

(51) International Patent Classification:
G06F 13/14 (2006.01)(21) International Application Number:
PCT/US2014/047707(22) International Filing Date:
22 July 2014 (22.07.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, LP** [US/US]; 11445 Compaq Center Drive W, Houston, Texas 77070 (US).(72) Inventors: **LI, Sheng**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **FARABOSCHI, Paolo**; 3000 Hanover st, Palo Alto, California 94304-1112 (US). **LIM, Kevin T**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **ZHAO, Jishen**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US).(74) Agents: **KUIPERS, Luke** et al.; Hewlett-Packard Company, Intellectual Property Administration, 3404 E. Harmony Road, Mail Stop 35, Fort Collins, Colorado 80528-9599 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, 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, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, 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).

Declarations under Rule 4.17:

— as to the identity of the inventor (Rule 4.17(i))

Published:

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

(54) Title: NODE-BASED COMPUTE DEVICE WITH PROTOCOL-BASED PRIORITY

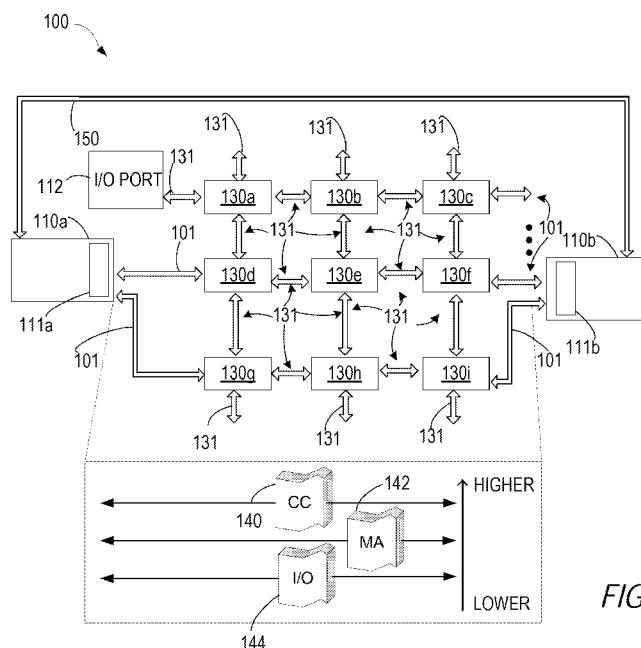


FIG. 1

(57) **Abstract:** According to an example, a node-based compute device includes memory nodes that forms a main memory address space for a processor node. In an example, a memory node from the memory nodes may assign message data a priority based on a protocol type of the message data. The memory node may then select the message data over other message data based on the priority assigned to the message data relative to a priority assigned to the other message data. Responsive to the message data being selected, the memory node may then process the message data.

NODE-BASED COMPUTE DEVICE WITH PROTOCOL-BASED PRIORITY

BACKGROUND

[0001] Computer networks and systems have become indispensable tools for modern business. Today terabytes or more of information on virtually every subject imaginable are stored and accessed across networks. Some applications, such as telecommunication network applications, mobile advertising, social media applications, etc., demand short response times for their data. As a result, new memory-based implementations of programs, such as in-memory databases, are being employed in an effort to provide the desired faster response times. These memory-intensive programs primarily rely on large amounts of directly addressable physical memory (e.g., random access memory) for storing terabytes of data rather than hard drives to reduce response times.

-2-

BRIEF DESCRIPTION OF DRAWINGS

[0002] The following description illustrates various examples with reference to the following figures:

[0003] FIG. 1 is a diagram showing a node-based compute device, according to an example;

[0004] FIG. 2 is a block diagram illustrating modules of a memory node, according to an example;

[0005] FIG. 3 is flowchart illustrating a method for co-scheduling and prioritizing different protocol types transmitted through and by the node-based compute device, according to an example;

[0006] FIG. 4 is a diagram illustrating an example of routing logic that receives message data from given protocol types over dedicated virtual channels;

[0007] FIG. 5 is a diagram illustrating the routing logic being configured to co-schedule different protocol types using a protocol type identifier, according to an example;

[0008] FIG. 6 is a diagram showing an example where message data are promoted to a higher priority than the priority in which the message data were originally assigned; and

[0009] FIG. 7 is a block diagram of a computing device capable of co-scheduling message data, according to one example.

-3-

DETAILED DESCRIPTION

[0010] For simplicity and illustrative purposes, the principles of this disclosure are described by referring mainly to examples thereof. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the examples. It is apparent that the examples may be practiced without limitation to all the specific details. Also, the examples may be used together in various combinations.

[0011] A node-based compute device, according to an example, includes memory nodes that are connected via interconnects, such as point-to-point links. Each memory node may be a memory subsystem including a memory controller and memory to store data. The memory node may also include routing logic to route message data to a destination, which may be another memory node, processor node, or input/output (“I/O”) port in the node-based compute device. The node-based compute device may provide a main memory address space for processor nodes.

[0012] Examples may use the memory nodes and the point-to-point links of a node-based compute device as a messaging fabric to communicate different protocol types carrying different types of data, such as cache coherency messages, memory access command messages, and I/O messages, to given memory nodes, I/O ports, or processor nodes of the node-based compute device.

[0013] In an example discussed herein, a node-based compute device may include a first memory node and a second memory node that together form an address space of main memory for a processor node. In this example, the first memory node may assign message data a priority based on a protocol type of the message data. The first memory node may also select the message data over other message data based on the priority assigned to the message data relative to a priority assigned to the other message data. Responsive to the message data being selected, the first memory node may process the message data. By way of example and not limitation, the first memory node may process the message data

-4-

by transmitting the message data to the second memory node over a point-to-point link if the first memory node is not the destination for the message data.

[0014] FIG. 1 is a diagram showing a node-based compute device 100, according to an example. The node-based compute device 100 may include processor nodes 110a,b and memory nodes 130a-i. The processor nodes 110a,b may be compute units that are configured to execute computer-readable instructions and operate on data stored in the memory nodes 130a-i. As FIG. 1 shows, the processor nodes 110a,b may include processor-side memory controllers 111a,b that are connected (directly or indirectly) to the memory nodes 130a-i of the node-based compute device 100 via point-to-point links 101. A point-to-point link may be a wire or other connection medium that links two circuits. In an example, a point-to-point link connects only two circuits which is unlike a shared bus or crossbar switches that connect more than two circuits or devices. A processor node and processor-side memory controller connected to the node-based compute device 100, such as 110a and 111a or 110b and 111b, may be provided on the same chip, or may be provided on separate chips. Also, more or fewer processor nodes, processor-side memory controllers and memory nodes than shown in FIG. 1 may be used in the node-based compute device 100. Also, an I/O port 112 may be connected to the node-based compute device 100. The I/O port 112 may be linked to a network device, a memory device, a data link or bus, a display terminal, a user input device, or the like.

[0015] The processor nodes 110a,b may, in some cases, be connected to each other with a direct processor node-to-processor node link 150, which may be a point-to-point link that provides a direct communication channel between the processor nodes 110a,b. In some cases, the processor nodes 110a,b may be configured to use the direct processor node-to-processor node link 150 to communicate high priority message data that is destined for each other. For example, the processor node 110a may send a cache coherency message destined

-5-

for the processor node 110b through the direct processor node-to-processor node link 150 to avoid multiple hops through the memory nodes 130a-i.

[0016] The node-based compute device 100 may include memory nodes 130a-i that may also be connected together via point-to-point links 131, which are inter-node point-to-point links. Each memory node can operate as a destination of message data if the data to be accessed is stored at the memory node, and as a router that forwards message data to its appropriate destination memory node if the data to be accessed is at a different memory node. For example, the processor-side memory controllers 111a,b can send memory access command messages, e.g., read, write, copy, etc., to the memory nodes 130a-i to perform memory access operations for the processor nodes 110a,b. Each memory node receiving message data may execute the command if that memory node is the destination or route the command to its destination memory node. The node-based compute device 100 may provide memory scalability through the point-to-point links 131 and through the ability to add memory nodes as needed, which may satisfy the memory capacity requirements of big-data workloads. Scaling up memory capacity in the node-based compute device 100 may involve, in some cases, cascading additional memory nodes.

[0017] The node-based compute device 100 may be used to co-schedule and prioritize protocols of different types transmitted through and by the node-based compute device 100 to provide quality of service ("QoS") provisions for messages communicated through the node-based compute device 100. For example, the node-based compute device 100 may be used to co-schedule and prioritize, among other things, cache coherency messages 140, memory access command messages 142, and I/O commands 144. In some cases, the cache coherency messages 140 may have a level of priority, the memory access messages 142 may have a lower level of priority relative to the cache coherency messages 140, and I/O commands 144 may have a lower level of priority relative to the memory access messages 142.

-6-

[0018] FIG. 2 is a block diagram illustrating modules of a memory node, according to an example. It is to be appreciated that although FIG. 2 is discussed relative to the memory node 130e, examples of the other memory nodes 130a-d,f-l may also include the modules/logic discussed with reference to FIG. 2. As shown in FIG. 2, the memory node 130e may include a memory-side memory controller 210 and local memory 215. The local memory 215 may be dynamic random-access memory (DRAM) or any type of semiconductor memory that is volatile or nonvolatile.

[0019] The memory-side memory controller 210 may include memory logic 211 and routing logic 212. The memory-side memory controller 210 may include hardware and/or machine readable instructions stored on a storage device and executable by hardware. The memory logic 211 and the routing logic 212 may be implemented by separate hardware circuits or a single circuit. The memory logic 211 may perform the operations involved in executing memory access operations on the local memory 215. For example, the memory logic 211 can receive packets from other memory nodes, decode the packets to extract the memory access commands and enforce memory management mechanisms that may be implemented and the actual execution of the read, write, and block copy commands from local memory. For example, with temporary reference to FIG. 1, after receiving a read command from the processor-side memory controller 111a of the processor node 110a, the memory logic 211 can fetch the data from local memory 215 and notify the processor-side memory controller 111a that the data is ready to be accessed or directly sends a data packet with a transaction identifier and the requested data back to the processor-side memory controller 111a. These mechanisms depend on the specific type of the memory technology employed in the node; for example, the memory-side co-memory controller for DRAM is different from the co-memory controller for a DRAM stack, for flash memory, or for other forms of non-volatile memory.

-7-

[0020] The routing logic 212 may receive message data, determine whether the message data relate to a memory address mapped to the local memory 215, and, if yes, sends the message data to the memory logic 211 for execution. If the memory node 130e is not the destination, the routing logic 212 sends the message data to a next hop in the node-based compute device 100 toward the destination along one of the point-to-point links 131.

[0021] To manage the incoming message data, the routing logic 212 shown in FIG. 2 may include a message scheduler 202, message buffers 203a-c, and a message dispatcher 204. The message scheduler 202 may be a computer-implemented module configured to receive message data over the point-to-point link 131 and schedule message data for transmission to another memory node within a node-based compute device. In some cases, the message scheduler 202 may schedule message data for processing by storing message data received over the point-to-point link 131 in a selected message buffer according to the virtual channel in which the message data is received or according to a protocol type identifier tagged in the message data.

[0022] Each of the message buffers 203a-c may be computer readable memory that provides the message scheduler 202 and the message dispatcher 204 the capability of storing and reading message data. In some cases, the message buffers 203a-c may be implemented as circular queues or arrays that allow the message scheduler 202 to push message data to the back of one of the selected message buffers and to pop message data from the front of one of the selected message buffers.

[0023] The message dispatcher 204 may be a computer-implemented module configured to process message data stored in the message buffers 203a-c. The message dispatcher 204 may process message data by forwarding the message data to another memory node if the message data is not destined for the memory node 130e or sending the message data to the memory logic 211 if the message data is destined for the memory node 130e.

-8-

[0024] The order in which the message dispatcher 204 selects message data among the message buffers 203a-c may in some cases be determined based on priorities assigned to the message buffers 203a-c. For example, in some cases, the message buffer 203a may be assigned a first priority, the message buffer 203b may be assigned a second priority, and the message buffer 203c may be assigned a third priority. In such a case, in each round in which the message dispatcher 204 selects message data to process, the message dispatcher 204 may process the message data from the message buffer 203a if the message buffer 203a is non-empty. If the message buffer 203a is empty, the message dispatcher 204 will then process message data from the message buffer 203b. If message buffers 203a,b are both empty, the message dispatcher 204 will the process the message data from the message buffer 203c.

[0025] The memory-side memory controller 210 may be used to co-schedule and prioritize different types of communication transmitted through and by the node-based compute device to provide QoS provisions for messages communicated through the node-based compute device. FIG. 3 is flowchart illustrating a method 300 for co-scheduling and prioritizing different protocol types communicating data transmitted through and by the node-based compute device, according to an example. The method 300 may be performed by the modules, logic, components, or systems shown in FIGS. 1 and 2 and, accordingly, is described herein merely by way of reference thereto. It will be appreciated that the method 300 may, however, be performed on any suitable hardware.

[0026] The method 300 may begin at operation 302 when the message scheduler 202 of the memory node 130e receives message data. In some cases, the message data may be sent to the memory node 130e from an adjacent memory node in the node-based compute device 100. For example, one of the memory nodes 130b, 130d, 130f, or 130h may send the message data to the memory node 130e through a connecting point-to-point link 131. Further, the message data may

-9-

have originated from any of the memory nodes, processor nodes 110a,b, or I/O devices.

[0027] At operation 304, upon receiving the message data, the message scheduler 202 may assign the message data a priority based on a protocol type of the message data. For example, if the protocol type of the message data relates to cache coherency messages then the message scheduler 202 may assign the message data one priority, or if the protocol type of the message data relates to memory access command messages then the message scheduler 202 may assign the message data another priority, or if the protocol type of the message data relates to I/O messages then the message scheduler 202 may assign the message data yet another priority. Example techniques for assigning message data are discussed in greater below.

[0028] At operation 306, the message dispatcher 204 may select the message data that was received at operation 302 over other message data based on the priority assigned to the message data relative to a priority assigned to the other message data. For example, the message dispatcher 204 may select the message data that was received at operation 302 over the other message data because the message data that was received at operation 302 was assigned a priority relating to cache coherency messages, while the other message data may have been assigned a priority relating to an I/O message.

[0029] At operation 308, the message dispatcher 204 may, responsive to the message data being selected, process the message data. In some cases, processing the message data may involve transmitting the message data to the second memory node through one of the point-to-point links 131, as may occur upon determining that the memory node 130e is not the destination of the message data. In other cases, processing the message data may involve sending the message data to the memory logic 211 of the memory node 130e, so that the memory logic 211 can perform a memory operation on the local memory 215.

-10-

[0030] In some cases, a memory-side memory controller may use protocol separation to co-schedule and prioritize the different types of protocols carrying message data transmitted through and by the node-based compute device 100. An example of a way in which a memory-side controller may provide protocol separation is by dedicating virtual channels to different protocol types of communication data. FIG. 4 is a diagram illustrating an example of routing logic 212 that receives message data of given protocol types over dedicated virtual channels. A virtual channel may be a logical partitioning of the physical bandwidth of the point-to-point link 131. By way of illustration and not limitation, the point-to-point link 131 is transmitting message data through virtual channel segments 410-413. A virtual channel segment may refer to data sent over a point-to-point link that relates to a given virtual channel. Virtual channel segments 410, 412 may relate to the same virtual channel, for example, virtual channel 1. Virtual channel 1 may be used to transmit one type of protocol, such as cache coherency messages. The virtual channel segment 411 may relate to a virtual channel different from the virtual channel of the virtual channel segments 410, 412, such as, for example, virtual channel 2. Virtual channel 2 may be used to transmit another type of protocol, such as memory access command messages. The virtual channel segment 413 may relate to yet another virtual channel, for example, virtual channel 3. Virtual channel 3 may be used to transmit yet another type of protocol, such as commands to I/O devices.

[0031] Each of the virtual channel segments 410-413 may include virtual channel header data and message data. Virtual channel header data may be header data that identifies a virtual channel and allows for control of the flow of data sent through the corresponding virtual channel. As an illustration of virtual channel header data and message data that may be sent in a virtual channel, FIG. 4 shows that the virtual channel segment 410 may include a virtual channel identifier 420, message data 421, and an end of transmission tag 422. The virtual channel identifier 420 may be data that signal transmission is beginning over a specified

-11-

virtual channel (e.g., virtual channel 1). That is, header data and message data that follows the virtual channel identifier 420 belong to the virtual channel identified by the virtual channel identifier 420. The message data 421 may be the data for a cache coherency message. For example, in a cache coherency message, the message data 421 may include data specifying a memory address and/or a memory value. The end of transmission tag 422 may signal the transmission of data involving a virtual channel has ended. In some cases, rather than using an explicit end-of-transmission data message, sending another virtual channel identifier data message may be used to implicitly signal the end of one virtual channel.

[0032] The virtual channel segments 411-413 may include a similar format of data to that used by virtual channel segment 410. For example, the virtual channel segment 411 may include a virtual channel identifier 440 that signals transmission is beginning over a specified virtual channel (e.g., virtual channel 2), message data 441 that includes data specifying a memory access command, and an end of transmission tag 442 that signals that transmission of data for the virtual channel specified by the virtual channel identifier 440 has ended. Although not shown in FIG. 4, virtual channel segments 412 and 413 may include similar formats. That is, the virtual channel segments 412, 413 may each include a virtual channel identifier, message data, and an end of transmission tag.

[0033] The message data transmitted by the different virtual channels may vary in data size. For example, the virtual channel segment 411 may include message data 441 that is comparatively larger than the message data 421 transmitted in the virtual channel segment 410. Such may be the case where message data includes a data payload that is to be written in a local memory of a memory node or communicating blocks of memory through an input/output interface.

[0034] As described above, some examples of the message scheduler 202 may schedule message data for processing by storing message data received over

-12-

the point-to-point link 131 in a selected message buffer according to the virtual channel in which the message data is received. For example, the message buffers 203a-c may each be assigned to a given virtual channel (e.g., the message buffer 203a may be assigned to virtual channel 1, the message buffer 203b may be assigned to virtual channel 2, and the message buffer 203c may be assigned to virtual channel 3). Accordingly, when receiving the virtual channel segment 410, the message scheduler 202 may store the message data 421 in the message buffer 203a based on the message buffer 203a being assigned to virtual channel 1 and the virtual channel identifier 420 specifying that the virtual channel segment 410 belongs to virtual channel 1. Similarly, the message scheduler 202 may store the message data 441 in the message buffer 203b based on the message buffer 203b being assigned to virtual channel 2 and the virtual channel identifier 440 specifying that the virtual channel segment 411 belongs to virtual channel 2.

[0035] As FIG. 4 illustrates, assigning message data is performed implicitly, as each virtual channel is assigned or otherwise dedicated for a protocol type, and each virtual channel has a priority rank. Thus, a memory node does not need to reassign priority to message data once the memory node receives the message data a given virtual channel. Instead, the memory node can just schedule the routing operations based on the virtual channel priority ranking.

[0036] Although FIG. 4 illustrates examples where virtual channels are dedicated to a single type of protocol, other examples may transmit multiple types of protocols over the same virtual channel. FIG. 5 is a diagram illustrating the routing logic 212 being configured to co-schedule different types of protocols using a protocol type identifier, according to an example. In FIG. 5, the point-to-point link 131 transmits virtual channel segments 510, 511. The virtual channel segments 510, 511 may transmit message data for multiple protocol types. For example, as shown in FIG. 5, the virtual channel segment 510 may include a virtual channel identifier 520, message data packets 521, 522, and an end of transmission tag 523.

-13-

The roles of a virtual channel identifier and an end of transmission tag are discussed above with respect to FIG. 4.

[0037] The message data packets 521, 522 may each carry data relating to the same or different message data. By way of example and not limitation, the message data packet 521 may include data 521b for a cache coherency message and the message data packet 522 may include data 522b for a memory access command (e.g., a read or write of a memory address). To identify the protocol type of the message data packets 521, 522, the message data packets 521, 522 may include protocol type identifiers 521a, 522a. A protocol type identifier may be a field in a message data packet that signals a protocol type for the message data packet. To illustrate, with reference to FIG. 5, the protocol type identifier 521a may specify that the message data packet 521 is a cache coherency message and the protocol type identifier 522a may specify that the message data packet 522 is a memory access command. In some cases, the protocol type identifier may be part of a protocol header transmitted with the message data packets. A protocol header may include other information usable to control and manage the flow of the message data packets.

[0038] Upon receiving a message data packet, the message scheduler 202 may assign a priority to the message data packet by decoding the message data packet to identify a protocol type associated with the message data packet and then storing the message data packet in a message buffer mapped to the identified protocol type. To illustrate, upon receiving the message data packet 521, the message scheduler 202 may decode the message data packet 521 to identify a protocol type (e.g., cache coherency) specified by the protocol type identifier 521a. Based on the protocol type specified by the protocol type identifier 521a, the message scheduler 202 may store the message data packet 521 in a message buffer mapped to that protocol type. If, for example, the protocol type identifier 521a specifies a cache coherency type and if the message buffer 203a is assigned to cache coherency type messages, the message scheduler 202, upon making

-14-

these determinations, will store the message data packet 521 in the message buffer 203a. The same process may be used for scheduling memory packet 522. That is, upon receiving the message data packet 522, the message scheduler 202 may decode the message data packet 522 to identify a protocol type (e.g., memory access) specified by the protocol type identifier 522a. Based on the protocol type specified by the protocol type identifier 522a, the message scheduler 202 may store the message data packet 522 in a message buffer assigned to that protocol type. If, for example, the protocol type identifier 522a specifies a memory access command messages type and if the message buffer 203b is assigned to memory access command messages, the message scheduler 202, upon making these determinations, may store the message data packet 522 in the message buffer 203b.

[0039] Some examples may provide techniques to reduce starvation of scheduling message data of lower priority. "Starvation," as used herein, may refer to a situation where routing logic fails to schedule message data for processing because the routing logic repeatedly selects other message data to process because those other message data are assigned to a comparatively higher priority. One technique for reducing starvation of a message data is to promote the message data that is starved for scheduling. Promoting a message data may refer to a mechanism in which message data is selected for transmission contrary to a priority originally assigned to the message data.

[0040] FIG. 6 is a diagram showing an example where message data are promoted to a higher priority than the priority in which the message data were originally assigned. In FIG. 6, the message buffer 203a includes message data 602, 604, which may relate to cache coherency messages; the message buffer 203b includes message data 606, which may relate to a memory access command; and the message buffer 203c includes message data 608, which may relate to I/O messages. FIG. 6 also shows that the message scheduler 202 is receiving a stream of cache coherency message data 612.

-15-

[0041] As discussed above, the message dispatcher 204 may operate by scheduling message data for processing based on the priority of the message buffers storing the message data. For example, the message dispatcher 204 may schedule message data from the message buffer 603a until the message buffer 603a is empty. If the message buffer 603a is empty, the message dispatcher 204 may schedule message data stored in message buffer 603b for processing until the message buffer 603b is empty or until the message buffer 603a stores new message data. If the message buffers 603a,b are empty, the message dispatcher 204 may schedule message data stored in message buffer 603c for processing until the message buffer 603c is empty or until the message buffers 603a,b store new message data.

[0042] In some cases, it may take an unfair amount of time to empty a message buffer assigned to a comparatively high priority. Such may be the case when the rate in which a stream of message data assigned to a high priority (e.g., the stream of message data 612) exceeds the rate in which the message dispatcher 204 can process the message data stored in the message buffer associated with that high priority (e.g., the message data 602, 604 stored in message buffer 203a). In this case, or if this case persists for an extended period of time, message data stored in the lower priority message buffers (e.g., message buffers 203b,c) may be viewed as a starved resource.

[0043] To provide fair scheduling, or to limit resource starvation, of the message data 606, 608, the message dispatcher 204 may promote the message data 606, 608. In selecting which message data to promote, the message scheduler 202 may determine if the age (e.g., a time period in which the message data has been queued in a message buffer) of a message data exceeds an age threshold. In some examples, the message dispatcher 204 may be configured to promote a message data by re-queuing or moving the message data to a message buffer that is mapped to a higher priority. FIG. 6 shows that the message data 606

-16-

may be promoted to the message buffer 203a, while the message data 608 may be promoted to the message buffer 203b.

[0044] Other examples may promote message data using techniques other than re-queuing message data in higher priority message buffers. For example, the message scheduler 202 may periodically identify and mark message data as stale if the message data has been queued in a message buffer for a time period above a threshold time period. In selecting message data for processing, the message dispatcher 204 may first select message data that is marked as stale before processing message data based on priorities of the message buffers 203a-c. As another example, the message scheduler 202 can increase the priority assigned to a message buffer (203b) with stale message data so that that message buffer (e.g., 203b) has a higher priority relative to another message buffer (e.g. 203a) for a period of time

[0045] FIG. 7 is a block diagram of a computing device 700 capable of co-scheduling message data, according to one example. The computing device 700 includes, for example, a processor 710, and a computer-readable storage device 720 including instructions 722, 724. The computing device 700 may be, for example, a memory node (see FIG. 1) or any other suitable computing device capable of providing the functionality described herein.

[0046] The processor 710 may be a central processing unit (CPU), a semiconductor-based microprocessor, a graphics processing unit (GPU), other hardware devices or circuitry suitable for retrieval and execution of instructions stored in computer-readable storage device 720, or combinations thereof. For example, the processor 710 may include multiple cores on a chip, include multiple cores across multiple chips, multiple cores across multiple devices, or combinations thereof. The processor 710 may fetch, decode, and execute one or more of the instructions 722, 724 to implement methods and operations discussed above, with reference to FIGS. 1-6. As an alternative or in addition to retrieving and executing instructions, processor 710 may include at least one integrated circuit (IC), other

-17-

control logic, other electronic circuits, or combinations thereof that include a number of electronic components for performing the functionality of instructions 722, 724.

[0047] Computer-readable storage device 720 may be any electronic, magnetic, optical, or other physical storage device that contains or stores executable instructions. Thus, computer-readable storage device may be, for example, Random Access Memory (RAM), an Electrically Erasable Programmable Read-Only Memory (EEPROM), a storage drive, a Compact Disc Read Only Memory (CD-ROM), non-volatile memory, and the like. As such, the machine-readable storage device can be non-transitory. As described in detail herein, computer-readable storage device 720 may be encoded with a series of executable instructions for co-scheduling message data through a node-based compute device.

[0048] As used herein, the term “computer system” may refer to one or more computer devices, such as the computer device 700 shown in FIG. 7. Further, the terms “couple,” “couples,” “communicatively couple,” or “communicatively coupled” is intended to mean either an indirect or direct connection. Thus, if a first device, module, or engine couples to a second device, module, or engine, that connection may be through a direct connection, or through an indirect connection via other devices, modules, logic, engines and connections. In the case of electrical connections, such coupling may be direct, indirect, through an optical connection, or through a wireless electrical connection.

[0049] While this disclosure makes reference to some examples, various modifications to the described examples may be made without departing from the scope of the claimed features.

-18-

CLAIMS

What is claimed is:

1. A node-based compute device comprising:
 - a first memory node connected to a second memory node through a point-to-point link, the first memory node includes:
 - a message scheduler to assign message data a priority based on a protocol type of the message data, and
 - a message dispatcher to:
 - select the message data over other message data based on the priority assigned to the message data relative to a priority assigned to the other message data, and
 - responsive to the message data being selected, processing the message data.
2. The node-based compute device of claim 1, wherein the message data includes protocol header data that specifies that the message data is of the protocol type, and the message scheduler is to determine the protocol type based on an inspection of the protocol header.
3. The node-based compute device of claim 2, wherein the message scheduler is to:
 - select a first message buffer based on the priority assigned to the message data; and
 - store the message data in the first message buffer.

-19-

4. The node-based compute device of claim 3, wherein the message scheduler is to:

determine that a second message buffer has an available slot, the second message buffer being assigned to a different priority; and

move the message data to the second message buffer based on the available slot.

5. The node-based compute device of claim 3, wherein the message scheduler is to:

determine that an age of the message data exceeds an age threshold; and

increase the priority assigned to the first message buffer so that the first message buffer has a higher priority relative to a second message buffer for a period of time.

6. The node-based compute device of claim 1, wherein the protocol type includes at least one of: a cache coherency protocol, a memory access protocol, or an input/output protocol.

7. The node-based compute device of claim 1, wherein the protocol type of the message data is used to communicate cache coherency messages, and a protocol type of another message data is used to communicate memory access command messages.

8. The node-based compute device of claim 1, wherein:

-20-

the message scheduler is further to assign an additional message data an additional priority based on a protocol type of the additional message data, and

the message dispatcher is further to abstain from communicating the additional message data to the second memory node until communication of the message data completes.

9. The node-based compute device of claim 1, wherein the message data is received through a virtual channel, and the message scheduler is to assign the message data the priority by storing the message data in a message buffer assigned to the virtual channel, the message buffer being assigned the priority according to a priority ranking of the message buffer.

10. The node-based compute device of claim 1, wherein the message dispatcher is to process the message data by determining that the first memory node is a destination for the message data, and forwarding the message data to a memory logic of the first memory node.

11. A method comprising:

assigning, by a first memory node, message data received from a second memory node a priority based on a protocol type of the message data, the first memory node and the second memory node forming a main memory address space for a processor node;

selecting the message data over other message data based on the priority assigned to the message data relative to a priority assigned to the other message data, and

-21-

responsive to the message data being selected, transmitting the message data to the second memory node over a point-to-point link.

12. The method of claim 11, wherein the message data is received through a virtual channel assigned to a priority, and the message scheduler is to assign the message data the priority based on the message data being received through the virtual channel.

13. The method of claim 11, wherein the protocol type of the message data is used to communicate cache coherency messages, and a protocol type of the another message data is used to communicate memory access command messages.

14. The method of claim 11, further comprising, prior to selecting the message data over other message data:

determining that an age of the message data exceeds an age threshold; and

promoting the priority of the message data.

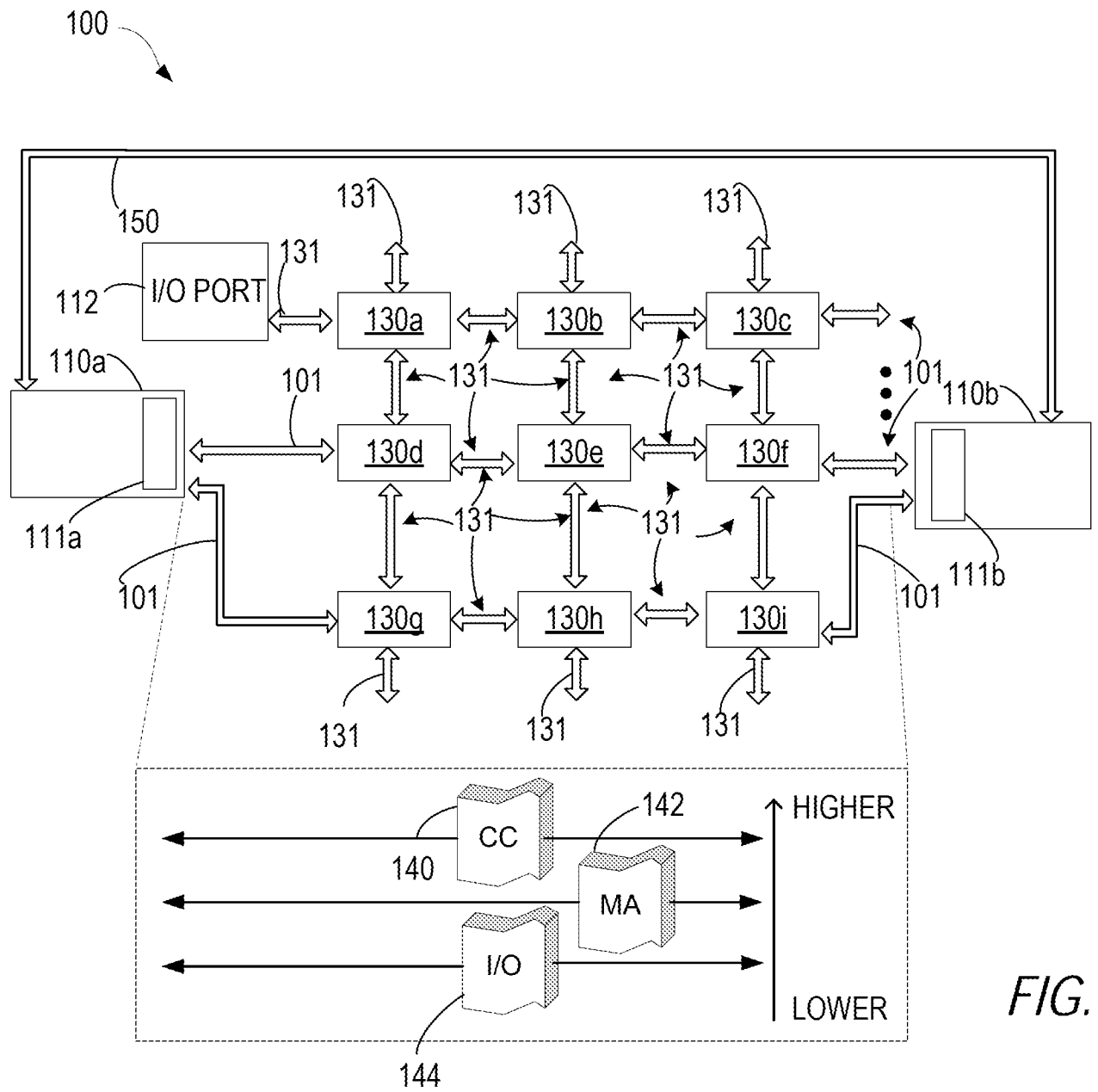
15. A computer-readable storage device comprising instructions that, when executed, cause a processor node of a computing device to:

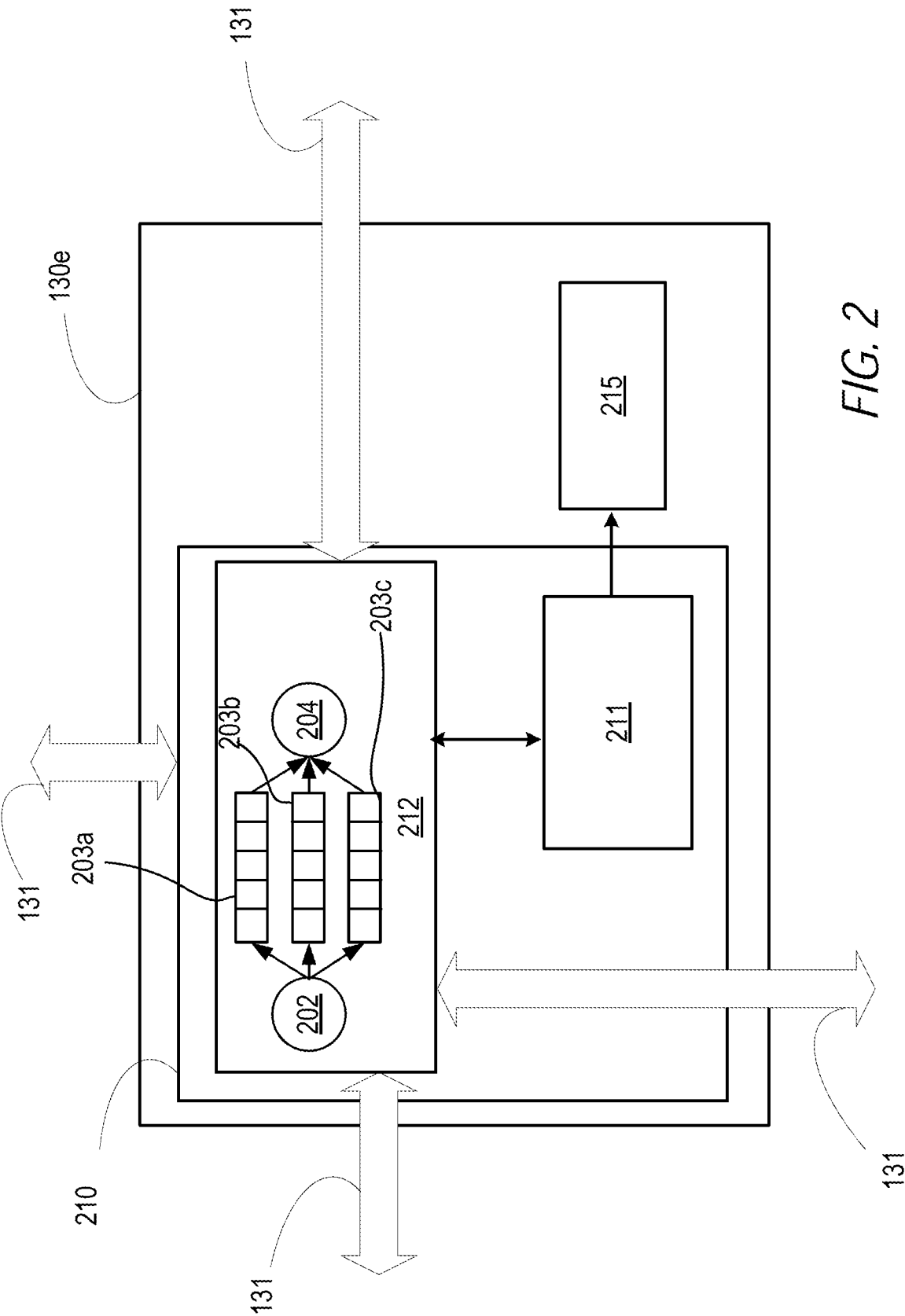
assign, by a first memory node, message data received from a second memory node a priority based on a protocol type of the message data, the first memory node and the second memory node forming a main memory address space for a processor node;

-22-

select the message data over other message data based on the priority assigned to the message data relative to a priority assigned to the other message data, and

responsive to the message data being selected, transmit the message data to the second memory node over a point-to-point link.





3/7

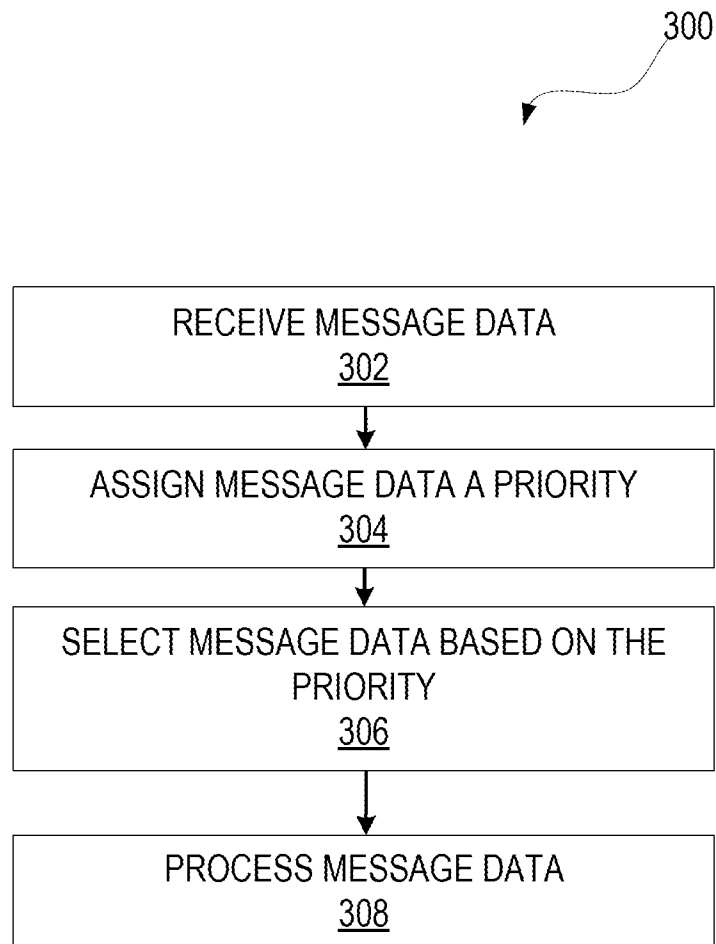


FIG. 3

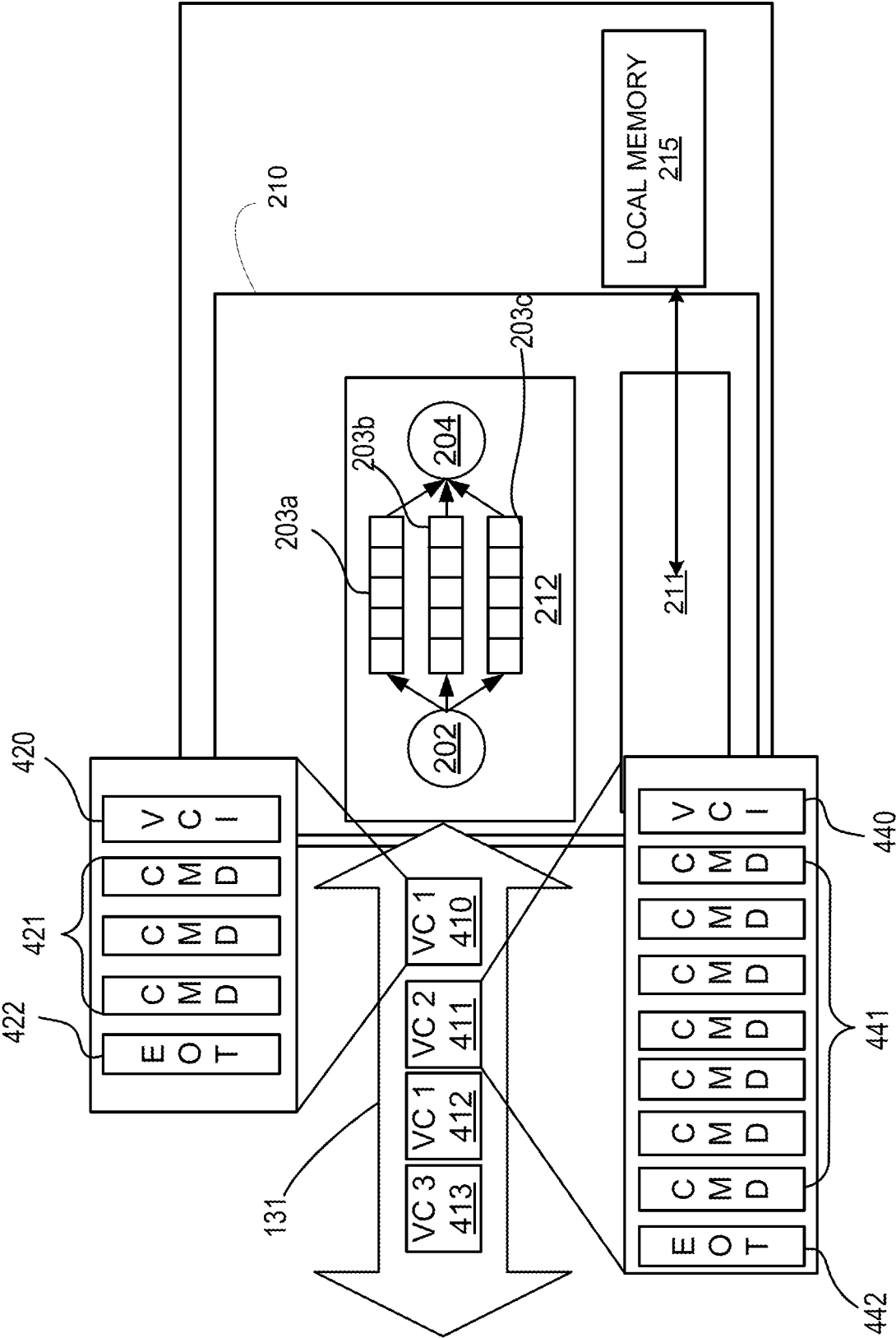


FIG. 4

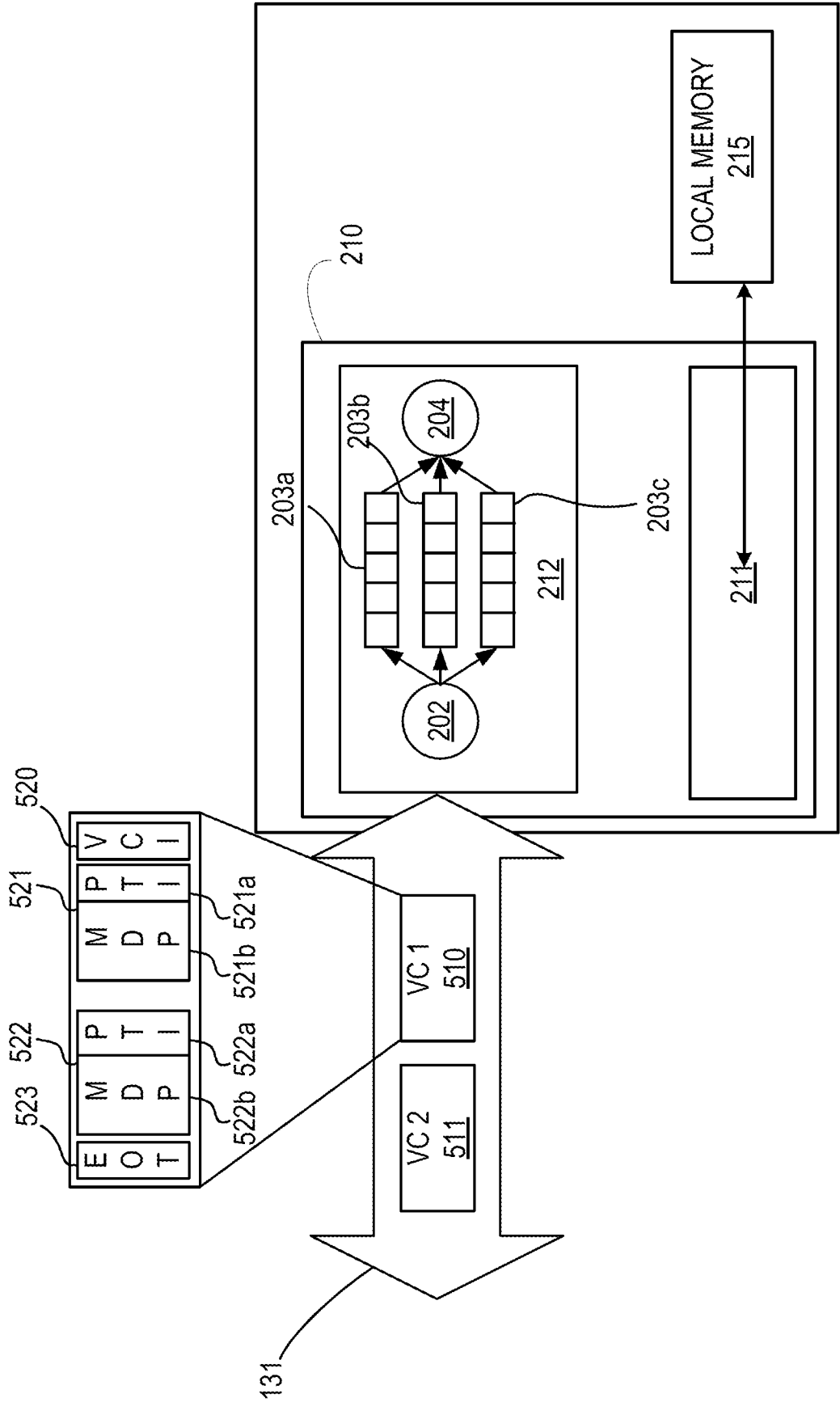


FIG. 5

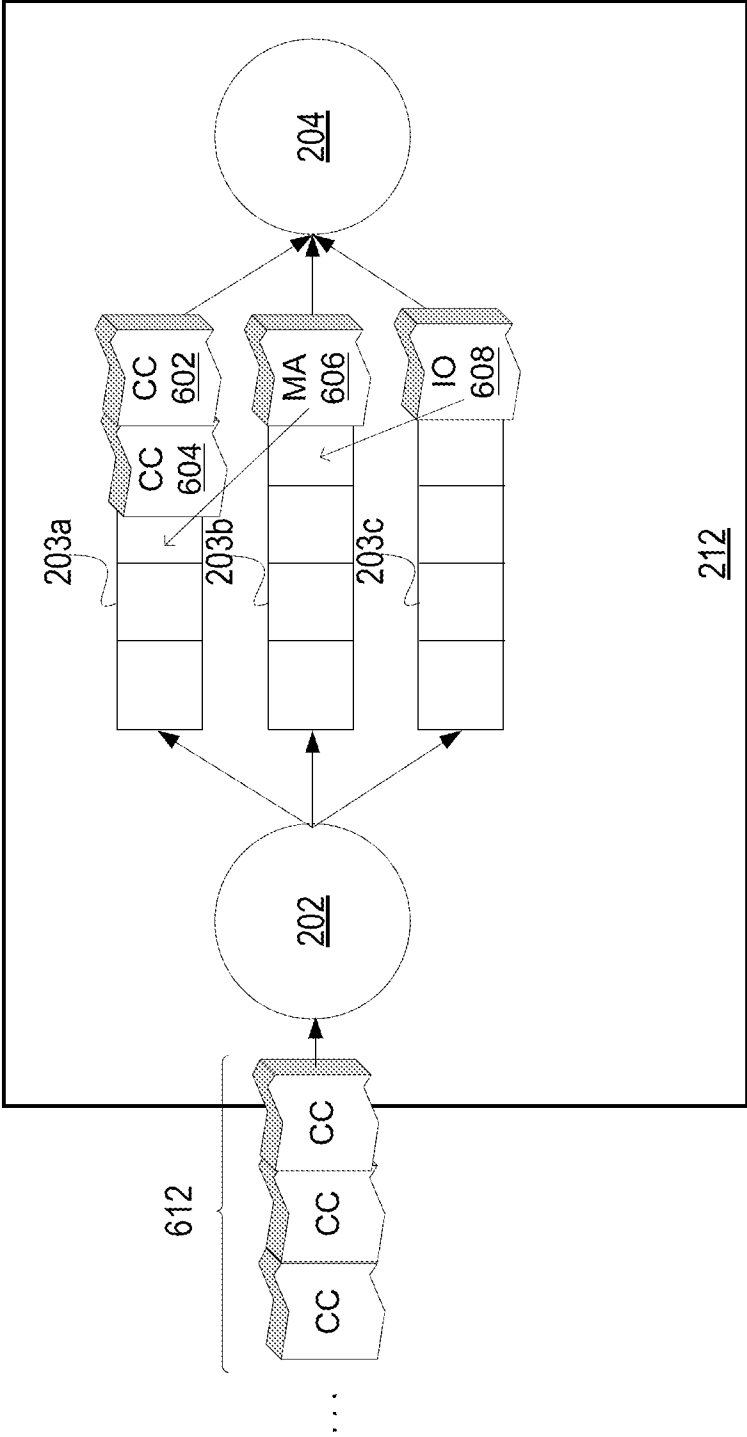


FIG. 6

7/7

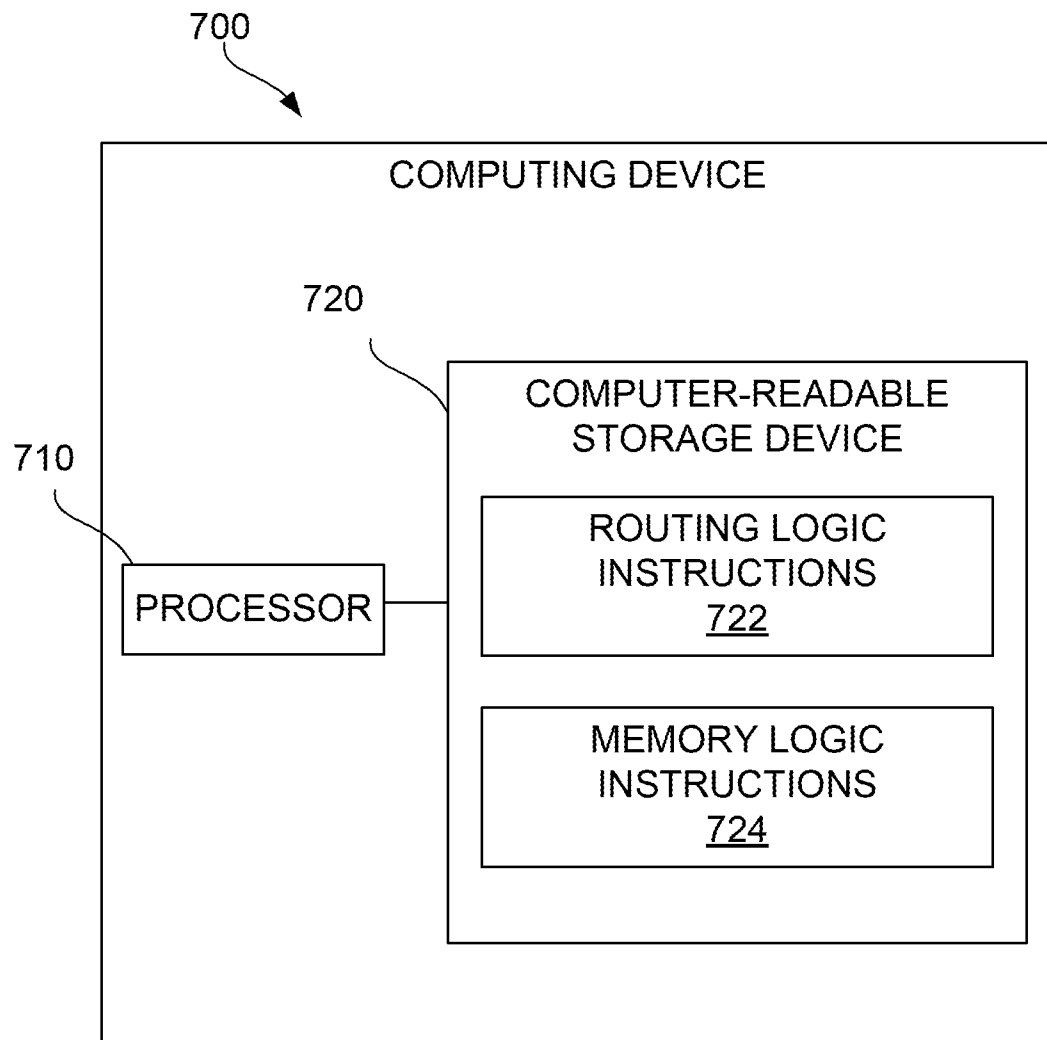


FIG. 7

A. CLASSIFICATION OF SUBJECT MATTER**G06F 13/14(2006.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 13/14; G11C 8/00; H04L 29/06; G06F 12/00; G06F 13/16; H04M 7/00; H04L 12/64; H04L 12/56

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: point-to-point link, memory node, priority, message, protocol type, and similar terms

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	EP 2234021 A1 (HONEYWELL INTERNATIONAL INC.) 29 September 2010 See paragraphs [0005], [0023]–[0025], and [0036]; claims 1 and 4; and figures 1–4.	1–3, 6–7, 9–13, 15
A		4–5, 8, 14
Y	US 2013–0294243 A1 (WILLIAM L. WILEY et al.) 07 November 2013 See paragraphs [0026], [0127], [0201], [0268], and [0330]; and claim 1.	1–3, 6–7, 9–13, 15
A	WO 2012–178117 A2 (CYGNUS BROADBAND, INC.) 27 December 2012 See paragraphs [009] and [060]–[077].	1–15
A	US 2003–0061447 A1 (RICHARD E. PEREGO et al.) 27 March 2003 See paragraphs [0034]–[0035].	1–15
A	US 2002–0034118 A1 (PETER POCHMULLER) 21 March 2002 See paragraphs [0003] and [0051]–[0059]; and figure 2.	1–15
A	WO 00–41368 A2 (NORTEL NETWORKS CORPORATION) 13 July 2000 See page 2, lines 7–13; page 3, line 23 – page 7, line 15; and figure 1.	1–15



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

27 March 2015 (27.03.2015)

Date of mailing of the international search report

27 March 2015 (27.03.2015)

Name and mailing address of the ISA/KR

International Application Division
Korean Intellectual Property Office
189 Cheongsu-ro, Seo-gu, Daejeon Metropolitan City, 302-701,
Republic of Korea

Facsimile No. ++82 42 472 7140

Authorized officer

NHO, Ji Myong

Telephone No. +82-42-481-8528



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2014/047707

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 2234021 A1	29/09/2010	JP 2010-262631 A US 2010-0241783 A1	18/11/2010 23/09/2010
US 2013-0294243 A1	07/11/2013	CA 2656409 A1 CA 2656412 A1 CA 2656552 A1 IL 196283 A IL 196283 D0 IL 196284 A IL 196284 D0 IL 196285 D0 US 2008-002676 A1 US 2008-002716 A1 US 2008-005156 A1 US 2012-201139 A1 US 2014-043977 A1 US 7765294 B2 US 8184549 B2 US 8477614 B2 US 8570872 B2 WO 2008-097254 A1	10/01/2008 13/03/2008 14/08/2008 30/09/2013 22/09/2009 30/09/2014 22/09/2009 22/09/2009 03/01/2008 03/01/2008 03/01/2008 09/08/2012 13/02/2014 27/07/2010 22/05/2012 02/07/2013 29/10/2013 14/08/2008
WO 2012-178117 A2	27/12/2012	CA 2840048 A1 CN 103650440 A EP 2724490 A2 KR 10-1489414 B1 KR 10-2014-0040798 A US 2012-0008499 A1 US 2012-0140633 A1 US 2012-0327778 A1 US 2014-0126364 A1 US 8665724 B2 WO 2012-178117 A3	27/12/2012 19/03/2014 30/04/2014 04/02/2015 03/04/2014 12/01/2012 07/06/2012 27/12/2012 08/05/2014 04/03/2014 11/04/2013
US 2003-0061447 A1	27/03/2003	EP 1587112 A2 EP 1587112 A3 EP 1587112 B1 EP 1890294 A2 EP 1890294 A3 US 2004-0186956 A1 US 2004-0256638 A1 US 2005-0007805 A1 US 2005-0010737 A1 US 2005-0041504 A1 US 2005-0044303 A1 US 2005-0149662 A1 US 2005-0156934 A1 US 2005-0166026 A1 US 2005-0193163 A1	19/10/2005 21/06/2006 20/08/2008 20/02/2008 30/04/2008 23/09/2004 23/12/2004 13/01/2005 13/01/2005 24/02/2005 24/02/2005 07/07/2005 21/07/2005 28/07/2005 01/09/2005

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2014/047707

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		US 2005-0207255 A1	22/09/2005
		US 2005-0210196 A1	22/09/2005
		US 2005-0223179 A1	06/10/2005
		US 2006-0067141 A1	30/03/2006
		US 2008-0034130 A1	07/02/2008
		US 2008-0109596 A1	08/05/2008
		US 2009-0319719 A1	24/12/2009
		US 6502161 B1	31/12/2002
		US 6832284 B1	14/12/2004
		US 7000062 B2	14/02/2006
		US 7003618 B2	21/02/2006
		US 7010642 B2	07/03/2006
		US 7017002 B2	21/03/2006
		US 7051151 B2	23/05/2006
		US 7062597 B2	13/06/2006
		US 7200710 B2	03/04/2007
		US 7206895 B2	17/04/2007
		US 7206896 B2	17/04/2007
		US 7206897 B2	17/04/2007
		US 7266634 B2	04/09/2007
		US 7320047 B2	15/01/2008
		US 7356639 B2	08/04/2008
		US 7363422 B2	22/04/2008
		US 7404032 B2	22/07/2008
		US 7523248 B2	21/04/2009
		US 7526597 B2	28/04/2009
		WO 2005-116838 A1	08/12/2005
US 2002-0034118 A1	21/03/2002	DE 10046578 A1	04/04/2002
		DE 10046578 B4	22/11/2007
		EP 1191537 A1	27/03/2002
		TW 544573 B	01/08/2003
		US 6542430 B2	01/04/2003
WO 00-41368 A2	13/07/2000	AU 2000-22249 A1	24/07/2000
		CA 2358525 A1	13/07/2000
		CA 2358525 C	01/04/2008
		DE 60024228 D1	29/12/2005
		EP 1142213 A2	10/10/2001
		EP 1142213 B1	23/11/2005
		US 2004-0076161 A1	22/04/2004
		US 7710871 B2	04/05/2010
		WO 00-41368 A3	28/12/2000
		WO 00-41368 A9	04/10/2001