stored in main memory 154 may include a page table 156 that includes a tag field 180 and attribute field 182. A memory controller 155 in die 102 may facilitate movement of data to and from main memory 154. Values stored in page table 156 may identify a list of pages stored in caches in die 102. Page table 156 may also store indications as to how each of the pages may be cached. For example, page table 156 may include fields indicating whether the respective page is cacheable, non-cacheable, or one-cacheable.

[0019] Caches 110, 112 may be coupled to a respective one of cache controllers 158, 160; where the cache controllers 158, 160 are operable to control movement of data into and out of caches 110, 112. Accelerator 108 may include a buffer controller 162; where the buffer controller 162 is operable to control movement of data into and out of buffer 150. Cache 110 may be configured to store information in a cache tag field 166 and/or a data field 168. Cache 112 may include a cache tag field 170 and/or a data field 172. Buffer 150 may include information in a buffer tag field 171 and/or a data field 173. In some examples, buffer 150 may not include a buffer tag field 171. In another example, buffer 150 may be a hardware-managed cache, a hardware-managed prefetching buffer, a software-managed buffer, or a software-managed cache. Data fields 168, 172, 173 may be operable to store data for data blocks identified in a corresponding one of cache or buffer tag fields 166, 170, 171.

[0020] Fig. 3 illustrates an example system that can be utilized to implement one-cacheable multi-core architectures in accordance with at least some embodiments described herein. Those components in Fig. 3 that are labeled identically to components of Figs. 1 or 2 will not be described again for the purposes of clarity.

[0021] Processor 102 may be configured to keep track of whether a block belongs to a group of data blocks (such as a page) that is indicated as one-cacheable. In one example, cache 110 may be utilized to maintain a table 157. Values stored in table 157 may indicate pages that are one-cacheable for which one or more blocks are stored in a respective cache. Table 157 may be indexed using a physical address of a block or a page, and may contain a bitmap that indicates which blocks in a page are currently cached. As blocks from one or more pages can be stored in cache 110, cache controller 158 may update values in table 157 accordingly. As blocks are evicted from cache 110, cache controller 158 may update values in table 157 to reflect the evicted blocks. This example, using table 157, may have relatively greater complexity in implementation, than without table 157. The greater complexity may be because the table should track all one-cacheable pages correctly, the table should be addressable by physical address, and the table should be kept coherent against translation lookaside buffers.

[0022] In another example, where table 157 is not included, as cores 104, 106 or accelerator 108 initiate requests (e.g., read or write requests) for data stored in a cache, the requests may include an indication as to whether a page is cached as one-cacheable. A core (such as through a cache controller) or an accelerator (such as through a buffer controller) may initiate a request for data stored in the cache of another core. Such a request may be generated when a core or accelerator imitates a request to read from, or write data to, a particular data block. In response to a request to read from, or write to, at least one data block of a one-cacheable group of data blocks, a cache controller may invalidate other blocks from the group of data blocks that are stored in the corresponding cache.

[0023] In the example illustrated in Fig. 3, core 106 may initiate a request to write to data block "E" on group or page 5 ("Write P5.E"). Core 106 may first access translation lookaside buffer 118 to determine whether an indication corresponding to

page 5 is stored in tag field 126 of translation lookaside buffer 118. In examples where an indication corresponding to page 5 is stored in translation lookaside buffer 118, core 106 may analyze attribute field 128 to determine how page 5 is cached. In an example, processor 106 may determine that page 5 is cached as one-cacheable.

[0024] In examples where an indication corresponding to page 5 is not stored in translation lookaside buffer 118, core 106 may initiate a request to memory 154 requesting the corresponding portion of page table 156 that includes page 5. The corresponding portion of page table 156 may then be sent to core 106 in response to the request to memory 154. Cache controller 160 or a memory management unit may then store the corresponding portion of page table 156 in translation lookaside buffer 118. In the example, an indication of page P5 may be stored in translation lookaside buffer 118 with a one-cacheable attribute ("OC") in attribute field 128.

[0025] After analyzing translation lookaside buffer 118 for page 5, and perhaps initiating a request to memory 154, cache controller 160 may then initiate a request 174. In the example, request 174 may be a request to write to block E of page 5. Request 174 may be initiated by cache controller 160 over a bus or other interconnect. Request 174 may also include an indication that page 5 is one-cacheable ("OC"). The indication of one-cacheable ("OC") may provide information to tile 140 that page 5 is one-cacheable so that cache controller 158 may enforce the one-cacheable attribute semantic - discussed in more detail below.

[0026] Similarly, in an example, accelerator 108 may initiate a request to read from data block "A" on page 1 ("Read P1.A"). Accelerator 108 may first analyze values in the translation lookaside buffer 120 to determine whether an indication corresponding to page 1 is stored in tag field 130 of translation lookaside buffer 120. In examples where an indication corresponding to page 1 is stored in translation lookaside buffer 120, accelerator 108 may analyze values in the attribute field 132 to determine how page 1 is cached. In examples where an indication corresponding to page 1 is not stored in translation lookaside buffer 120, accelerator 108 may initiate a request to memory 154 for the corresponding portion of page table 156 that includes page 1. The corresponding portion of page table 156 may then be sent to accelerator 108. Buffer controller 162 may then store the corresponding portion of page table 156

in translation lookaside buffer 120. In the example, an indication of page P1 may be stored in translation lookaside buffer 118 with a one-cacheable attribute ("OC").

[0027] After analyzing values in the translation lookaside buffer 120 for page 1, buffer controller 162 may then initiate a request 176. In the example, request 176 can be a request to read for block A of page 1. Request 176 may be sent from buffer controller 162 of tile 144 such as over a bus or other interconnect. Request 176 may include a request to read from block A of page 1 along with an indication that page 1 is one-cacheable ("OC"). The indication of one-cacheable ("OC") may provide information to tile 140 that page 1 is one-cacheable so that cache controller 158 may enforce the one-cacheable attribute semantic.

[0028] Fig. 4 illustrates an example system that can be utilized to implement one-cacheable multi-core architectures arranged in accordance with at least some embodiments described herein. Those components in Fig. 4 that are labeled identically to components of Figs. 1, 2 or 3 will not be described again for the purposes of clarity.

[0029] In the example described above with reference to Fig. 3, cache controller 158 may receive a read request 176 for a data block. In the example, read request 176 can be a request to read block A of page 1. In responding to read request 176, cache controller 158 may identify one or more modified or dirty blocks of data in cache 110 corresponding to page 1. Such modified or dirty blocks of data may have been modified after the blocks were retrieved from memory 154 or shared cache 190. In one example, shared cache 190 may be an L3 or L4 cache. Cache controller 158 may, before responding to read request 176, initiate a write back of the modified or dirty blocks to shared cache 190. For example, if cache 110 corresponds to an L2 cache, cache controller 158 may initiate a write back of the modified blocks to an L3 cache. Shared cache 190 may be shared by multiple tiles in the multi-core processor 102.

[0030] In further response to read request 176, cache controller 158 may invalidate one or more entries in cache 110 for blocks corresponding to page 1 - as illustrated by the "X". Such invalidation may ensure that blocks identified as belonging to a group of blocks or page that are identified as one-cacheable are stored

(e.g., exclusively for a defined coherence region, exclusive to all tiles, etc.) in one cache. Cache controller 158 may then send block A of page 1 through bus or communication link 152. Write requests may be handled in a manner similar to read requests.

[0031] In another example, prior to the invalidation, cache controller 158 may send one or more additional blocks stored in cache 110 to tile 144. The additional blocks may correspond to the page identified in read request 176. In the example, read request 176 can request to read block A of page 1. Cache controller 158 may send a reply message 178 including additional blocks of page 1 that are stored in cache 110. In the example, block A and C of page 1 are stored in cache 110. Therefore, blocks A and C of page 1 may be sent in reply message 178 to tile 144. Buffer controller 162 of tile 144 may receive blocks A and C in reply message 178 and store blocks A and C in buffer 150. Buffer controller 162 may fetch additional blocks in page 1 not supplied in reply message 178 from memory 154.

[0032] In some examples, cache controller 158 may be configured to invalidate blocks in response to read or write requests but not send reply message 178. Invalidation alone, without reply message 178 may be beneficial in circumstances where cache 110 is not the lowest level cache - such as when a shared cache 190 is used. In this example, where cache 110 is not the lowest level cache, data blocks invalidated from cache 110 may still be fetched from shared cache 190 at a lower level latency than if cache controller 158 were to fetch the data from memory 154. In examples where a requesting core is physically far away, such as may be the case with an accelerator, utilizing reply message 178 may be beneficial because data blocks of a page can be transferred to a requesting core or accelerator proactively. Such transfer of data blocks may result in fast and energy efficient operation of the multi-core processor 102.

[0033] In an example, accelerator 108 may make a request to access a data block P1.B. The access request may result in a miss in buffer 150. Translation lookaside buffer 120 may indicate that page P1 is a one-cacheable page. Buffer controller 162 may determine that at least one block in page P1 (P1.A or P1.C) is currently stored in buffer 150. Due to the property of one-cacheable page, buffer

controller 162 can infer that none of page P1's blocks are stored in other caches. Thus, buffer controller 162 can issue a non-coherent memory fetch directly to memory controller 155. The memory fetch request may not be broadcasted on bus 152 in a snoopy coherence protocol, and may not inquire a directory in a directory-based coherence protocol. Instead, the memory fetch request may be directly sent to the memory controller, bypassing coherence protocols.

[0034] In an example, accelerator 108 may make a request to access and initialize a data block P1.D with a zero value. The access request may result in a miss in buffer 150. Translation lookaside buffer 120 may indicate that page P1 is a one-cacheable page. Buffer controller 162 may determine that at least one block in page P1 (P1.A or P1.C) is currently stored in buffer 150. Due to the property of one-cacheable page, buffer controller 162 can infer that none of page P1's blocks are stored in other caches. Thus, buffer controller 162 can install a block initialized to zero value directly in the buffer 150. The block installation may not be broadcasted on bus 152 in a snoopy coherence protocol, and may not inquire a directory in a directory-based coherence protocol. The block installation bypasses the coherence protocol.

[0035] Among other possible benefits, a system configured in accordance with the present disclosure may be able to simplify cache coherence. In examples where a group of data blocks or a page is indicated as being one-cacheable, the system may know that once a first block is found, no other tiles will cache (e.g., no other tiles within the coherence region) blocks in the page. Thus, once a first block of a page is found, processors need not look in other caches. Consequently, blocks may be transferred in bulk. Once a first block of a one-cacheable page is transferred, other blocks in the page should also be transferred without requiring a specific request to transfer the other blocks.

[0036] Coherence mechanisms can take advantage of a communication pattern between accelerators and cores. Accelerators may access relatively large blocks of data sequentially. For example, a processor core may process a large block of data, followed by the accelerator processing the large block of data. Using one of the disclosed systems, the accelerator may obtain data from an input stream from one

cache used by a thread where the thread dispatched a task. The accelerator need not probe other caches. By obtaining data from one cache, the accelerator or processor core may reduce the number of cache miss requests and number of coherence messages, thereby reducing data transfer latency, and improving performance and energy efficiency.

[0037] A one-cacheable attribute may provide for a level of coherence at a granularity level that may be particularly useful for use in multi-core processors that include an accelerator or a special purpose core. Validation messages may be sent at task boundaries resulting in less communication traffic. Synchronization and communication between an accelerator and a core may be realized. Bulk transfer of data stored in a cache may be realized because a request for one block may result in other related blocks being transferred. Bus broadcasting may be avoided. If a private cache has one block belonging to a group of blocks or a page identified as one-cacheable, no blocks from such a group would be available in other private caches (e.g. other private caches within a defined coherence region). In response to a miss for a block of the one-cacheable page, non-coherent requests for the block may be made to a shared cache or the main memory. By limiting broadcast to other caches and directing requests to a shared cache or main memory, lower latency and lower power consumption may be realized.

[0038] In libraries where blocks are zeroed out, when there is a cache miss in a one-cacheable page where at least one block is found in the cache, the cache controller may avoid fetching further blocks. Any cache misses to other blocks in the group can be serviced by installing blocks initialized to a zero value. By directly installing zero valued blocks in the cache, lower latency, lower bandwidth consumption from coherence traffic, and lower power consumption may be realized.

[0039] Fig. 5 depicts a flow diagram for an example process for implementing one-cacheable multi-core architectures arranged in accordance with at least some embodiments described herein. In some examples, the process in Fig. 5 could be implemented using system 100 discussed above to process a request for a data block in a multi-core processor.

[0040] An example process may include one or more operations, actions, or functions as illustrated by one or more of blocks S2, S4, S6, S8 and/or S10. Although illustrated as discrete blocks, various blocks may be divided into additional blocks, combined into fewer blocks, or eliminated, depending on the desired implementation.

[0041] Processing may begin at block S2, "Receive the request for the data block at a first tile that includes a first core and a first cache, wherein the request is received from a second tile that includes a second processor." At block S2, a first tile may receive a request for a data block. The first tile may include a first core and a first cache. The request may be received from a second tile that includes a second core. The request may include an indication that blocks in the group are stored in one cache in the processor. The request may be a read, write, or pre-fetch request.

[0042] Processing may continue from block S2 to block S4, "Determine, by the first tile, that the data block is part of a group of data blocks." At block S4, the first tile may determine that the data block is part of a group of data blocks.

[0043] Processing may also continue from block S4 to block S6, determine, by the first tile, that the one or more data blocks of the group are stored in the first cache in the first cache in the first tile. At block S6, the first tile may determine that the group of data blocks is stored in one cache in the first tile and that the group of data blocks is stored in the first cache in the first tile.

[0044] Processing may continue from block S6 to block S8, "Send, by the first tile, the data block from the first cache in the first tile to the second tile." At block S8, the first tile may send the data block from the first cache in the first tile to the second tile. The first tile may further send other data blocks of the group stored in the first cache from the first cache to the second tile.

[0045] Processing may continue from block S8 to block S10, "Invalidate, by the first tile, each block in the one or more data blocks of the group of data blocks that are stored in the first cache in the first tile." At block S10, the first tile may invalidate each block (and write the block back if the block is dirty) in the one or more data blocks in the first cache in the first tile. Prior to the invalidation, the first tile may store the data blocks in a shared cache of the multi-core processor. The first

tile may store a table that identifies the group and indicates that the group is cacheable in one L2 cache(e.g., exclusively within a defined coherence region).

[0046] Fig. 6 illustrates an example computer program product 300 that can be utilized to implement a one-cacheable multi-core architecture arranged in accordance with at least some embodiments described herein. Program product 300 may include a signal bearing medium 302. Signal bearing medium 302 may include one or more instructions 304 that, when executed by, for example, a processor, may provide the functionality described above with respect to Figs. 1-5. Thus, for example, referring to system 100, one or more of core 104, core 106 and/or accelerator 108 in tiles 140, 142, 144 may undertake one or more of the blocks shown in Fig. 6 in response to instructions 304 conveyed to the system 100 by medium 302.

[0047] In some implementations, signal bearing medium 302 may encompass a computer-readable medium 306, such as, but not limited to, a hard disk drive, a Compact Disc (CD), a Digital Video Disk (DVD), a digital tape, memory, etc. In some implementations, signal bearing medium 302 may encompass a recordable medium 308, such as, but not limited to, memory, read/write (R/W) CDs, R/W DVDs, etc. In some implementations, signal bearing medium 302 may encompass a communications medium 310, such as, but not limited to, a digital and/or an analog communication medium (e.g., a fiber optic cable, a waveguide, a wired communications link, a wireless communication link, etc.). Thus, for example, program product 300 may be conveyed to one or more modules of the system 100 by an RF signal bearing medium 302, where the signal bearing medium 302 is conveyed by a wireless communications medium 310 (e.g., a wireless communications medium conforming with the IEEE 802.11 standard).

[0048] Fig. 7 is a block diagram illustrating an example computing device 400 that is arranged to implement a one-cacheable multi-core architecture arranged in accordance with at least some embodiments described herein. In a very basic configuration 402, computing device 400 typically includes one or more processors 404 and a system memory 406. A memory bus 408 may be used for communicating between processor 404 and system memory 406.

[0049] Depending on the desired configuration, processor 404 may be of any type including but not limited to a microprocessor (μP), a microcontroller (μC), a digital signal processor (DSP), or any combination thereof. Processor 404 may include one more levels of caching, such as a level one cache 410 and a level two cache 412, a processor core 414, and registers 416. An example processor core 414 may include an arithmetic logic unit (ALU), a floating point unit (FPU), a digital signal processing core (DSP Core), or any combination thereof. An example memory controller 418 may also be used with processor 404, or in some implementations memory controller 418 may be an internal part of processor 404.

[0050] Depending on the desired configuration, system memory 406 may be of any type including but not limited to volatile memory (such as RAM), non-volatile memory (such as ROM, flash memory, etc.) or any combination thereof. System memory 406 may include an operating system 420, one or more applications 422, and program data 424. Application 422 may include a one-cacheable architecture algorithm 426 that is arranged to perform the functions as described herein including those described with respect to system 100 of Figs. 1-6. Program data 424 may include one-cacheable architecture data 428 that may be useful to implement a one-cacheable architecture as is described herein. In some embodiments, application 422 may be arranged to operate with program data 424 on operating system 420 such that a one-cacheable architecture may be provided. This described basic configuration 402 is illustrated in Fig. 7 by those components within the inner dashed line.

[0051] Computing device 400 may have additional features or functionality, and additional interfaces to facilitate communications between basic configuration 402 and any required devices and interfaces. For example, a bus/interface controller 430 may be used to facilitate communications between basic configuration 402 and one or more data storage devices 432 via a storage interface bus 434. Data storage devices 432 may be removable storage devices 436, non-removable storage devices 438, or a combination thereof. Examples of removable storage and non-removable storage devices include magnetic disk devices such as flexible disk drives and hard-disk drives (HDD), optical disk drives such as compact disk (CD) drives or digital versatile disk (DVD) drives, solid state drives (SSD), and tape drives to name a few.

Example computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data.

[0052] System memory 406, removable storage devices 436 and non-removable storage devices 438 are examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which may be used to store the desired information and which may be accessed by computing device 400. Any such computer storage media may be part of computing device 400.

[0053] Computing device 400 may also include an interface bus 440 for facilitating communication from various interface devices (e.g., output devices 442, peripheral interfaces 444, and communication devices 446) to basic configuration 402 via bus/interface controller 430. Example output devices 442 include a graphics processing unit 448 and an audio processing unit 450, which may be configured to communicate to various external devices such as a display or speakers via one or more A/V ports 452. Example peripheral interfaces 444 include a serial interface controller 454 or a parallel interface controller 456, which may be configured to communicate with external devices such as input devices (e.g., keyboard, mouse, pen, voice input device, touch input device, etc.) or other peripheral devices (e.g., printer, scanner, etc.) via one or more I/O ports 458. An example communication device 446 includes a network controller 460, which may be arranged to facilitate communications with one or more other computing devices 462 over a network communication link via one or more communication ports 464.

[0054] The network communication link may be one example of a communication media. Communication media may typically be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and may include any information delivery media. A "modulated data signal" may be a signal

that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media may include wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, radio frequency (RF), microwave, infrared (IR) and other wireless media. The term computer readable media as used herein may include both storage media and communication media.

[0055] Computing device 400 may be implemented as a portion of a small-form factor portable (or mobile) electronic device such as a cell phone, a personal data assistant (PDA), a personal media player device, a wireless web-watch device, a personal headset device, an application specific device, or a hybrid device that include any of the above functions. Computing device 400 may also be implemented as a personal computer including both laptop computer and non-laptop computer configurations.

[0056] The present disclosure is not to be limited in terms of the particular embodiments described in this application, which are intended as illustrations of various aspects. Many modifications and variations can be made without departing from its spirit and scope, as will be apparent to those skilled in the art. Functionally equivalent methods and apparatuses within the scope of the disclosure, in addition to those enumerated herein, will be apparent to those skilled in the art from the foregoing descriptions. Such modifications and variations are intended to fall within the scope of the appended claims. The present disclosure is to be limited only by the terms of the appended claims, along with the full scope of equivalents to which such claims are entitled. It is to be understood that this disclosure is not limited to particular methods, reagents, compounds compositions or biological systems, which can, of course, vary. It is also to be understood that the terminology used herein is for the purpose of describing particular embodiments only, and is not intended to be limiting.

[0057] With respect to the use of substantially any plural and/or singular terms herein, those having skill in the art can translate from the plural to the singular and/or from the singular to the plural as is appropriate to the context and/or

application. The various singular/plural permutations may be expressly set forth herein for sake of clarity.

[0058] It will be understood by those within the art that, in general, terms used herein, and especially in the appended claims (*e.g.*, bodies of the appended claims) are generally intended as "open" terms (*e.g.*, the term "including" should be interpreted as "including but not limited to," the term "having" should be interpreted as "having at least," the term "includes" should be interpreted as "includes but is not limited to," etc.). It will be further understood by those within the art that if a specific number of an introduced claim recitation is intended, such an intent will be explicitly recited in the claim, and in the absence of such recitation no such intent is present. For example, as an aid to understanding, the following appended claims may contain usage of the introductory phrases "at least one" and "one or more" to introduce claim recitations. However, the use of such phrases should not be construed to imply that the introduction of a claim recitation by the indefinite articles "a" or "an" limits any particular claim containing such introduced claim recitation to embodiments containing only one such recitation, even when the same claim includes the introductory phrases "one or more" or "at least one" and indefinite articles such as "a" or "an" (*e.g.*, "a" and/or "an" should be interpreted to mean "at least one" or "one or more"); the same holds true for the use of definite articles used to introduce claim recitations. In addition, even if a specific number of an introduced claim recitation is explicitly recited, those skilled in the art will recognize that such recitation should be interpreted to mean at least the recited number (*e.g.*, the bare recitation of "two recitations," without other modifiers, means at least two recitations, or two or more recitations). Furthermore, in those instances where a convention analogous to "at least one of A, B, and C, etc." is used, in general such a construction is intended in the sense one having skill in the art would understand the convention (*e.g.*, " a system having at least one of A, B, and C" would include but not be limited to systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together, etc.). In those instances where a convention analogous to "at least one of A, B, or C, etc." is used, in general such a construction is intended in the sense one having skill in the art would understand the convention

(*e.g.*, " a system having at least one of A, B, or C" would include but not be limited to systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together, etc.). It will be further understood by those within the art that virtually any disjunctive word and/or phrase presenting two or more alternative terms, whether in the description, claims, or drawings, should be understood to contemplate the possibilities of including one of the terms, either of the terms, or both terms. For example, the phrase "A or B" will be understood to include the possibilities of "A" or "B" or "A and B."

[0059] In addition, where features or aspects of the disclosure are described in terms of Markush groups, those skilled in the art will recognize that the disclosure is also thereby described in terms of any individual member or subgroup of members of the Markush group.

[0060] As will be understood by one skilled in the art, for any and all purposes, such as in terms of providing a written description, all ranges disclosed herein also encompass any and all possible subranges and combinations of subranges thereof. Any listed range can be easily recognized as sufficiently describing and enabling the same range being broken down into at least equal halves, thirds, quarters, fifths, tenths, etc. As a non-limiting example, each range discussed herein can be readily broken down into a lower third, middle third and upper third, etc. As will also be understood by one skilled in the art all language such as "up to," "at least," "greater than," "less than," and the like include the number recited and refer to ranges which can be subsequently broken down into subranges as discussed above. Finally, as will be understood by one skilled in the art, a range includes each individual member. Thus, for example, a group having 1-3 cells refers to groups having 1, 2, or 3 cells. Similarly, a group having 1-5 cells refers to groups having 1, 2, 3, 4, or 5 cells, and so forth.

[0061] While various aspects and embodiments have been disclosed herein, other aspects and embodiments will be apparent to those skilled in the art. The various aspects and embodiments disclosed herein are for purposes of illustration and are not intended to be limiting, with the true scope and spirit being indicated by the following claims.

CLAIMS

What is claimed is:

1. A method for processing a request for a data block in a multi-core processor, the method comprising:

receiving the request for the data block at a first tile that includes a first core and a first cache, wherein the request is received from a second tile that includes a second processor;

determining, by the first tile, that the data block is part of a group of data blocks;

determining, by the first tile, that one or more data blocks of the group are stored in the first cache in the first tile;

sending, by the first tile, the data block from the first cache in the first tile to the second tile; and

invalidating, by the first tile, each block in the one or more data blocks of the group of data blocks that are stored in the first cache in the first tile.

2. The method of claim 1, further comprising determining, by the first tile, that the group of data blocks corresponds to a memory page.

3. The method of claim 1, further comprising sending other data blocks of the group stored in the first cache from the first tile to the second tile.

4. The method of claim 1, further comprising sending other data blocks of the group stored in the first cache from the first tile to the second tile without receiving an additional request to send the other data blocks of the group.

5. The method of claim 1, further comprising prior to invalidating, writing back dirty data blocks to a shared cache of the multi-core processor.

6. The method of claim 1, wherein receiving the request for the data block includes receiving the request with an indication that the block is a part of a group of data blocks that are cacheable exclusively in one cache in the processor.

7. The method of claim 1, wherein receiving the request for the data block includes receiving a request to read the data block.

8. The method of claim 1, wherein receiving the request for the data block includes receiving a request to write the data block.

9. The method of claim 1, wherein receiving the request for the data block includes receiving a request to read or write the data block sent over an interconnect of the multi-core processor.

10. The method of claim 1, further comprising, storing, in the first tile, a table, wherein the table includes an indication of the group and indicates that a block in the group is cacheable in exclusively one cache in the processor.

11. A first tile configured to process a request for a data block in a multi-core processor, the first tile comprising:

a first cache;

a first core;

a controller configured in communication with the first cache, wherein the controller is configured to:

receive the request for the data block, wherein the request is received from a second tile comprising a second cache and a second core,

determine that the data block is part of a group of data blocks,

determine that the group is stored in one cache in the processor,

determine that the group is stored in the first cache in the first tile,

send the data block from the first cache in the first tile to the second tile, and

invalidate the group of data blocks in the first cache in the first tile.

12. The first tile of claim 11, wherein the first core and the second core are both general purpose processor cores.

13. The first tile of claim 11, wherein the first core is a general purpose processor core and the second core is a special purpose processor core.

14. The first tile of claim 11, wherein the group of data blocks corresponds to a memory page.

15. The first tile of claim 11, wherein the controller is further configured to send other data blocks of the group stored in the first cache from the first tile to the second tile.

16. The first tile of claim 11, wherein the controller is further configured to send other data blocks of the group stored in the first cache from the first tile to the second tile without receipt of an additional request to send the other data blocks of the group.

17. The first tile of claim 11, wherein the controller is further configured to, prior to the invalidation, write back dirty data blocks to a shared cache of the multi-core processor.

18. The first tile of claim 11, wherein the controller is further configured to receive the request with an indication that the block is part of a group of data blocks that are cacheable exclusively in one cache in the processor.

19. The tile of claim 11, wherein the controller is further configured to receive a request to read the data block.

20. The tile of claim 11, wherein the controller is further configured to receive a request to write the data block.

21. A multi-core processor configured to process a request for a data block, the processor comprising:

a first tile, wherein the first tile includes a first cache and a first controller configured in communication with the first cache; and

a second tile configured in communication with the first tile,

wherein the second tile is configured to

generate the request for the data block, and

send the request for the data block to the first tile,

wherein the first tile is configured to:

receive the request for the data block,

determine that the data block is part of a group of data blocks,

determine that one or more data blocks of the group are stored in the first cache in the processor,

send the data block from the first cache in the first tile to the second tile, and

invalidate each block in the one or more data blocks of the group of data blocks that are stored in the first cache in the first tile.

22. The multi-core processor of claim 21, wherein the group of data blocks corresponds to a memory page.

23. The multi-core processor of claim 21, wherein the first tile is further configured to send other data blocks of the group stored in the first cache from the first tile to the second tile.

24. The multi-core processor of claim 21, wherein the first tile is further configured to send other data blocks of the group stored in the first cache from the first tile to the second tile without receipt of an additional request to send the other data blocks of the group.

25. The multi-core processor of claim 21, wherein the first tile is further configured to, prior to the invalidation, store the data block in a shared cache of the multi-core processor.

26. The multi-core processor of claim 21, wherein the first tile is further configured to receive the request with an indication that the block is part of a group of data blocks that are cacheable exclusively in one cache in the processor.

27. The multi-core processor of claim 21, wherein the controller is further configured to receive a request to read the data block.

28. The multi-core processor of claim 21, wherein the controller is further configured to receive a request to write the data block.

29. The multi-core processor of claim 21, wherein the second tile includes an accelerator.

30. The multi-core processor of claim 21, wherein the second tile is further configured to, prior to generation of the request:

send a request to a main memory for a section of a page table that corresponds to the data block;

receive the section of the page table from the main memory; and

based on the received section of the page table, determine that the data block is part of a group of data blocks cacheable exclusively in one tile in the multi-core processor.

31. The multi-core processor of claim 21, wherein the second tile is further configured to, prior to generation of the request, analyze a translation lookaside buffer in the second tile and determine that the data block is part of a group of data blocks cached exclusively in one cache in the processor.

32. The multi-core processor of claim 21, wherein the second tile is further configured to, prior to generation of the request:

based on a translation lookaside buffer in the second tile, determine that an indication corresponding to the data block is not stored in the translation lookaside buffer;

send a request to a main memory for a section of a page table that corresponds to the data block;

receive the section of the page table from the main memory; and

based on the received section of the page table, determine that the data block is part of a group of data blocks cacheable exclusively in one tile in the multi-core processor.

33. The multi-core processor of claim 21, wherein:

the first tile is further configured to send one or more reply messages to the second tile, wherein the one or more reply messages include other data blocks of the group stored in the first cache; and

the second tile is further configured to:

determine that a particular block is part of the group but not in the reply message, and

send a request to a shared cache for the particular block.

34. A multi-core processor configured to process a request for a data block, the multi-core processor comprising:

a first tile in a die, wherein the first tile includes a first cache and a first controller configured in communication with the first cache;

a second tile in the die and configured in communication with the first tile; and

a memory configured in communication with the first and the second tile;

wherein the first tile is configured to:

generate the request for the data block;

determine that the data block is part of a group of data blocks that is stored exclusively in one cache in the multi-core processor;

determine that the data block is not stored in the first cache;

determine that another block in the group is stored in the first cache;

in response to the determination that the data block is not stored in the first cache and the determination that another block in the group is stored in the first cache, send the request to the memory.

35. A multi-core processor configured to process a request for a data block, the multi-core processor comprising:

a first tile, wherein the first tile includes a first cache and a first controller configured in communication with the first cache;

a second tile configured in communication with the first tile; and

a memory configured in communication with the first and the second tile;

wherein the first tile is configured to:

generate a request to initialize a value of the data block;

determine that the data block is part of a group of data blocks that is stored exclusively in one cache in the multi-core processor;

determine that a block in the group is stored in the first cache;

in response to the determination that a block in the group is stored in the first cache, store the data block in the first cache and initialize the data block to the value.

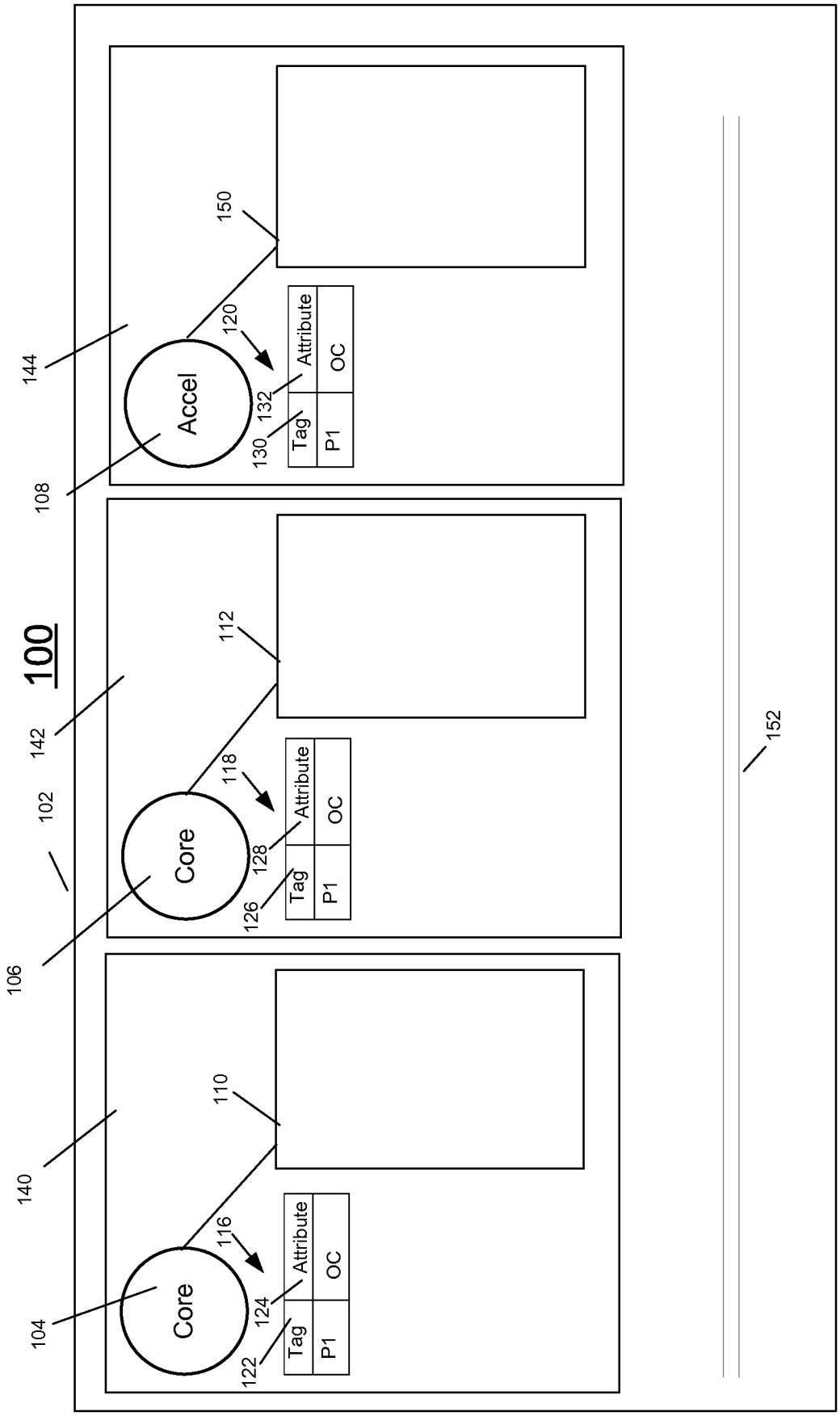

100
102
104
106
108
110
112
116
118
120
122
124
126
128
130
132
140
142
144
150
152
Core
Core
Accel
Tag
P1
Attribute
OC

Fig. 1

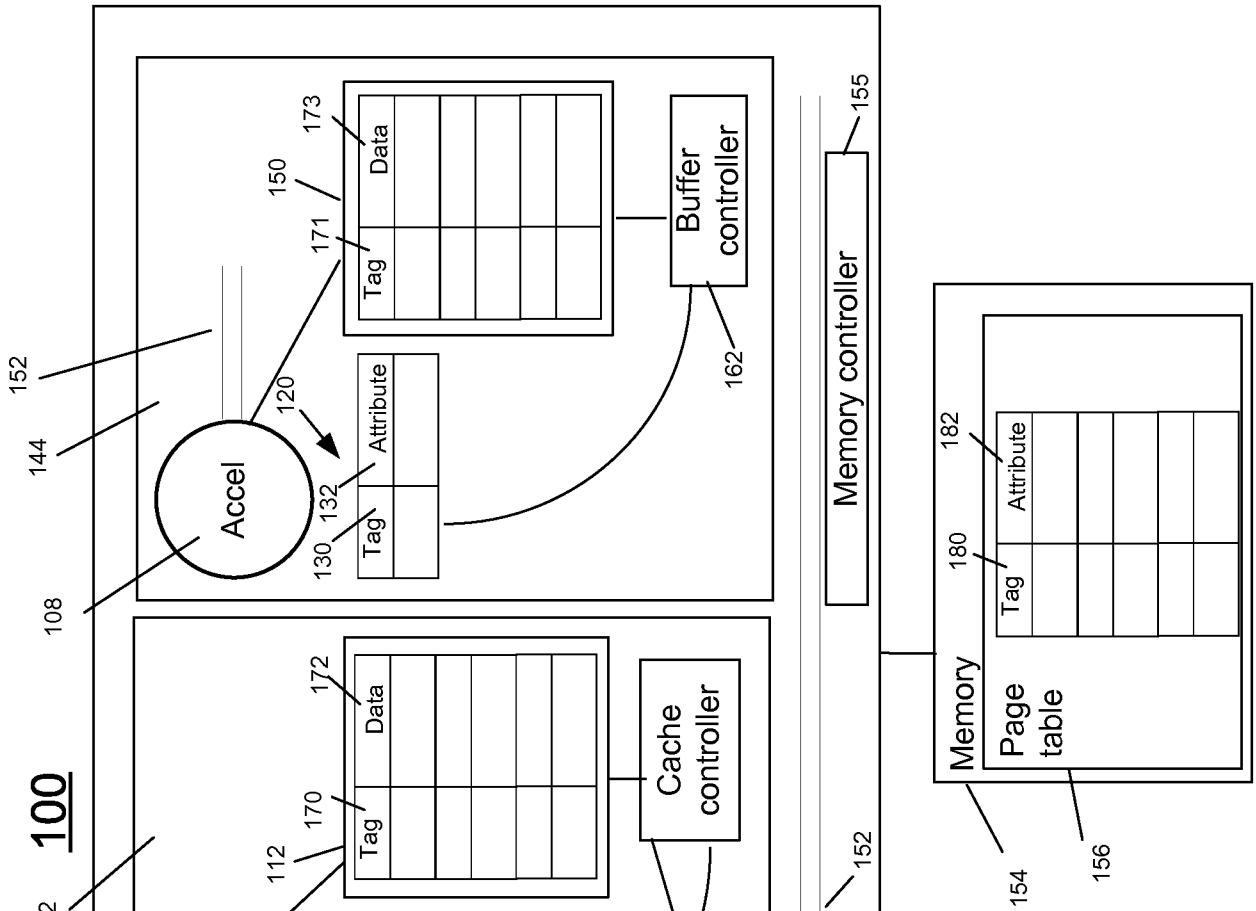

100
102
104
106
108
110
112
120
122
124
126
128
130
132
140
142
144
150
152
154
155
156
158
160
162
166
168
170
171
172
173
180
182
Core
Core
Accel
Cache controller
Cache controller
Buffer controller
Memory controller
Tag
Attribute
Data
Memory
Page table

Fig. 2

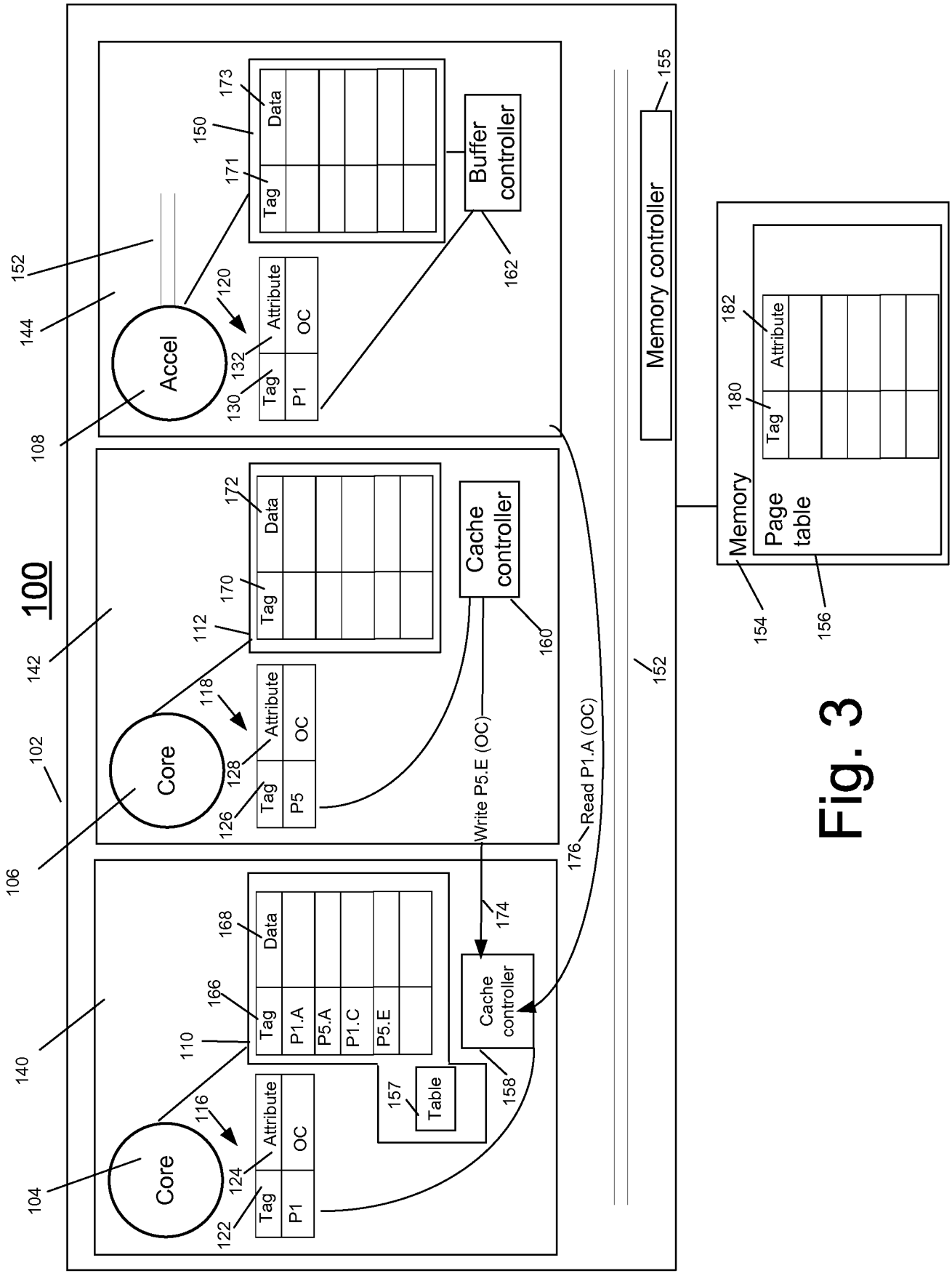

100
Core
Accel
Cache controller
Buffer controller
Memory controller
Memory
Page table
Tag
Attribute
OC
Data
P1
P5
P1.A
P5.A
P1.C
P5.E
Table
Write P5.E (OC)
Read P1.A (OC)
102
104
106
108
110
112
116
118
120
122
124
126
128
130
132
140
142
144
150
152
154
155
156
157
158
160
162
166
168
170
171
172
173
174
176
180
182

Fig. 3

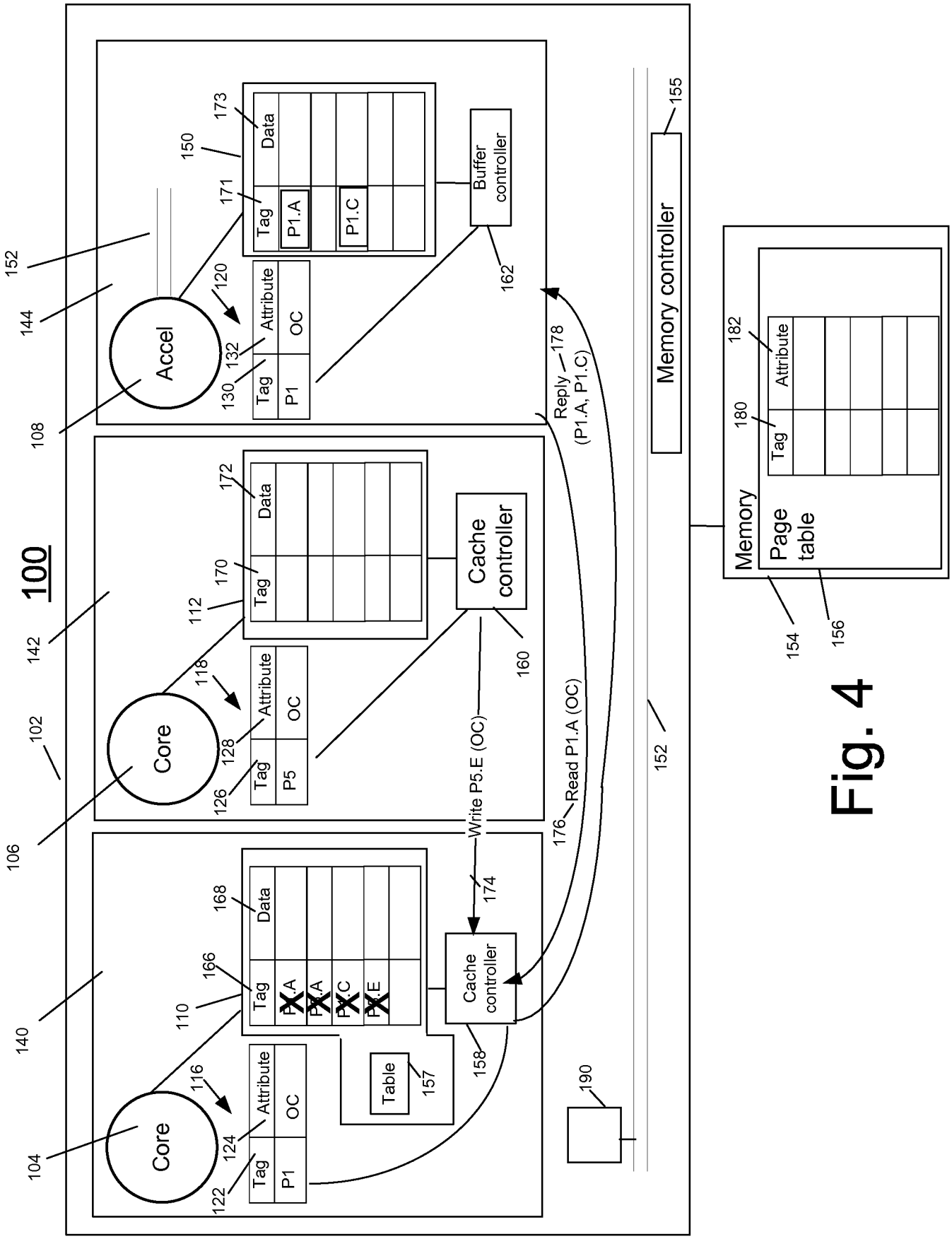

100
Core
Accel
Core
Attribute
OC
Tag
P1
P5
Data
Tag
P1.A
P1.C
P1.E
Table
Cache controller
Buffer controller
Memory controller
Memory
Page table
Write P5.E (OC)
Read P1.A (OC)
Reply (P1.A, P1.C)
102
104
106
108
110
112
116
118
120
122
124
126
128
130
132
140
142
144
150
152
154
155
156
157
158
160
162
166
168
170
171
172
173
174
176
178
180
182
190

Fig. 4

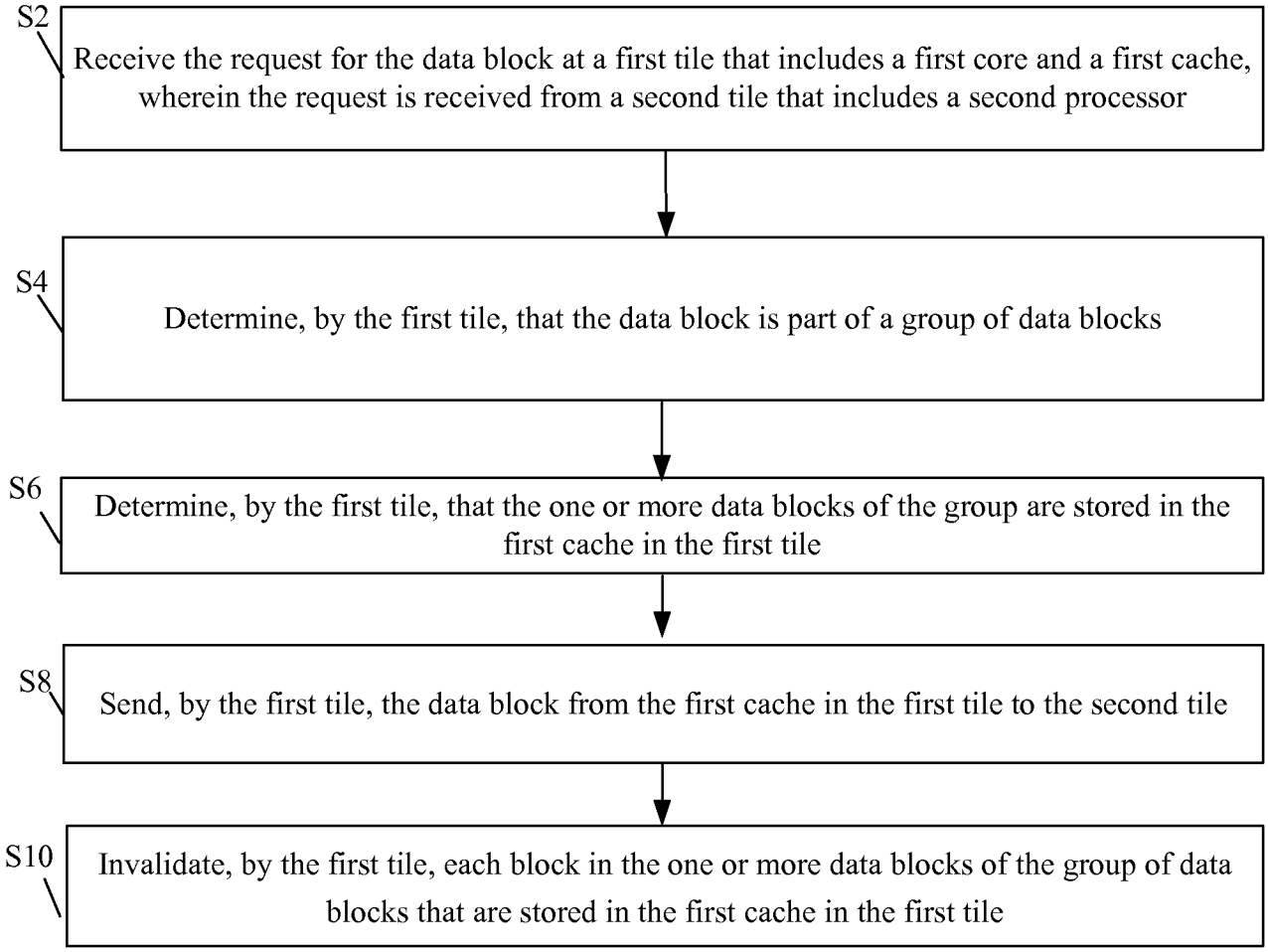

S2
Receive the request for the data block at a first tile that includes a first core and a first cache, wherein the request is received from a second tile that includes a second processor
S4
Determine, by the first tile, that the data block is part of a group of data blocks
S6
Determine, by the first tile, that the one or more data blocks of the group are stored in the first cache in the first tile
S8
Send, by the first tile, the data block from the first cache in the first tile to the second tile
S10
Invalidate, by the first tile, each block in the one or more data blocks of the group of data blocks that are stored in the first cache in the first tile

Fig. 5

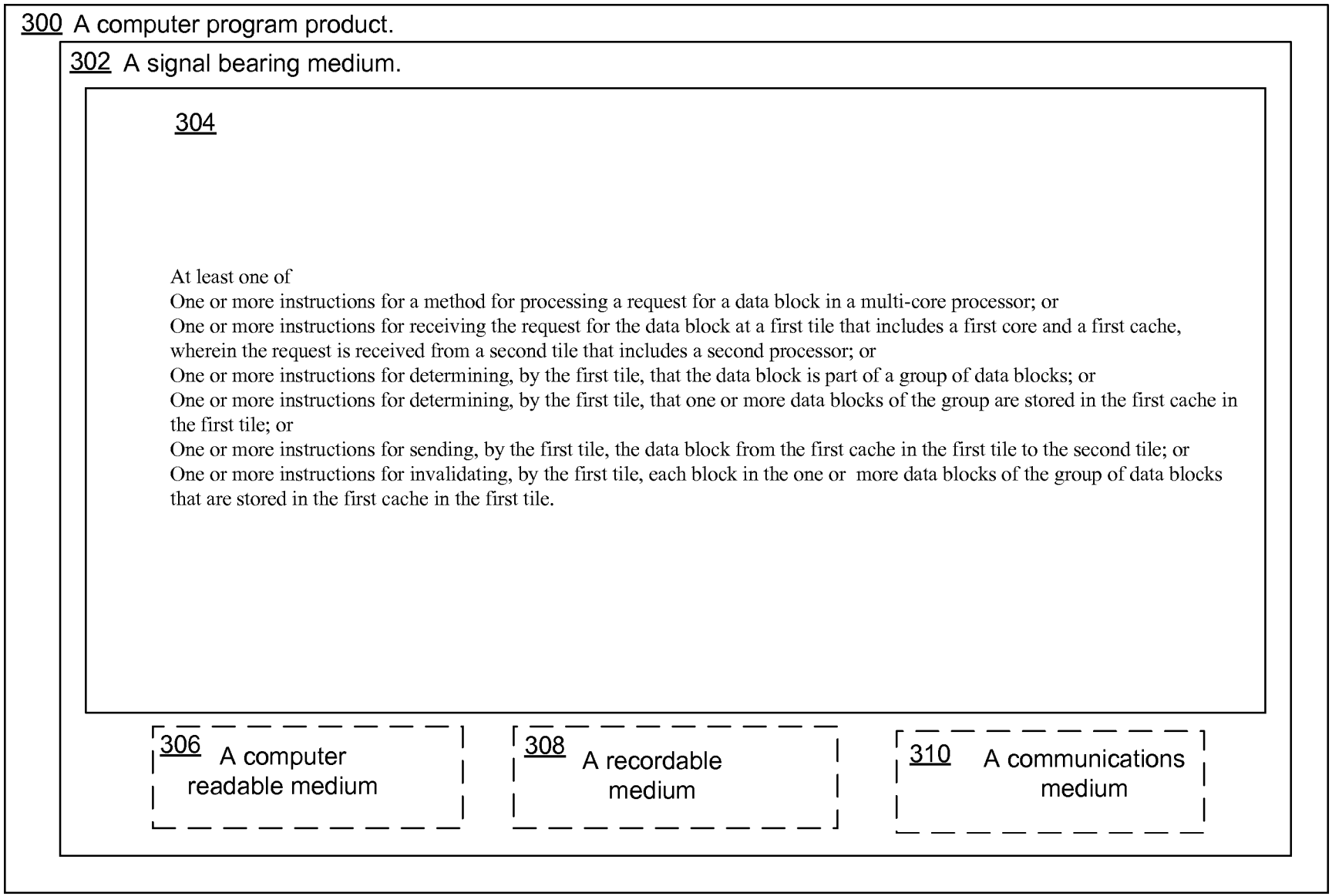
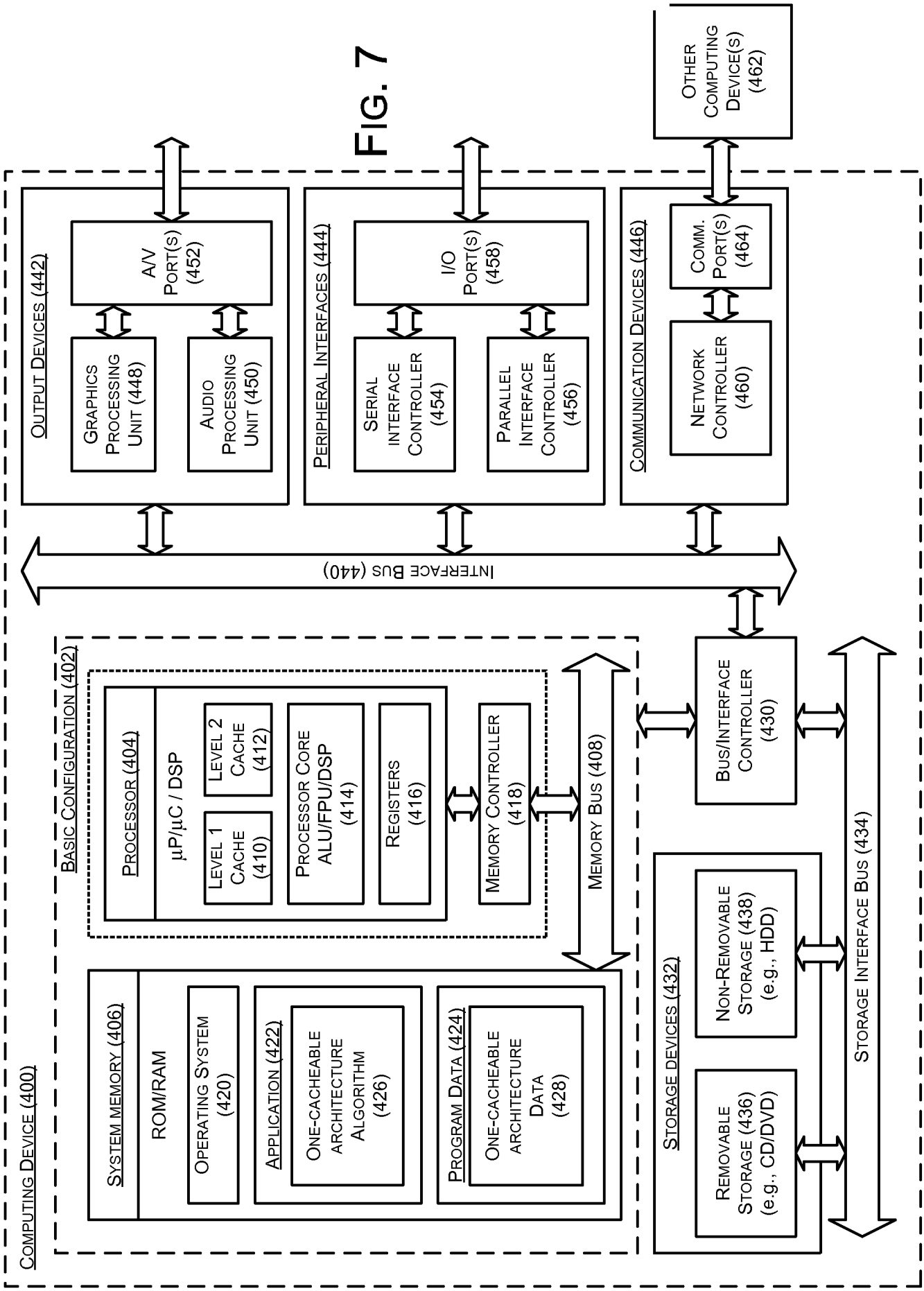

300 A computer program product.
302 A signal bearing medium.
304
At least one of
One or more instructions for a method for processing a request for a data block in a multi-core processor; or
One or more instructions for receiving the request for the data block at a first tile that includes a first core and a first cache, wherein the request is received from a second tile that includes a second processor; or
One or more instructions for determining, by the first tile, that the data block is part of a group of data blocks; or
One or more instructions for determining, by the first tile, that one or more data blocks of the group are stored in the first cache in the first tile; or
One or more instructions for sending, by the first tile, the data block from the first cache in the first tile to the second tile; or
One or more instructions for invalidating, by the first tile, each block in the one or more data blocks of the group of data blocks that are stored in the first cache in the first tile.
306 A computer readable medium
308 A recordable medium
310 A communications medium

Fig. 6

FIG. 7


COMPUTING DEVICE (400)
BASIC CONFIGURATION (402)
PROCESSOR (404)
μP/μC / DSP
LEVEL 1 CACHE (410)
LEVEL 2 CACHE (412)
PROCESSOR CORE ALU/FPU/DSP (414)
REGISTERS (416)
MEMORY CONTROLLER (418)
MEMORY BUS (408)
SYSTEM MEMORY (406)
ROM/RAM
OPERATING SYSTEM (420)
APPLICATION (422)
ONE-CACHEABLE ARCHITECTURE ALGORITHM (426)
PROGRAM DATA (424)
ONE-CACHEABLE ARCHITECTURE DATA (428)
BUS/INTERFACE CONTROLLER (430)
STORAGE DEVICES (432)
REMOVABLE STORAGE (436) (e.g., CD/DVD)
NON-REMOVABLE STORAGE (438) (e.g., HDD)
STORAGE INTERFACE BUS (434)
INTERFACE BUS (440)
OUTPUT DEVICES (442)
GRAPHICS PROCESSING UNIT (448)
AUDIO PROCESSING UNIT (450)
A/V PORT(S) (452)
PERIPHERAL INTERFACES (444)
SERIAL INTERFACE CONTROLLER (454)
PARALLEL INTERFACE CONTROLLER (456)
I/O PORT(S) (458)
COMMUNICATION DEVICES (446)
NETWORK CONTROLLER (460)
COMM. PORT(S) (464)
OTHER COMPUTING DEVICE(S) (462)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/027118

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/08 (2006.01) G06F 13/14 (2006.01)

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)

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPAT, EPODOC, Goolge (Keywords: core, processor, cache, group and similar words)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	Documents are listed in the continuation of Box C	

[X] Further documents are listed in the continuation of Box C [X] 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
17 May 2013	17 May 2013

Name and mailing address of the ISA/AU

AUSTRALIAN PATENT OFFICE
PO BOX 200, WODEN ACT 2606, AUSTRALIA
Email address: pct@ipaustralia.gov.au
Facsimile No.: +61 2 6283 7999

Authorised officer

Tim Yang
AUSTRALIAN PATENT OFFICE
(ISO 9001 Quality Certified Service)
Telephone No. 0262837923

INTERNATIONAL SEARCH REPORT C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		International application No. **PCT/US2013/027118**
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2011/0161346 A1 (SOLIHIN) 30 June 2011 see the abstract; paragraphs 18, 30	1-35
A	US 2007/0239938 A1 (PONG) 11 October 2007	
A	HACKENBERG D., et al; Comparing Cache Architectures and Coherency Protocols on x86-64 Multicore SMP Systems; MICRO'09, December 12–16, 2009, New York, NY, USA.; retrieved from http://people.freebsd.org/~lstewart/articles/cache-performance-x86-2009.pdf on 13 May 2013	

INTERNATIONAL SEARCH REPORT Information on patent family members	International application No. **PCT/US2013/027118**

This Annex lists known patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document/s Cited in Search Report		Patent Family Member/s	
Publication Number	**Publication Date**	**Publication Number**	**Publication Date**
US 2011/0161346 A1	30 Jun 2011	US 2011161346 A1	30 Jun 2011
		US 8244986 B2	14 Aug 2012
		US 2012166735 A1	28 Jun 2012
		US 8407426 B2	26 Mar 2013
		WO 2011090515 A2	28 Jul 2011
US 2007/0239938 A1	11 Oct 2007	US 2007239938 A1	11 Oct 2007
		US 7711902 B2	04 May 2010

End of Annex

Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001.