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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 February 2014 (06.02.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/021995 A1

(51) International Patent Classification:
G06F 9/46 (2006.01) *G06F 15/80* (2006.01)

(21) International Application Number:
PCT/US2013/045129

(22) International Filing Date:
11 June 2013 (11.06.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
3140/CHE/2012 31 July 2012 (31.07.2012) IN

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

(72) Inventor: **VAJAPEYAM, Sriram**; No. 65, Apt. S2, "shyamapriya", H.b. Samaja Road, Basavangudi, Bangalore, Karnataka 560 004 (IN).

(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, 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, KM, ML, MR, NE, SN, TD, TG).

Published:

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

(54) **Title**: THREAD MIGRATION ACROSS CORES OF A MULTI-CORE PROCESSOR

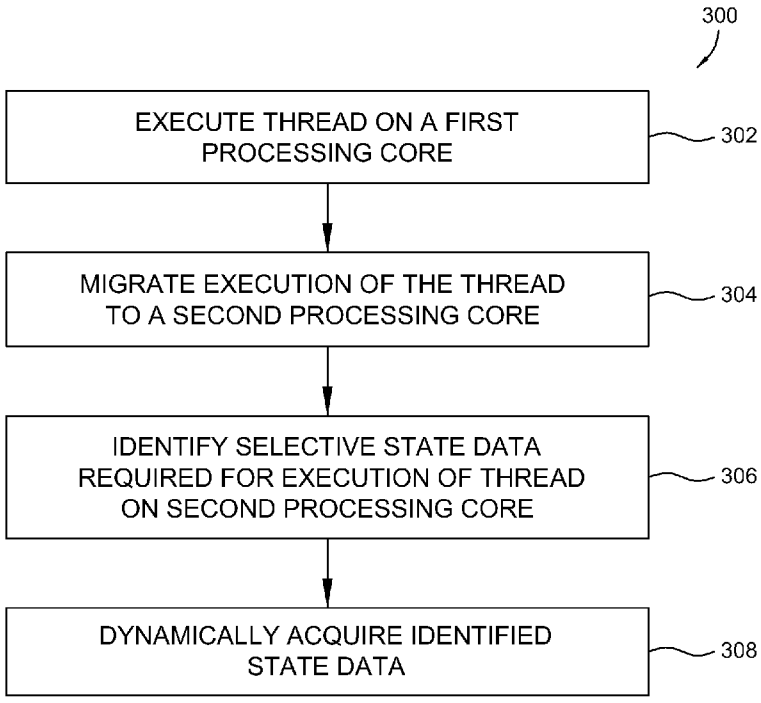


FIG. 3

(57) **Abstract**: Techniques described herein are generally related to thread migration across processing cores of a multi-core processor. Execution of a thread may be migrated from a first processing core to a second processing core. Selective state data required for execution of the thread on the second processing core can be identified and can be dynamically acquired from the first processing core. The acquired state data can be utilized by the thread executed on the second processing core.

WO 2014/021995 A1

Thread Migration Across Cores of a Multi-Core Processor

CROSS- REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to Indian Patent Application 3140/CHE/2012, filed July 31, 2012 entitled THREAD MIGRATION ACROSS CORES OF A MULTI-CORE PROCESSOR, the entirety of which is hereby incorporated by reference.

BACKGROUND

[0002] Unless otherwise indicated herein, the approaches 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.

[0003] Many modern computing systems utilize multi-core processors having two or more processor cores interfaced for enhanced performance and efficient processing of multiple tasks and threads. Thread migration can be utilized in multi-core processors to mitigate issues such as thermal hotspots, cache utilization, load balancing, communication localization, and hardware error tolerance. For example, the execution of a thread may be moved away from a processing core that develops a thermal hotspot during operation. In another example, thread migration is used to move execution of a thread closer to data that the processor accesses for execution of the thread.

[0004] In some examples, when execution of a thread is migrated from a source core to a target core of a multi-core processor, the entire architectural state and micro-architectural state of the source core can be proactively migrated to the target core either through a network message or through data cache coherence operations. For example, registers and translation lookaside buffer (TLB) entries may be stored in data cache of the source core and may be subsequently migrated to data cache of the target core before being transferred to the corresponding registers and TLB of the target core. In other

examples, such values may be stored on on-chip memory such as SRAM. In certain processors, additional machine instructions are executed in both the source core and the target core to achieve this migration of registers and the TLB entries. For example, additional micro-code instructions may be executed in the source core to save the architectural states and the micro-architectural states to data caches. Moreover, micro-code instructions may be executed in the target core to load these values to the corresponding registers and TLB of the target core.

SUMMARY

[0005] The following 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.

[0006] Techniques described herein are generally related to thread migration across processing cores of a multi-core processor. The various described techniques may be applied to methods, systems, devices or combinations thereof. Execution of a thread may be migrated from a first processing core to a second processing core. Selective state data required for execution of the thread on the second processing core can be identified and can be dynamically acquired from the first processing core. The acquired state data can be utilized by the thread executed on the second processing core.

[0007] According to some examples of the present disclosure, various methods related to migrating a thread across cores in a multi-core processor are described. Some example methods may include executing a thread on a first processing core of the multi-core processor and migrating execution of the thread from the first processing core to a second processing core of the multi-core processor. Selective state data required for execution of the thread on the second processing core may be identified and the identified state data may be dynamically acquired from the first processing core. The acquired state data can be utilized by the thread executing on the second processing core.

[0008] According to additional examples of the present disclosure, additional methods related to migrating a thread across cores in a multi-core processor are described. Some example methods may include augmenting state data associated with architectural states and micro-architectural states of two or more of the plurality of processing cores with tags. The execution of a thread can be migrated from a first processing core to a second processing core of the two or more of the plurality of processing cores. A read access of portions of the state data of the first processing core by the thread executed on the second processing core may be detected using the tags associated with the state data of the second processing core. The portions of state data from the first processing core are dynamically acquired in response to detection of the read access and the acquired portions of the state data are used by the thread executing on the second processing core. A status indicator of the portions of the state data in the first processing core can be updated to indicate a migrated status of the data.

[0009] According to still further examples of the present disclosure, multi-core processors devices are described. Some described multi-core processors may include a plurality of processing cores including at least a first processing core and a second processing core. Each of the plurality of processing cores may include at least one architectural state component configured to store state data associated with an architectural state of the processing core and at least one micro-architectural state component configured to store state data associated with a micro-architectural state of the processing core. The state data associated with the architectural state and the micro-architectural state may be implemented with associated tags indicative of a status of the state data as execution of a thread is migrated from the first processing core to the second processing core. Each of the plurality of processing cores may also include a coherence module that may be coupled to the at least one architectural state component and to the at least one micro-architectural state component. The coherence module can be configured to acquire selective state data from the first processing core required for execution of the migrated thread on the second processing core.

BRIEF DESCRIPTION OF THE FIGURES

[0010] In the drawings:

FIG. 1 is a schematic diagram illustrating functional components of an example multi-core processor;

FIG. 2 illustrates an example architectural state component;

FIG. 3 is an illustration of a process for migrating a thread across processing cores of the multi-core processor of FIG. 1;

FIG. 4 is a block diagram illustrating an example implementation of a state coherence protocol to dynamically acquire state data for thread migration from a first processing core to a second processing core of a multi-core processor;

FIG. 5 illustrates example scenarios of thread migration across multiple cores of a multi-core processor; and

FIG. 6 is a block diagram illustrating an example computing device that is arranged for migration of threads across cores of a multi-core processor;

all arranged in accordance with at least some embodiments of the present disclosure.

DETAILED DESCRIPTION

[0011] 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 used, 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.

[0012] Example embodiments of the present disclosure are generally directed to techniques for migrating threads across processing cores in a multi-core processor. The multi-core processor may include two or more processing cores. One or more threads are migrated from a source core to a target core selected from the two or more processing cores. In the described embodiments, a state coherence protocol is utilized to acquire selective state data from the source core while a migrated thread is executed on the target core. The need-based acquisition of the state data may reduce migration-related traffic of the threads and may enhance the performance of the multi-core processor.

[0013] Example Multi-Core Processor: FIG. 1 is a schematic diagram illustrating functional components of an example multi-core processor 100 arranged in accordance with at least some embodiments of the present disclosure. The multi-core processor 100 may be a general-purpose processor, a special-purpose processor, an embedded processor, a graphics processor, or the like. The multi-core processor 100 may include a plurality of processing cores including at least a first processing core 102 and a second processing core 104. Potentially, the number (N) of the processing cores may be any number greater than one. Each of the processing cores (such as the first processing core 102 and the second processing core 104) includes internal functional blocks.

[0014] In some embodiments, the first processing core 102 may include a processor 106A, at least one architectural state component 108A and at least one micro-architectural state component 110A. Similarly, the second processing core 104 may include a processor 106B, at least one architectural state component 108B and at least one micro-architectural state component 110B. The architectural state components 108A and 108B can be configured to store data associated with an architectural state of the first processing core 102 and the second processing core 104, respectively. The architectural states can include states that may be accessible (e.g., for read or write access) to the processing cores while executing machine instructions. The architectural state components 108A and 108B may include architectural registers, control registers, flags, a stack pointer, or combinations thereof.

[0015] Moreover, the micro-architectural state components 110A and 110B can be configured to store data associated with a micro-architectural state of the first processing core 102 and the second processing core 104, respectively. The micro-architectural states can include states that may be accessible (e.g., for read or write access) to a micro-code of the processing cores. The micro-architectural state components 110A and 110B may include physical registers, branch predictor tables, memory disambiguation buffers, translation lookaside buffer (TLB) registers, store buffers, segment registers, a program counter, or combinations thereof. It should be noted that the above examples of the architectural and micro-architectural states components are purely illustrative and states may be classified differently based upon the processor type. In various example embodiments, state data associated with the architectural state and the micro-architectural state of the first processing core 102 and the second processing core 104 includes tags. The tags can be indicative of a status of the state data as execution of a thread 112 is migrated from the first processing core 102 to the second processing core 104. For example, the state data stored in the at least one architectural state component 108A and at least one micro-architectural state component 110A may include tags such as one or more of a valid bit (Vbit), a thread ID (T-Id), a migrated bit (M), an instruction counter (IC), and/or combinations thereof.

[0016] In some embodiments, the execution of the thread 112 may be migrated from the first processing core 102 to the second processing core 104 using a thread migration scheduler (not shown). Each of the first processing core 102 and the second processing core 104 may include a coherence module represented by reference numerals 114A and 114B, respectively. The coherence modules 114A and 114B can be configured to facilitate state coherence while the thread 112 migrates from the first processing core 102 to the second processing core 104. It may be noted that although the technique is described with reference to two processing cores, the present technique may be applicable to migration of thread across more than two processing cores.

[0017] As the execution of the thread 112 migrates from the first processing core 102 to the second processing core 104, the coherence module 114B can acquire selective state data from the first processing core 102 that may be required for execution of the

thread 112 on the second processing core 104. The coherence module 114B can be configured to send a state coherence request to the coherence module 114A of the first processing core 102 over a communication bus such as an interconnection network 116.

[0018] The first processing core 102 and the second processing core 104 may include other functional components. For example, the first processing core 102 may include a local cache 118A. Similarly, the second processing core 104 may include a local cache 118B. The local caches 118A and 118B may include L1 and L2 caches. Moreover, the plurality of processing cores may be coupled to a shared cache 120 (L3 cache) through the interconnection network 116. It should be noted that the multi-core processor 100 could include numerous variations of the architecture described above.

[0019] Example Architectural State Component: The state data associated with an architectural state of each processing core (e.g., first processing core 102 of the multi-core processor 100) can be stored in the architectural state component 108A. Similarly, the state data associated with the micro-architectural state of each processing core (e.g., first processing core 102) can be stored in the micro-architectural state component 110A.

[0020] FIG. 2 illustrates an example architectural state component 200, arranged in accordance with at least some embodiments described herein. Here, the architectural state component 200 may be represented as a register file with tags 202 associated with entries of the register file 200. In various embodiments, one or more values (such as represented by reference numerals 204, 206, 208 and 210) stored in the registers (R0) 212, (R1) 214, (R2) 216 and (R3) 218 respectively of the register file 200 may be associated with tags such as one or more of a valid bit (Vbit) 220, a migrated bit (M) 222, a thread ID (T-Id) 224, and/or an instruction counter (IC) 226. However, other tags or tag structures may be similarly associated with the state data.

[0021] The register file 200 is shown here for illustrative purposes only. The entries of other architectural and micro-architectural state components such as control registers, physical registers, branch predictor tables etc. of the processing cores can be associated with similar tags. Such tags may be utilized to facilitate dynamic acquisition of state data from the first processing core 102 as the migrated thread 112 is executed on the second

processing core 104, as will be described with reference to FIGS. 3 and 4. In certain embodiments, one or more tags described above may not be used during acquisition of the state data.

[0022] Example Process Flow: FIG. 3 is an illustration of an example process 300 for migrating a thread across processing cores (e.g., first processing core 102 and the second processing core 104) of the multi-core processor 100 of FIG. 1, arranged in accordance with at least some embodiments described herein. Process 300 may include one or more operations, functions or actions as illustrated by one or more of blocks 302-308. Although the blocks are illustrated in a sequential order, these blocks may also be performed in parallel, and/or in a different order than those described herein. Also, the various blocks may be combined into fewer blocks, divided into additional blocks, and/or eliminated based upon the desired implementation. Process 300 may begin at block 302.

[0023] At block 302, "EXECUTE THREAD ON A FIRST PROCESSING CORE", a thread (112) can be executed on a first processing core (102) of the multi-core processor 100.

[0024] Processing may continue from block 302 to block 304, "MIGRATE EXECUTION OF THE THREAD TO A SECOND PROCESSING CORE." At block 304, the execution of the thread (112) can be migrated to a second processing core (104). For example, an instruction sequence for the thread (112) can be migrated from the first processing core (102) to the second processing core (104) using a thread migration scheduler. In some embodiments, the migration of the thread (112) may be in accordance with a pre-determined thread migration schedule. In some other embodiments, the migration of the thread (112) to the second processing core (104) may be initiated to move away from thermal hotspots developed during operation of first processing core (102).

[0025] Processing may continue from block 304 to block 306, "IDENTIFY SELECTIVE STATE DATA REQUIRED FOR EXECUTION OF THREAD ON SECOND PROCESSING CORE". At block 306, selective state data that may be required for execution of the thread (112) on the second processing core (104) can be

identified. In some examples, the processor (106B) of the second processing core may access entries of the architectural state component (108B) and/or the micro-architectural state component (110B) for executing a migrated instruction sequence for the thread (112) on the second processing core (104). The processor (106B) may then identify the respective state data required from the first processing core (102). In some examples, the selective state data required for execution of the thread (112) on the second processing core may be a portion of the state data of the first processing core (102). Examples of identification of the state data for the first processing core (102) will be described in a greater detail with reference to FIG. 4.

[0026] In various embodiments, the identified state data may be data associated with architectural states and/or micro-architectural states of the first processing core (102). Such state data may be stored in the at least one architectural state component (108A) and/or at least one micro-architectural state component (110A) of the first processing core (102). In operation, a first portion of the state data of the first processing core (102) can be acquired from the first processing core (102) to initiate the execution of the thread (112) on the second processing core (104). The first portion of the state data may include some values of the state data (e.g., program counter) that may be required to initiate the execution of the thread (112) on the second processing core (104).

[0027] In some examples, the first portion of the state data may include values (204) and (206) corresponding to registers (R0) (212) and (R1) (214) of the first processing core 102. Such values can be copied to the corresponding registers of the second processing core 104. A status indicator of the first portion of the state data in the first processing core (102) can then be updated to a migrated status. In this example, a migrated bit (M) of the first portion of the state data can be updated to indicate the migrated status of the first portion of the state data. For example, once the values (204) and (206) are copied to the corresponding registers of the second processing core (104), then the tag M corresponding to these values in the first processing core (102) can be set to a value (e.g., 1) to indicate the migrated status of these values. Similarly, the tag Vbit corresponding to the values (204) and (206) can be set to a value (e.g., 1) in the second processing core 104 to indicate valid entries of the registers (R0) (212) and (R1) (214).

Further, the tag M corresponding to each of these values can be set to a value (e.g., 0) in the second processing core (104) to indicate that these values have not been migrated yet. Any appropriate set of values as may be utilized (e.g., single bit or multi-bit tags of any desired value), and the above described values are merely examples that are not to be construed as limiting the concepts described herein.

[0028] The execution of the thread (112) can then be initiated on the second processing core (104). The execution of thread 112 may use the acquired first portion of the state data while executing on the second processing core (104). Moreover, a second portion of the state data may be retained with the first processing core (102). For example, the second portion of the state data may include values 208 corresponding to register (R2) (216) of the register file (200).

[0029] In some example embodiments, a read access of one or more parts of the second portion of the state data by the thread (112) executing on the second processing core (104) may be detected by the coherence module (114A) of the first processing core (102). In this example, a read access of the register R2 (216) may be detected. The one or more parts accessed by the thread (112) can be identified as the selective state data required for execution of the thread (112) on the second processing core (104). Here, the values 208 corresponding to the register R2 (214) may be identified as the selective state data. Such data can be dynamically acquired using a state coherence protocol (block 308). Moreover, the status indicator of the one or more parts of the second portion of the state data can subsequently be updated to migrated status.

[0030] In certain other embodiments, the selective state data may be acquired using other communication mechanisms. For example, in certain multi-core processors, a read access of a register of a target processing core that is marked as "Invalid" may trigger a messaging unit to send a request for the corresponding value to the source processing core. In another example, a read access of an "Invalid" register may result in a trap to the operating system, e.g., an interrupt, so that the operating system can direct the processing core to implement the functionality of the messaging unit.

[0031] In some alternate embodiments, a write access of one or more parts of the second portion of the state data by the thread (112) executing on the second processing core (104) may be detected by the coherence module (114B) of the second processing core (104). For example, a write access to values (210) of the register (R3) (218) of the second processing core (104) may be detected. An invalidation message identifying the one or more parts of the second portion of the state data such as values (210) can be sent to the first processing core (102) and the status indicator of the one or more parts of the second portion of the state data in the first processing core (102) can be updated to migrated status (e.g., setting the tag M to 1). In various example embodiments, the thread (112) executing on the second processing core (104) may overwrite the values (210) of the register R3 (218) without reading the values from the first processing core (102). In certain embodiments, when a write access to valid state entries of the second processing core (104) is detected, invalidation messages may not be sent to the first processing core (102).

[0032] As described above, the state data acquired from the first processing core (102) may be used by the thread (112) that is executed on the second processing core (104). The dynamic acquisition of the state data from the first processing core (102) will be described in a greater detail with reference to FIG. 4.

[0033] Dynamic Acquisition of State Data: FIG. 4 is a block diagram illustrating an example implementation 400 of a state coherence protocol to dynamically acquire state data for thread migration from the first processing core 102 to the second processing core 104 of the multi-core processor 100 of FIG. 1, arranged in accordance with at least some embodiments described herein. In various illustrated embodiments, execution of the thread 112 can be migrated from the first processing core 102 to the second processing core 104. In some example embodiments, the first processing core 102 includes an architectural state component such as a register file 200A and the second processing core 104 includes a register file 200B. For ease of description, a register file can be utilized to depict an architectural state component of each of the processing cores 102 and 104. However, the same technique applies to other architectural and micro-architectural state components.

[0034] Initially, entries/values such as 204A, 206A, 208A and 210A of registers (R0) 212A, (R1) 214A, (R2) 216A and (R3) 218A of the register file 200A can be associated with tags such as Vbit 220A and M 222A. Similarly, values (such as represented by reference numerals 204B, 206B, 208B and 210B) of the registers (R0) 212B, (R1) 214B, (R2) 216B and (R3) 218B of the register file 200B can be associated with tags such as (Vbit) 220B and (M) 222B. As the execution of the thread 112 is migrated to the second processing core 104, the tags of the values associated with the architectural states and the micro-architectural states on the second processing cores can be reset. As can be seen, the (Vbit) 220B of the value 204B of the register 212B of the second processing core 104 can be set as "false" (e.g., Vbit=0).

[0035] Once the execution of the thread 112 is initiated on the second processing core 104, the processor 106B accesses the value 204B of the register file 200B (operation 402). The thread 112 encounters a "register miss" as the Vbit of the value 204B can be set as "false" (operation 404). Similarly, the thread may encounter a "TLB miss" as the T-Id of the value does not match with the T-Id of the executing thread 112. A state coherence request can be sent to the coherence module 114B of the second processing core 104 to trigger the state coherence protocol (operation 406). The state coherence request can be sent from the coherence module 114B to the coherence module 114A of the first processing core 102 for acquiring the respective value 204A of the corresponding register 212A from the first processing core 102 (operation 408).

[0036] The coherence module 114A of the first processing core 102, upon receiving the state coherence request, can detect a valid entry (Vbit=1) 220A on the register (R0) 212A of the register file 200A of the first processing core 102 (operation 410). The coherence module 114A can then access the corresponding value 204A of the register (R0) 212A from the register file 200A (operation 412). The status of the entry 204A can then be updated to a migrated status by setting the migrated bit (Mbit) 222A (operation 414). The value of the entry 204A can then be passed on to the coherence module 114B of the second processing core 104 (operation 416). The value then replaces the current value 204B of the register (R0) 212B of the register file 200B (operation 418) and the Vbit 220B of the corresponding entry can be set to be valid (operation 420). This

updated value 204B can be utilized by the processor 106B while executing the thread 112 on the second processing core 104 (operation 422). Such selective acquisition of values of required entries of architectural and/or micro-architectural states is performed during execution of the thread 112 on the second processing core 104. Further, once status of all the entries of the first processing core 102 is updated to the migrated status, the corresponding entries may be removed from the first processing core 102. In some examples, the Vbits 220 of the entries may be set as "false" and the entries may be removed in a single-cycle flush operation, and the operating system and/or a thread scheduler may be notified that the first processing core 102 is available to receive a new thread.

[0037] Thread Migration Across More than Two Processing Cores: The embodiments illustrated above describe dynamic acquisition of portions of state data associated with architectural states and/or micro-architectural states when the thread migrates from a first processing core 102 to a second processing core 104 of a multi-core processor 100. In certain embodiments, a thread may migrate multiple times and across more than two cores.

[0038] FIG. 5 illustrates example scenarios 500 of thread migration across multiple cores of a multi-core processor 502, arranged in accordance with at least some embodiments described herein. As illustrated, the multi-core processor 502 may include four processing cores generally represented by reference numerals 504, 506, 508 and 510. A thread 512 can be migrated across multiple processing cores as illustrated in example implementations 514, 516, 518 and 520. As can be seen, the execution of the thread 512 can be migrated from the processing core 504 to the processing core 506 and subsequently from the processing core 506 to the processing core 510. Moreover, execution of another thread 522 may be initiated on the core 504 and subsequently migrated to the core 506.

[0039] As described with reference to FIG. 2, one or more values stored in the architectural state components and the micro-architectural state components may be associated with the instruction counter (IC) 226. The instruction counter 226 may be

utilized for acquisition of values during migration of the thread 512 across multiple processing cores such as the processing cores 504, 506, 508, and 510. The instruction counter 226 provides a dynamic instruction number to instructions that produced the value of an entry of the architectural state components and the micro-architectural state components. In some examples, the dynamic instruction number can be maintained in a register and may be incremented each time an instruction of the program is executed, thereby tracking the number of instructions executed by the program. As the execution of thread 512 migrates across the processing cores 504, 506, and 510, different versions of the architectural states and the micro-architectural states having different IC values are available in the processing cores 504, 506, and 510.

[0040] For example, the current processing core (e.g., processing core 510 in example implementation 520) may initiate the state coherence request to acquire value 210 for the register 218 of the core 510. In some embodiments, each of the cores 504, 506, and 508 can respond to the state coherence request and can send the corresponding values for the register 218 to the processing core 510. The current processing core 510 can receive values from various cores (such as cores 504, 506 and 508) and can also acquire the latest value among the received values for the register 218. In some other embodiments, the IC value for the values from each core (such as 504 and 506) can be compared and the values that correspond to the highest IC value can be acquired by the processing core 510 thereby ensuring acquisition of latest value of the entry. In yet other embodiments, the processing core 510 may send a state coherence request directly to another core (e.g., processing core 504) to acquire the corresponding value from the processing core 504.

[0041] In certain other embodiments, other tags/data structures may be used to facilitate the state coherence as the thread migrates across two or more cores. For example, as the thread 512 is executed on the processing core 504, a state bit vector indicative of each modified state component can be created and stored. Once the thread 512 migrates from the processing core 504 to the processing core 506, the previously stored state bit vector can also be migrated with the thread 512 to the processing core 506. The state bit vector may be used by the processing core 506 for optimizing certain

operations such as reducing the number of invalidation messages sent to the processing core 504.

[0042] A similar state bit vector corresponding to the thread activity on the processing core 506 may also be created. As the thread 512 migrates from the processing core 506 to the processing core 510, both bit vectors can be migrated to the processing core 510 and a state coherence protocol can utilize the two bit vectors to determine the processing core (504, or 506) to which a state coherence request may be sent by the processing core 510.

[0043] The state coherence protocol described herein may be physically and/or logically integrated with a standard cache coherence protocol employed by the multi-core processor. In some examples, the physical integration of the state coherence protocol can be implemented by having the state coherence protocol and the state coherence bits/tags (e.g., Vbit, Mbit and IC) along with the cache coherence protocol. In various examples, the logical integration of the state coherence protocol may be implemented by memory mapping state bits addresses while sending state coherence requests to coherence modules such that a single coherence protocol can handle both memory/cache coherence and state coherence. The various techniques described herein may result in reduced implementation costs for design and verification of the protocol.

[0044] The example methods and systems described above provide an on-chip thread migration mechanism that implements a state coherence protocol to dynamically acquire state data associated with architectural and/or micro-architectural states from a source core while a migrated thread is executed on a target core. The need-based acquisition of state data may reduce the amount of data transferred to the target core and speed up the migration process by reducing the migration-related traffic.

[0045] Example Computing Device: FIG. 6 is a block diagram illustrating an example computing device 600 that is arranged for migration of threads across cores of a multi-core processor in accordance with at least some embodiments of the present disclosure. In a very basic configuration 602, the computing device 600 typically includes one or more processors 604 and a system memory 606. A memory bus 608 may

be used for communicating between processor 604 and system memory 606. The processor 604 includes a multi-core processor.

[0046] Depending on the desired configuration, processor 604 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 604 may include one more levels of caching, such as a level one cache 610 and a level two cache 612, two or more processor cores 614, and registers 616. An example processor core 614 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 618 may also be used with processor 604, or in some implementations memory controller 618 may be an internal part of processor 604. The processor 604 may include a coherence module such as described above to facilitate selective acquisition of state data from the registers 616 during migration of a thread across two or more processing cores of the processor 604.

[0047] Depending on the desired configuration, system memory 606 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 106 may include an operating system 620, one or more applications 622, and program data 624. In some embodiments, application 622 may be arranged to operate with program data 624 on operating system 620. This described basic configuration 602 is illustrated in FIG. 6 by those components within the inner dashed line.

[0048] Computing device 600 may have additional features or functionality, and additional interfaces to facilitate communications between basic configuration 602 and any required devices and interfaces. For example, a bus/interface controller 630 may be used to facilitate communications between basic configuration 602 and one or more data storage devices 632 via a storage interface bus 634. Data storage devices 632 may be removable storage devices 636, non-removable storage devices 638, or a combination thereof.

[0049] 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.

[0050] System memory 606, removable storage devices 636 and non-removable storage devices 638 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 600. Any such computer storage media may be part of computing device 600.

[0051] Computing device 6600 may also include an interface bus 640 for facilitating communication from various interface devices (e.g., output devices 642, peripheral interfaces 644, and communication devices 646) to basic configuration 602 via bus/interface controller 630. Example output devices 642 include a graphics processing unit 648 and an audio processing unit 650, which may be configured to communicate to various external devices such as a display or speakers via one or more A/V ports 652.

[0052] Example peripheral interfaces 644 include a serial interface controller 654 or a parallel interface controller 656, 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 658. An example communication device 646 includes a network controller 660, which may be arranged to facilitate communications with one or more other computing devices 662 over a network communication link via one or more communication ports 664.

[0053] 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.

[0054] Computing device 600 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 600 may also be implemented as a personal computer including both laptop computer and non-laptop computer configurations.

[0055] 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.

[0056] 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.

[0059] 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.

[0060] 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.).

[0061] 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."

[0062] 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.

[0063] 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.

[0064] 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

1. A method for migrating a thread across cores in a multi-core processor, the method comprising:

executing a thread on a first processing core of a multi-core processor;

migrating execution of the thread from the first processing core to a second processing core of the multi-core processor;

identifying selective state data required for execution of the thread on the second processing core; and

dynamically acquiring identified state data from the first processing core, wherein the acquired state data is utilized by the thread executing on the second processing core.

2. The method of claim 1, wherein identifying selective state data comprises identifying state data required from the first processing core by a processor of the second processing core.

3. The method of claim 1, wherein identifying selective state data comprises identifying a portion of the state data of the first processing core.

4. The method of claim 1, wherein acquiring identified state data comprises acquiring the identified state data from the first processing core using a coherence module of the second processing core.

5. The method of claim 4, further comprising transferring the identified state data from the first processing core to the second processing core over a communication bus coupled to the first processing core and to the second processing core.

6. The method of claim 1, wherein dynamically acquiring the identified state data comprises acquiring data associated with architectural states and/or micro-architectural states of the first processing core.

7. The method of claim 6, wherein acquiring data associated with architectural states comprises acquiring one or more values stored in architectural registers, control registers, flags, a stack pointer, or combinations thereof.

8. The method of claim 6, wherein acquiring data associated with the micro-architectural states comprises acquiring one or more values stored in physical registers, branch predictor tables, memory disambiguation buffers, translation lookaside buffer (TLB) registers, store buffers, segment registers, a program counter, or combinations thereof.

9. The method of claim 1, wherein migrating execution of the thread from the first processing core to the second processing core comprises:

acquiring a first portion of the state data from the first processing core by the second processing core prior to initiating execution of the thread on the second processing core;

updating a status indicator of the first portion of the state data in the first processing core to a migrated status; and

initiating execution of the thread in the second processing core using the acquired first portion of the state data.

10. The method of claim 1, wherein identifying selective state data comprises:

detecting a read access of one or more parts of a second portion of the state data by the thread executing on the second processing core; and

identifying the one or more parts of the second portion of the state data in the first processing core as the selective state data required for execution of the thread on the second processing core.

11. The method of claim 10, wherein dynamically acquiring the identified state data comprises:

acquiring the one or more parts of the second portion of the state data from the first processing core; and

updating the status indicator of the one or more parts of the second portion of the state data in the first processing core to a migrated status.

12. The method of claim 1, wherein identifying selective state data comprises:

detecting a write access of one or more parts of the second portion of the state data by the thread executing on the second processing core;

sending an invalidation message identifying the one or more parts of the second portion of the state data to the first processing core; and

updating the status indicator of the one or more parts of the second portion of the state data in the first processing core to a migrated status.

13. The method of claim 1, wherein dynamically acquiring the identified state data comprises acquiring the state data from the first processing core through a state coherence protocol.

14. A method for migrating a thread across cores in a multi-core processor, the method comprising:

augmenting state data associated with architectural states and micro-architectural states of two or more processing cores of a multi-core processor with one or more tags;

migrating execution of a thread from a first processing core to a second processing core of the two or more processing cores;

detecting a read access of portions of the state data of the first processing core by the thread executing on the second processing core using the tags associated with the state data of the second processing core;

dynamically acquiring the portions of the state data from the first processing core in response to detection of the read access, wherein the acquired portions of the state data are utilized by the thread executing on the second processing core; and

updating a status indicator of the portions of the state data in the first processing core to indicate a migrated status of the data.

15. The method of claim 14, wherein augmenting state data comprises associating the tags with one or more values of the architectural states and micro-architectural states by a coherence module of the respective processing core.

16. The method of claim 14, wherein augmenting state data associated with architectural states comprises associating the tags with one or more values stored in architectural registers, control registers, flags, a stack pointer, or combinations thereof.

17. The method of claim 14, wherein augmenting state data associated with micro-architectural states comprises associating the tags with one or more values stored in physical registers, branch predictor tables, memory disambiguation buffers, translation lookaside buffer (TLB) registers, store buffers, segment registers, a program counter, or combinations thereof.

18. The method of claim 14, wherein augmenting state data associated with the architectural states and the micro-architectural states with tags comprises augmenting the state data with a valid bit (Vbit), a thread ID (T-Id), an instruction counter (IC), or combinations thereof.

19. The method of claim 14, wherein migrating execution of the thread comprises migrating the thread using a thread migration scheduler such that the thread migration scheduler facilitates migration of an instruction sequence for the thread from the first processing core to the second processing core.

20. The method of claim 14, wherein detecting a read access of portions of the state data comprises detecting the read access of the state data of the first processing core by a coherence module of the second processing core.

21. The method of claim 14, wherein acquiring portions of the state data comprises acquiring the state data using a coherence module of the second processing core.

22. The method of claim 21, further comprising transferring the state data from the first processing core to the second processing core using a communication bus coupled to the first processing core and to the second processing core

23. The method of claim 14, wherein dynamically acquiring the portions of the state data associated with the architectural states and/or micro-architectural states comprises acquiring the portions of the state data using a state coherence protocol.

24. The method of claim 14, further comprising:

migrating execution of the thread from the second processing core to a third processing core selected from the two or more processing cores of the multi-core processor;

comparing an instruction counter of the state data associated with the architectural states and/or the micro-architectural states of the first processing core and the second processing core to identify a selected processing core with a relatively high value of the instruction counter;

acquiring the state data associated with the architectural states and/or the micro-architectural states of the selected processing core; and

executing the thread on the third processing core using the acquired state data from the selected processing core.

25. The method of claim 14, wherein updating the status indicator comprises setting a migrated bit of the portions of the state data by the coherence module of the first processing core.

26. A multi-core processor, comprising:

a plurality of processing cores including at least a first processing core and a second processing core, each of the plurality of processing cores comprising:

at least one architectural state component configured to store state data associated with an architectural state of the processing core;

at least one micro-architectural state component configured to store state data associated with a micro-architectural state of the processing core, wherein the state data associated with the architectural state and the micro-architectural state comprises associated tags indicative of a

status of the state data as execution of a thread is migrated from the first processing core to the second processing core; and

a coherence module that is coupled to the at least one architectural state component and to the at least one micro-architectural state component, wherein the coherence module is configured to acquire selective state data from the first processing core required for execution of the migrated thread on the second processing core.

27. The multi-core processor of claim 26, wherein the multi-core processor comprises either a general-purpose processor, a special-purpose processor, a graphics processor, or an embedded processor.

28. The multi-core processor of claim 26, wherein the at least one architectural state component comprises architectural registers, control registers, flags, a stack pointer, or combinations thereof.

29. The multi-core processor of claim 26, wherein the at least one micro-architectural state component comprises physical registers, branch predictor tables, memory disambiguation buffers, translation lookaside buffer (TLB) registers, store buffers, segment registers, a program counter, or combinations thereof.

30. The multi-core processor of claim 26, wherein the tags associated with the architectural states and the micro-architectural states comprise a valid bit (Vbit), a thread ID (T-Id), an instruction counter (IC), or combinations thereof

31. The multi-core processor of claim 26, wherein the coherence module of the first processing core is further configured to detect a read access of portions of the state data of the first processing core by the thread executing on the second processing core, and also configured to identify the respective portions of the state data as the selective state data.

32. The multi-core processor of claim 31, further comprising a communication bus that is coupled to the first processing core and the second processing core,

wherein the coherence module of the second processing core is further configured to send a state coherence request to the first processing core over the communication bus.

33. The multi-core processor of claim 32, wherein the coherence module of the second processing core is further configured to acquire the selective state data from the first processing core over the communication bus, and also configured to update the status of the acquired state data in the first processing core to a migrated state.

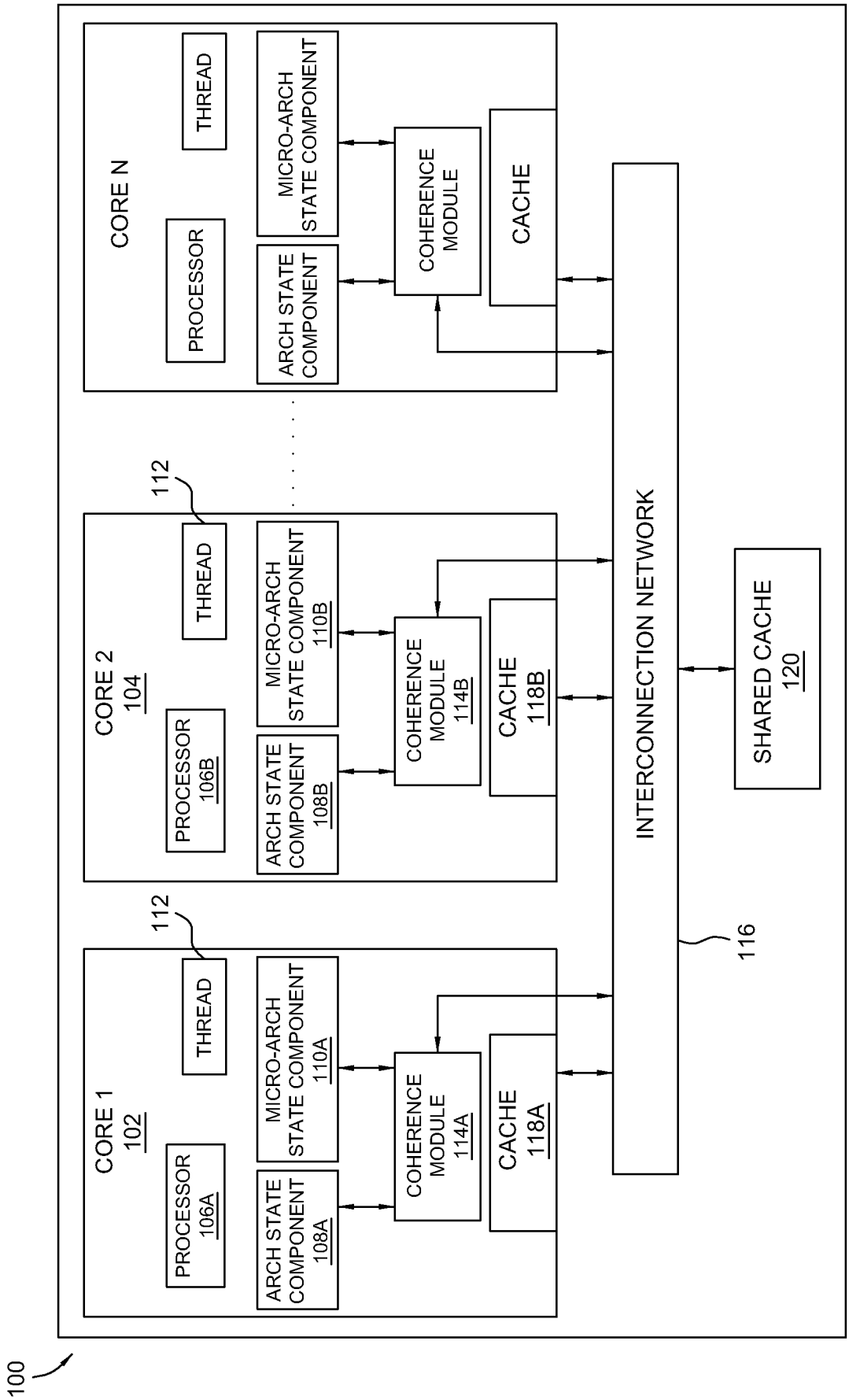

100
CORE 1
102
PROCESSOR
106A
THREAD
112
ARCH STATE
COMPONENT
108A
MICRO-ARCH
STATE COMPONENT
110A
COHERENCE
MODULE
114A
CACHE
118A
CORE 2
104
PROCESSOR
106B
THREAD
ARCH STATE
COMPONENT
108B
MICRO-ARCH
STATE COMPONENT
110B
COHERENCE
MODULE
114B
CACHE
118B
CORE N
PROCESSOR
THREAD
ARCH STATE
COMPONENT
MICRO-ARCH
STATE COMPONENT
COHERENCE
MODULE
CACHE
INTERCONNECTION NETWORK
116
SHARED CACHE
120

FIG. 1

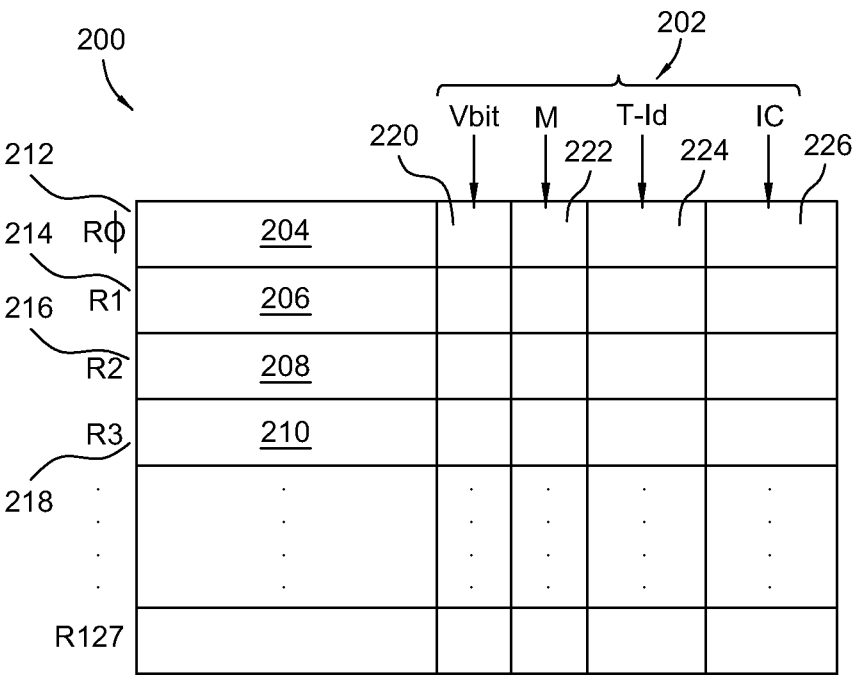

200
202
Vbit
M
T-Id
IC
220
222
224
226
212
214
R0
216
R1
R2
R3
218
R127
204
206
208
210

FIG. 2

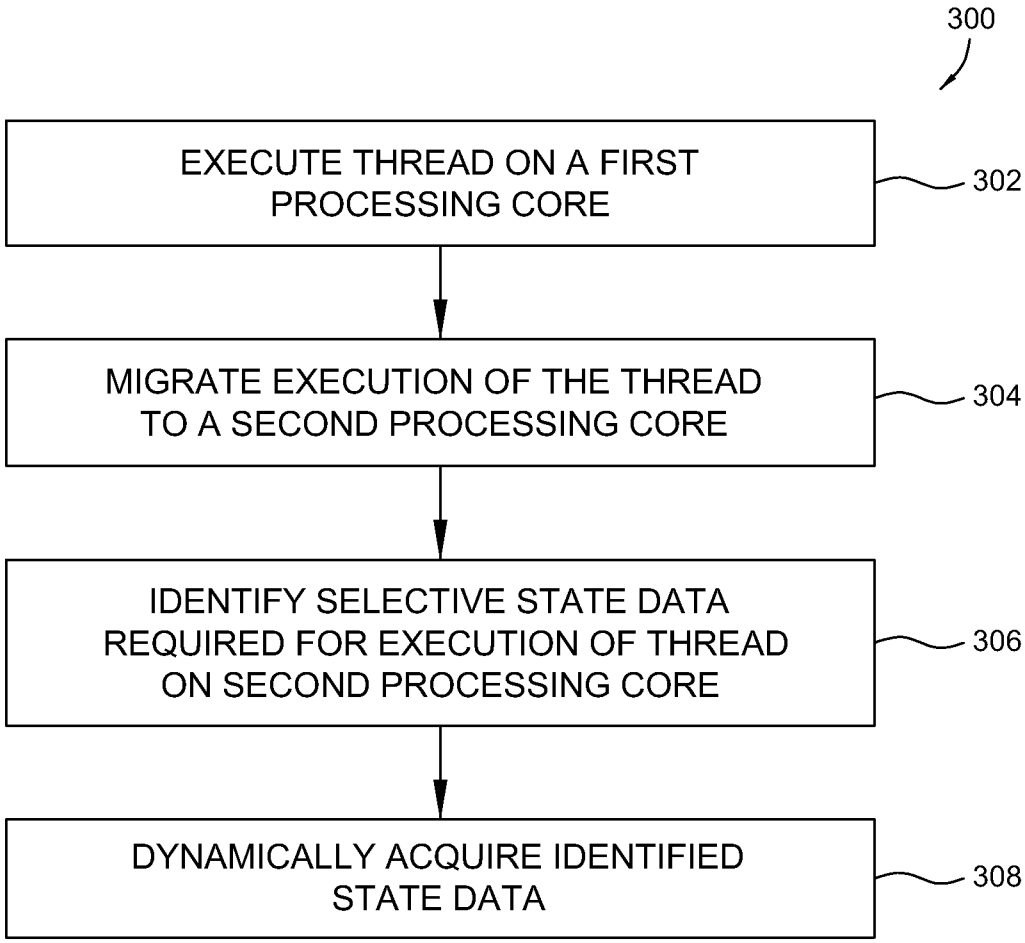

300
EXECUTE THREAD ON A FIRST PROCESSING CORE
302
MIGRATE EXECUTION OF THE THREAD TO A SECOND PROCESSING CORE
304
IDENTIFY SELECTIVE STATE DATA REQUIRED FOR EXECUTION OF THREAD ON SECOND PROCESSING CORE
306
DYNAMICALLY ACQUIRE IDENTIFIED STATE DATA
308

FIG. 3

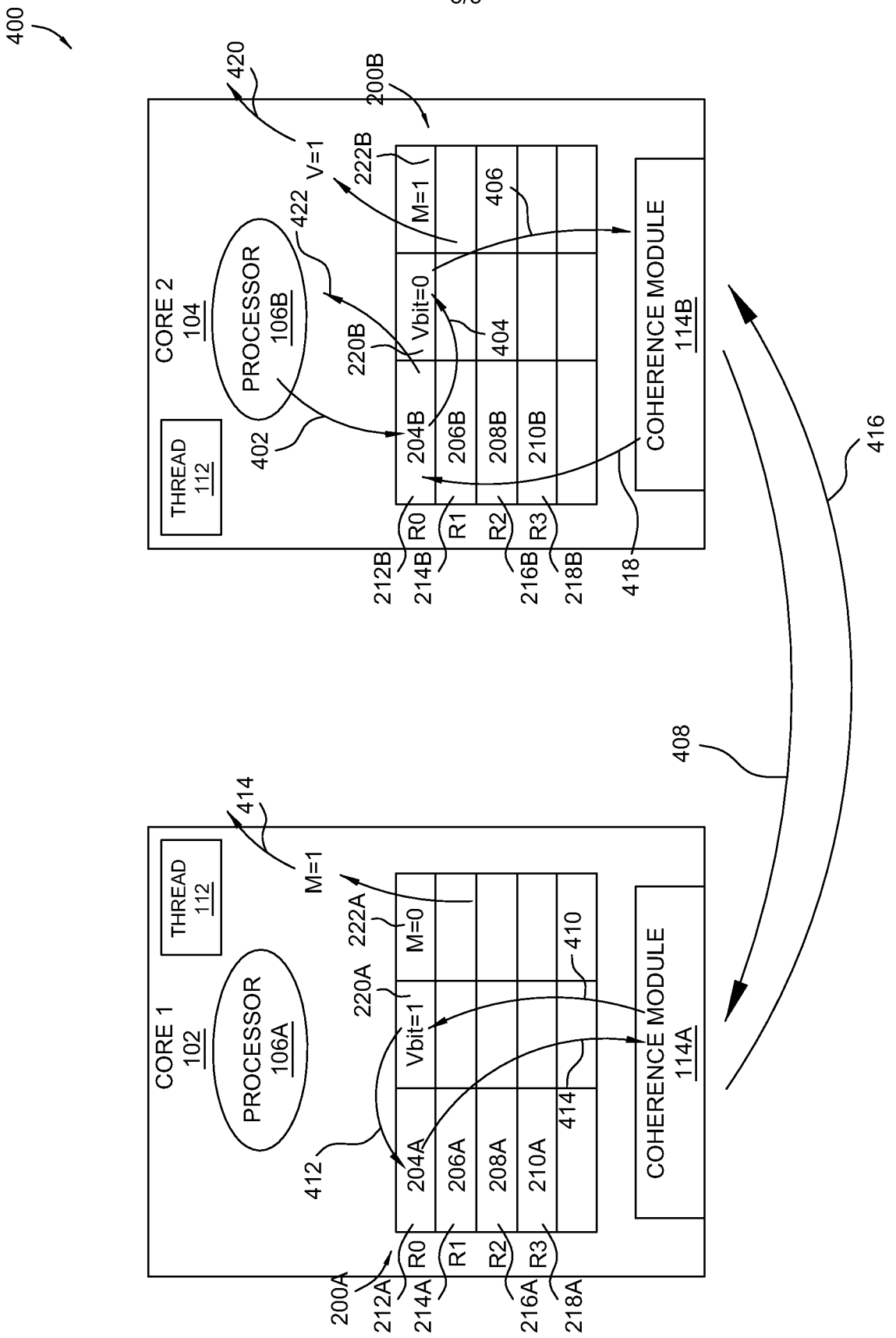

400
CORE 1
102
THREAD
112
PROCESSOR
106A
CORE 2
104
PROCESSOR
106B
THREAD
112
200A
200B
204A
206A
208A
210A
204B
206B
208B
210B
212A
214A
216A
218A
212B
214B
216B
218B
R0
R1
R2
R3
220A
220B
222A
222B
Vbit=1
M=0
Vbit=0
M=1
V=1
402
404
406
408
410
412
414
416
418
420
422
COHERENCE MODULE
114A
COHERENCE MODULE
114B

FIG. 4

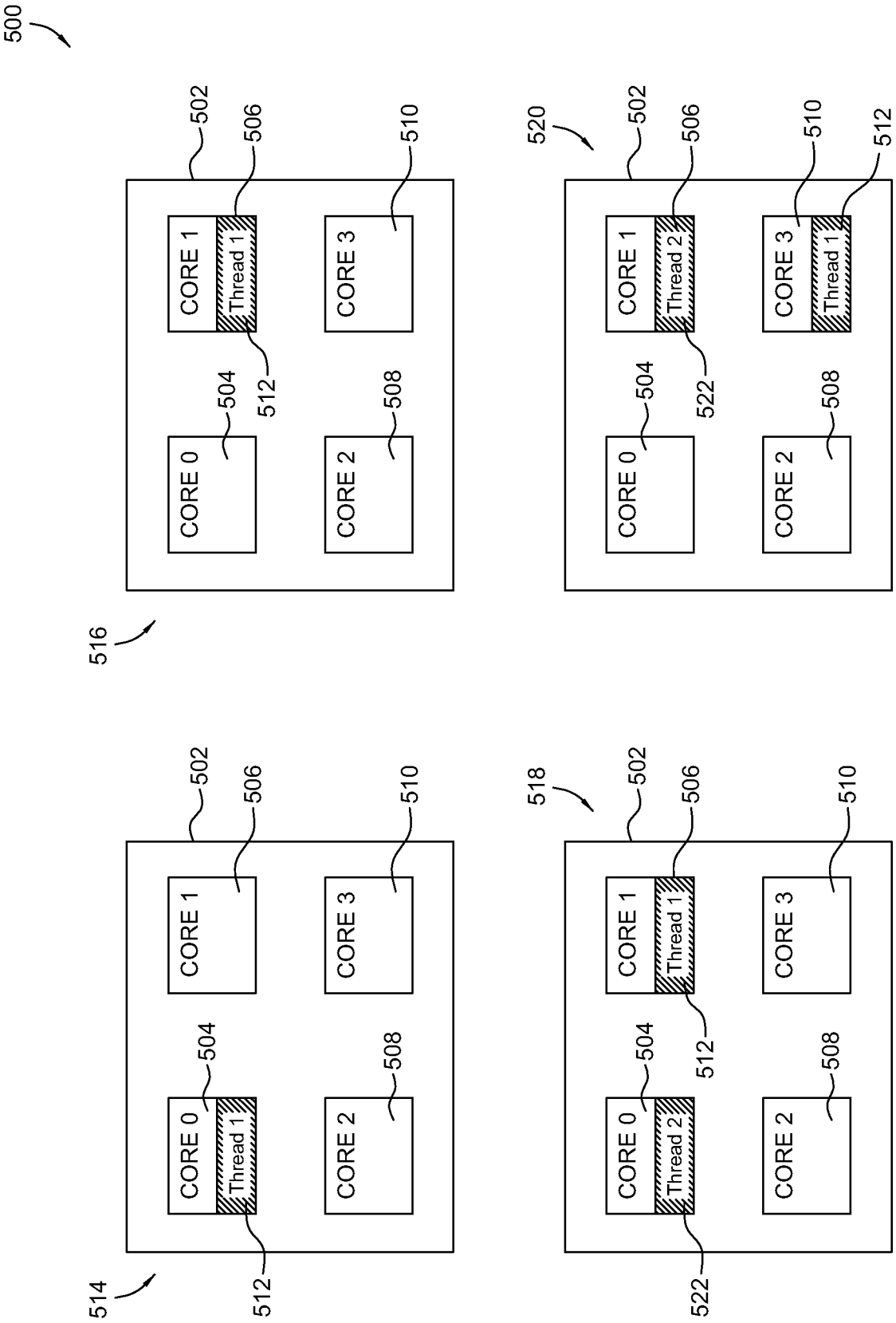

500
502
506
510
CORE 1
Thread 1
CORE 3
504
512
508
CORE 0
CORE 2
516
520
502
506
510
512
CORE 1
Thread 2
CORE 3
Thread 1
504
522
508
CORE 0
CORE 2
514
502
506
510
CORE 1
CORE 3
504
508
CORE 0
Thread 1
CORE 2
512
518
502
506
510
CORE 1
Thread 1
CORE 3
504
512
508
CORE 0
Thread 2
CORE 2
522

FIG. 5

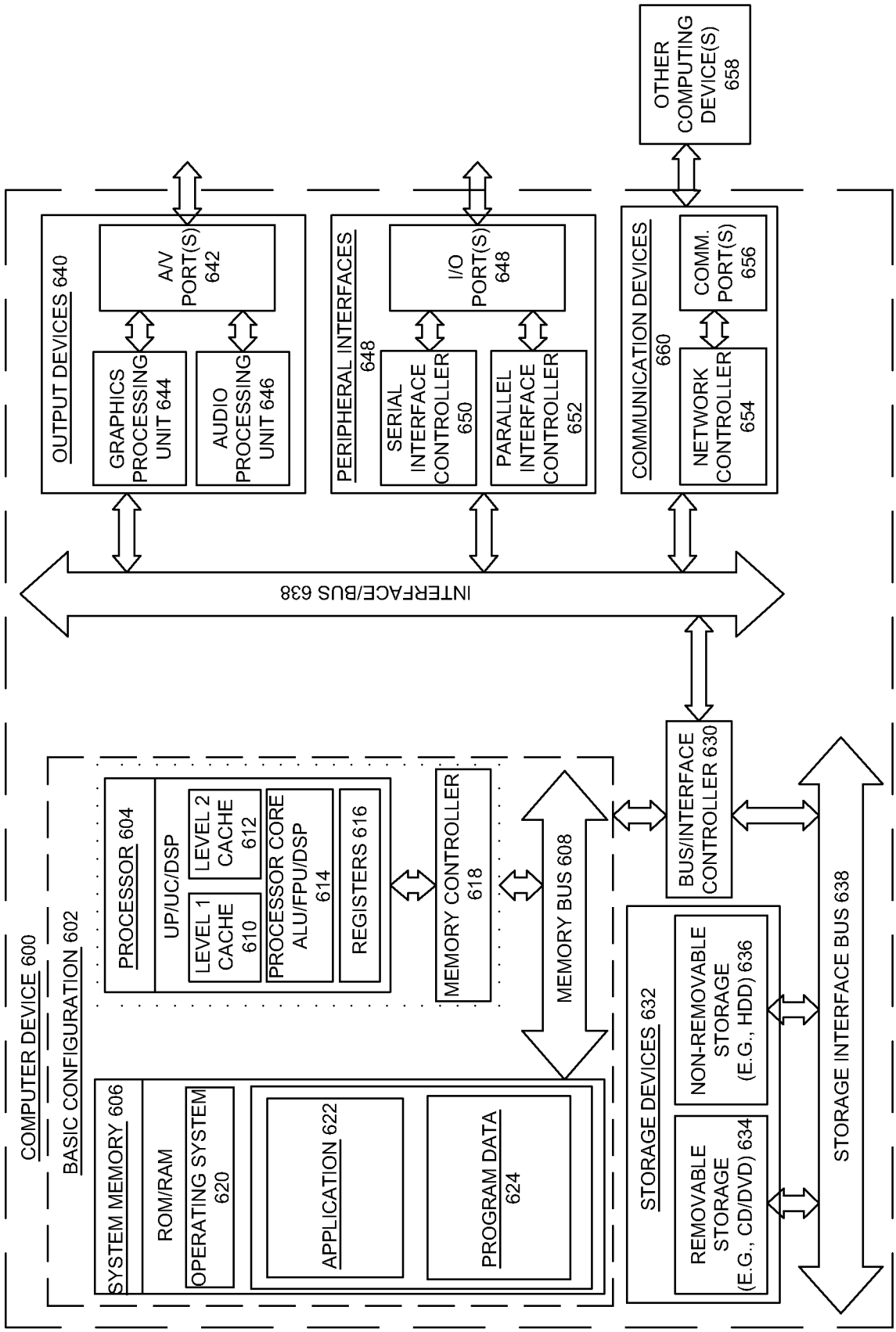

COMPUTER DEVICE 600
BASIC CONFIGURATION 602
PROCESSOR 604
UP/UC/DSP
LEVEL 1 CACHE 610
LEVEL 2 CACHE 612
PROCESSOR CORE ALU/FPU/DSP 614
REGISTERS 616
MEMORY CONTROLLER 618
SYSTEM MEMORY 606
ROM/RAM
OPERATING SYSTEM 620
APPLICATION 622
PROGRAM DATA 624
MEMORY BUS 608
BUS/INTERFACE CONTROLLER 630
STORAGE DEVICES 632
REMOVABLE STORAGE (E.G., CD/DVD) 634
NON-REMOVABLE STORAGE (E.G., HDD) 636
STORAGE INTERFACE BUS 638
INTERFACE/BUS 638
OUTPUT DEVICES 640
A/V PORT(S) 642
GRAPHICS PROCESSING UNIT 644
AUDIO PROCESSING UNIT 646
PERIPHERAL INTERFACES 648
I/O PORT(S) 648
SERIAL INTERFACE CONTROLLER 650
PARALLEL INTERFACE CONTROLLER 652
COMMUNICATION DEVICES 660
COMM. PORT(S) 656
NETWORK CONTROLLER 654
OTHER COMPUTING DEVICE(S) 658

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/045129

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/46 (2006.01) G06F 15/80 (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)

WPI, EPODOC, Google, Google Patents and Patent lens: Keywords (multi-core, processor, migrate, thread and like terms)

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
13 September 2013	13 September 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

Amardeep Singh
AUSTRALIAN PATENT OFFICE
(ISO 9001 Quality Certified Service)
Telephone No. 0262832803

INTERNATIONAL SEARCH REPORT C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		International application No. **PCT/US2013/045129**
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2012/029111 A1 (FUJITSU LIMITED) 08 March 2012 & US 2013/0179666 A1 The whole document	1-33
A	US 2011/0066830 A1 (WOLFE et al.) 17 March 2011 The whole document	1-33

INTERNATIONAL SEARCH REPORT Information on patent family members	International application No. **PCT/US2013/045129**

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**
WO 2012/029111 A1	08 Mar 2012	EP 2613269 A1	10 Jul 2013
		US 2013179666 A1	11 Jul 2013
		WO 2012029111 A1	08 Mar 2012
US 2011/0066830 A1	17 Mar 2011	JP 2013501296 A	10 Jan 2013
		KR 20120024974 A	14 Mar 2012
		US 2011066830 A1	17 Mar 2011
		WO 2011031355 A1	17 Mar 2011

End of Annex

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