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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2012 (29.11.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/162189 A1

(51) International Patent Classification:
G06F 13/14 (2006.01) *G06F 9/06* (2006.01)
G06F 13/38 (2006.01)

(21) International Application Number:
PCT/US2012/038713

(22) International Filing Date:
18 May 2012 (18.05.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/488,683 20 May 2011 (20.05.2011) US

(71) Applicant *(for all designated States except US)*: **SOFT MACHINES, INC.** [US/US]; 3211 Scott Boulevard, Suite 202, Santa Clara, CA 95054 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **ABDALLAH, Mohammad** [US/US]; 3868 Suncrest Avenue, San Jose, CA 95132 (US).

(74) Agent: **BARNES, Glenn D.**; Murabito Hao & Barnes LLP, Two North Market Street, Third Floor, San Jose, CA 95113 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

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

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) **Title**: AN INTERCONNECT STRUCTURE TO SUPPORT THE EXECUTION OF INSTRUCTION SEQUENCES BY A PLURALITY OF ENGINES

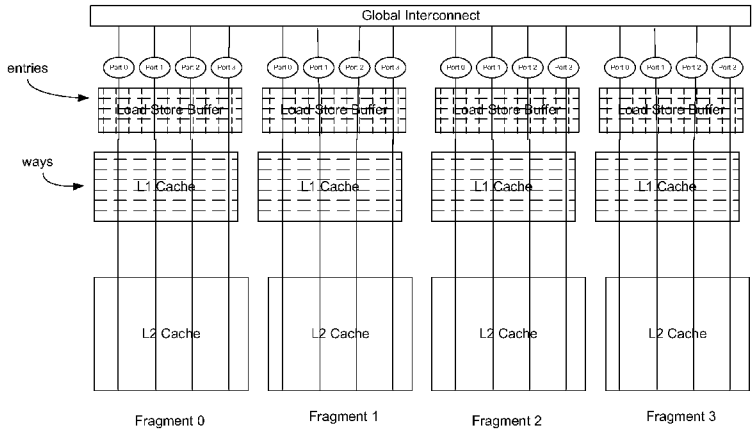

FIG. 4

(57) **Abstract**: A global interconnect system. The global interconnect system includes a plurality of resources having data for supporting the execution of multiple code sequences and a plurality of engines for implementing the execution of the multiple code sequences. A plurality of resource consumers are within each of the plurality of engines. A global interconnect structure is coupled to the plurality of resource consumers and coupled to the plurality of resources to enable data access and execution of the multiple code sequences, wherein the resource consumers access the resources through a per cycle utilization of the global interconnect structure.

WO 2012/162189 A1

AN INTERCONNECT STRUCTURE TO SUPPORT THE EXECUTION OF INSTRUCTION SEQUENCES BY A PLURALITY OF ENGINES

This application claims the benefit co-pending commonly assigned US Provisional Patent Application serial number 61/488683, titled "AN INTERCONNECT STRUCTURE TO SUPPORT THE EXECUTION OF INSTRUCTION SEQUENCES BY A PLURALITY OF ENGINES" by Mohammad A. Abdallah, filed on May 20, 2011, and which is incorporated herein in its entirety.

CROSS REFERENCE TO RELATED APPLICATION

This application is related to co-pending commonly assigned US Patent Application serial number 12/514,303, titled "APPARATUS AND METHOD FOR PROCESSING COMPLEX INSTRUCTION FORMATS IN A MULTITHREADED ARCHITECTURE SUPPORTING VARIOUS CONTEXT SWITCH MODES AND VIRTUALIZATION SCHEMES" by Mohammad A. Abdallah, filed on January 05, 2010, and which is incorporated herein in its entirety.

This application is related to co-pending commonly assigned US Patent Application serial number 12/296,919, titled "APPARATUS AND METHOD FOR PROCESSING AN INSTRUCTION MATRIX SPECIFYING PARALLEL IN DEPENDENT OPERATIONS" by Mohammad A. Abdallah, filed on December 19, 2008, and which is incorporated herein in its entirety.

FIELD OF THE INVENTION

[001] The present invention is generally related to digital computer systems, more particularly, to a system and method for selecting instructions comprising an instruction sequence.

BACKGROUND OF THE INVENTION

[002] Processors are required to handle multiple tasks that are either dependent or totally independent. The internal state of such processors usually consists of registers that might hold different values at each particular instant of program execution. At each instant of program execution, the internal state image is called the architecture state of the processor.

[003] When code execution is switched to run another function (e.g., another thread, process or program), then the state of the machine/processor has to be saved so that the new function can utilize the internal registers to build its new state. Once the new function is terminated then its state can be discarded and the state of the previous context will be restored and execution resumes. Such a switch process is called a context switch and usually includes 10's or hundreds of cycles especially with modern architectures that employ large number of registers (e.g., 64, 128, 256) and/or out of order execution.

[004] In thread-aware hardware architectures, it is normal for the hardware to support multiple context states for a limited number of hardware-supported threads. In this case, the hardware duplicates all architecture state elements for each supported thread. This eliminates the need for context switch when executing a new thread. However, this still has multiple draw backs, namely the area, power and complexity of duplicating all architecture state elements (i.e., registers) for each additional thread supported in hardware. In addition, if the number of software threads exceeds the number of explicitly supported hardware threads, then the context switch must still be performed.

[005] This becomes common as parallelism is needed on a fine granularity basis requiring a large number of threads. The hardware thread-aware architectures with duplicate context-state hardware storage do not help non-threaded software code and only reduces the number of context switches for software that is threaded. However, those threads are usually constructed for coarse grain parallelism, and result in heavy software overhead for initiating and synchronizing, leaving fine grain parallelism, such as function calls and loops parallel execution, without efficient

threading initiations/auto generation. Such described overheads are accompanied with the difficulty of auto parallelization of such codes using sate of the art compiler or user parallelization techniques for non-explicitly/easily parallelized/threaded software codes.

SUMMARY OF THE INVENTION

[006] In one embodiment the present invention is implemented as a global interconnect system. The global interconnect system includes a plurality of resources having data for supporting the execution of multiple code sequences and a plurality of engines for implementing the execution of the multiple code sequences. A plurality of resource consumers are within each of the plurality of engines. A global interconnect structure is coupled to the plurality of resource consumers and coupled to the plurality of resources to enable data access and execution of the multiple code sequences, wherein the resource consumers access the resources through a per cycle utilization of the global interconnect structure.

[007] The foregoing is a summary and thus contains, by necessity, simplifications, generalizations and omissions of detail; consequently, those skilled in the art will appreciate that the summary is illustrative only and is not intended to be in any way limiting. Other aspects, inventive features, and advantages of the present invention, as defined solely by the claims, will become apparent in the non-limiting detailed description set forth below.

BRIEF DESCRIPTION OF THE DRAWINGS

[008] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements.

[009] Figure 1A shows an overview of the manner in which the global front end generates code blocks and inheritance vectors to support the execution of code sequences on their respective engines.

[010] Figure 1B shows an overview diagram of engines and their components, including segmented scheduler and register files, interconnects and a fragmented memory subsystem for a multicore processor in accordance with one embodiment of the present invention.

[011] Figure 2 shows an overview diagram depicting additional features of the interconnect described in the discussion of Figures 1A and 1B, and a plurality of local interconnects in accordance with one embodiment of the present invention.

[012] Figure 3 shows components comprising a resource reservation mechanism that implements efficient access to a contested resource in accordance with one embodiment of the present invention.

[013] Figure 4 shows the interconnect and the ports into the memory fragments in accordance with one embodiment of the present invention.

[014] Figure 5 shows the interconnect and the ports into the segments in accordance with one embodiment of the present invention.

[015] Figure 6 shows a diagram depicting a segmented interconnect in accordance with one embodiment of the present invention.

[016] Figure 7 shows a table that illustrates the manner which requests for segments of the interconnect are contested for and allocated in accordance with one embodiment of the present invention.

[017] Figure 8 shows a table that illustrates the manner in which requests for a point-to-point bus are handled in accordance with one embodiment of the present invention.

[018] Figure 9 shows a diagram of an exemplary logic implementation that implements the functionality of the table of Figure 7 in accordance with one embodiment of the present invention.

[019] Figure 10 shows a diagram of an exemplary logic implementation that implements the functionality of the manner in which requests for a point-to-point bus are handled in accordance with one embodiment of the present invention.

[020] Figure 11 shows a diagram of an interconnect in accordance with one embodiment of the present invention.

[021] Figure 12 shows a table illustrating the manner in which the sender model interconnect structure of Figure 11 functions in accordance with one embodiment of the present invention.

[022] Figure 13 shows a diagram of an exemplary logic implementation that implements the functionality of the manner in which requests for shared bus interconnect structure are handled in accordance with one embodiment of the present invention.

[023] Figure 14 shows a diagram of an exemplary microprocessor pipeline in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[024] Although the present invention has been described in connection with one embodiment, the invention is not intended to be limited to the specific forms set forth herein. On the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the scope of the invention as defined by the appended claims.

[025] In the following detailed description, numerous specific details such as specific method orders, structures, elements, and connections have been set forth. It is to be understood however that these and other specific details need not be utilized to

practice embodiments of the present invention. In other circumstances, well-known structures, elements, or connections have been omitted, or have not been described in particular detail in order to avoid unnecessarily obscuring this description.

[026] References within the specification to "one embodiment" or "an embodiment" are intended to indicate that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. The appearance of the phrase "in one embodiment" in various places within the specification are not necessarily all referring to the same embodiment, nor are separate or alternative embodiments mutually exclusive of other embodiments. Moreover, various features are described which may be exhibited by some embodiments and not by others. Similarly, various requirements are described which may be requirements for some embodiments but not other embodiments.

[027] Some portions of the detailed descriptions, which follow, are presented in terms of procedures, steps, logic blocks, processing, and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, computer executed step, logic block, process, etc., is here, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals of a computer readable storage medium and are capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[028] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from

the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "accessing" or "writing" or "storing" or "replicating" or the like, refer to the action and processes of a computer system, or similar electronic computing device that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories and other computer readable media into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[029] Embodiments of the present invention utilize a front end scheduler, a plurality of segmented register files or a single register file, and a memory subsystem to implement fragmented address spaces for multiple cores of a multicore processor. In one embodiment, fragmentation enables the scaling of microprocessor performance by allowing additional virtual cores (e.g., soft cores) to cooperatively execute instruction sequences comprising one or more threads. The fragmentation hierarchy is the same across each cache hierarchy (e.g., L1 cache, L2 cache). The fragmentation hierarchy divides the address space into fragments using address bits, where the address bits are used such that the fragments are identified by bits that are above cache line boundaries and below page boundaries. Each fragment is configured to utilize a multiport bank structure for storage. Embodiments of the present invention are further described in the Figures 1A and 1B below.

[030] Figure 1A shows an overview diagram of a processor in accordance with one embodiment of the present invention. As depicted in Figure 1A, the processor includes a global front end fetch and scheduler 10 and a plurality of partitionable engines 11-14.

[031] Figure 1A shows an overview of the manner in which the global front end generates code blocks and inheritance vectors to support the execution of code sequences on their respective partitionable engines. Each of the code sequences 20-23 can belong to the same logical core/thread or to different logical cores/threads, depending upon the particular virtual core execution mode. The global front end fetch and scheduler will process the code sequences 20-23 to generate code blocks and

inheritance vectors. These code blocks and inheritance vectors are allocated to the particular partitionable engines 11-14 as shown.

[032] The engines implement virtual cores, in accordance with a selected mode. An engine includes a segment, a fragment and a number of execution units. The resources within the engines can be used to implement virtual cores that have multiple modes. As provisioned by the virtual core mode, one soft core, or many soft cores, can be implemented to support one logical core/thread. In the Figure 1A embodiment, depending on the selected mode, the virtual cores can support one logical core/thread or four logical cores/threads. In an embodiment where the virtual cores support four logical cores/threads, the resources of each virtual core are spread across each of the partitionable engines. In an embodiment where the virtual cores support one logical core/thread, the resources of all the engines are dedicated to that core/thread. The engines are partitioned such that each engine provides a subset of the resources that comprise each virtual core. In other words, a virtual core will comprise a subset of the resources of each of the engines 11-14. Communication between the resources of each of the engines 11-14 is provided by a global interconnection structure 30 in order to facilitate this process. Alternatively, the engines 11-14 can be used to implement a physical mode where the resources of the engines 11-14 are dedicated to support the execution of a dedicated core/thread. In this manner, the soft cores implemented by the engines comprise virtual cores that have resources spread across each of the engines. The virtual core execution modes are further described in the figures below.

[033] It should be noted that in a conventional core implementation, the resources within one core/engine are solely allocated to one logical thread/core. In contrast, in embodiments of the present invention, the resources of any engine/core can be partitioned, collectively with other engine/core partitions, to instantiate a virtual core that is allocated to one logical thread/core. Embodiments of the present invention can also implement multiple virtual execution modes in which those same engines can be partitioned to support many dedicated cores/threads or many dynamically allocated cores/threads, as well as configurations in which where all of the resources of all engines support the execution of a single core/thread. Some representative embodiments are further described below. In other embodiments of the current

invention, the techniques of the current invention can be applied directly to a conventional multi-core implementation to enable efficient contestation, reservation and allocation of multi-core shared resources and interconnects. Similarly the current invention can be applied within a single core or compute engine to enable efficient contestation, reservation and allocation of any shared resources or interconnects within the core (i.e., ports, busses, execution units, caches, structures),

[034] For example, the embodiments shown in Figure 1A , Figure 1B and Figure 5 could be replaced by a typical multi-core design that has no global front-end or inheritance vectors, but rather has engines that instantiate multiple cores or multiple threads having access to resources such as caches, shared interconnects (e.g., meshes or grids), or shared multi-directional busses. In such embodiments, the current invention can still be directly applied to allow efficient resource and interconnect contestation, reservation and allocation. Similarly, embodiments of the current invention can be applied to each core or engine in order to contend, reserve and allocate resources or interconnects.

[035] Figure 1B shows an overview diagram of partitionable engines and their components, including segmented scheduler and register files, global interconnects and a fragmented memory subsystem for a multicore processor in accordance with one embodiment of the present invention. As depicted in Figure 1, four fragments 101-104 are shown. The fragmentation hierarchy is the same across each cache hierarchy (e.g., L1 cache, L2 cache, and the load store buffer). Data can be exchanged between each of the L1 caches, each of the L2 caches and each of the load store buffers through the memory global interconnect 110a.

[036] The memory global interconnect comprises a routing matrix that allows a plurality of cores (e.g., the address calculation and execution units 121-124) to access data that may be stored at any point in the fragmented cache hierarchy (e.g., L1 cache, load store buffer and L2 cache). Figure 1 also depicts the manner whereby each of the fragments 101-104 can be accessed by address calculation and execution units 121-124 through the memory global interconnect 110a.

[037] The execution global interconnect 110b similarly comprises a routing matrix allows the plurality of cores (e.g., the address calculation and execution units 121-124) to access data that may be stored at any of the segmented register files. Thus, the cores have access to data stored in any of the fragments and 2 data stored in any of the segments through the memory global interconnect 110a or the execution global interconnect 110b.

[038] Figure 1B further shows a global front end fetch & scheduler 150 which has a view of the entire machine and which manages the utilization of the register files segments and the fragmented memory subsystem. Address generation comprises the basis for fragment definition. The global front end Fetch & scheduler functions by allocating instruction sequences to each segment's partition scheduler. The common partition scheduler then dispatches those instruction sequences for execution on the address calculation and execution units 121-124.

[039] Additionally, it should be noted that the partitionable engines shown in Figure 1A can be nested in a hierarchal way. In such an embodiment, a first level partitionable engine would include a local front end fetch and scheduler and multiple secondary partitionable engines connected to it.

[040] Figure 2 shows an overview diagram depicting additional features of the interconnect 30 described above in the discussion of Figures 1A and 1B, and a plurality of local interconnects 40-42 in accordance with one embodiment of the present invention. The Figure 2 structure illustrates an orchestrating model of an interconnect structure. Figure 2 shows a plurality of resources connected to a corresponding plurality of consumers. The resources are the data storage resources of each of the partitionable engines (e.g., register files, load store buffers, L1 cache and L2 cache). The consumers are the execution units and address calculation units of each of the partitionable engines. Figure 2 further shows a plurality of orchestrators 21-23.

[041] As described above, communication between the resources of each of the engines 11-14 is provided by an interconnection structure. By way of example, in the Figure 2 embodiment, the interconnect structure 30 is a dedicated point-to-point bus. In the Figure 2 embodiment, there are six buses which span across the resources of each of

the engines. Only one consumer/resource pair can utilize one of the six busses per cycle. The consumer/resource pairs contend with each other for use of the six busses through an OR-AND and a threshold detection logic of Figure 10, However the same orchestration for a shared multi-point busses configuration can be achieved using the reservation adder and threshold limit or process, as further described in the discussion of Figure 9.

[042] The orchestrators 21-23 comprise controlled entities that direct the routing of a resource to a consumer. For example, in one embodiment, an orchestrator can be a thread scheduler that schedules a resource for transfer through the interconnect to a consumer that is ready for execution. The orchestrator (e.g., thread scheduler) identifies the correct resource, reserves the necessary bus, and causes the transfer of that resource to a selected consumer. In this manner, the orchestrator monitors the readiness of instructions and selects the execution units that will be used to execute the instructions. This information is used to orchestrate the transfer of the resource across the interconnect to the selected execution units (e.g., selected consumer) by contending the requests at the interconnect using the reservation and allocation logic as illustrated by either of Figure 9 or Figure 10. In this manner, the execution units of the consumers themselves are treated as resources that need to be contended for by the orchestrators using similar resource reservation and allocation methods as illustrated for the interconnect. Where in the execution units are reserved and allocated by contending the requests that come from all orchestrators using either of the reservation and allocation logic of Figure 9 or Figure 10.

[043] The interconnect comprises a routing matrix that allows a plurality of resource consumers, in this case, a plurality of cores (e.g., the address calculation and execution units 121-124), to access a resource, in this case data, that may be stored at any point in the fragmented cache hierarchy (e.g., L1 cache, load store buffer and L2 cache). The cores can similarly access data that may be stored at any of the segmented register files. Thus, the cores have access to data stored in any of the fragments and to data stored in any of the segments through the interconnect structure 30. In one embodiment, the interconnect structure comprises two structures, the memory

interconnect 110a and the execution interconnect 110b, as shown and described above in the discussion of Figure 1B.

[044] Figure 2 also shows the plurality of local interconnects 40-42. The local interconnects 40-42 comprise a routing matrix that allows resource consumers from adjacent partitionable engines to quickly access resources of immediately adjacent partitionable engines. For example, one core can use a local interconnect 40 to quickly access resources of the adjacent partitionable engine (e.g., register file, load store buffer, etc.).

[045] Thus, the interconnect structure itself comprises a resource that must be shared by each of the cores of each of the partitionable engines. The interconnect structure 30 and the local interconnect structures 40-42 implement an interconnect structure that allows cores from any of the partitionable engines to access resources of any other of the partitionable engines. This interconnect structure comprises transmission lines that span all of the partitionable engines of the integrated circuit device, in the case of the interconnect structure, and span between engines of the integrated circuit device, in the case of the local interconnect structure.

[046] Embodiments of the present invention implement a non-centralized access process for using the interconnects and the local interconnects. The finite number of global buses and local buses comprise resources which must be efficiently shared by the orchestrators. Additionally, a non-centralized access process is used by the orchestrators to efficiently share the finite number of ports that provide read/write access to the resources of each of the partitionable engines. In one embodiment, the non-centralized access process is implemented by the orchestrators reserving a bus (e.g., a local interconnect bus or an interconnect bus) and a port into the desired resource. For example, orchestrator 21 needs to reserve an interconnect and a port in order for consumer 1 to access resource 3, while orchestrator 22 to needs to reserve an interconnect and the port in order for consumer for to access resource 2.

[047] Figure 3 shows components comprising a resource reservation mechanism that implements efficient access to a contested resource in accordance with one embodiment of the present invention. As shown in Figure 3, three reservation

adders 301-303 are shown coupled to threshold limiters 311-313, which control access to each of the four ports for each of the three resources. Each adder output sum (if not canceled) also serves as the port selector for each of the accesses, such that each request that succeeds can use the port number indicated by the sum at the output of that request adder. It should be noted that as indicated in the Figure 3 diagram, the sum of each depicted adder is also the assigned port number for the non-cancelled corresponding request.

[048] It should be noted that this port allocation and reservation problem can be illustrated similar to the bus segment allocation table of Figure 7 and thus its implementation logic can also be similar to Figure 9 wherein each segment in this case reflects a register file segment instead of a bus segment. With the same analogy in this case, an instruction trying to access multiple register file segments can only succeed if it can reserve all its register segments requests, and will fail if any register segment access for that instruction is canceled, similar to the illustrations of the bus segments in Figure 7.

[049] Embodiments of the present invention implement a non-centralized access process for using the interconnects and the local interconnects. Requests, accesses and controls can be initiated for shared interconnects, resources or consumers by multiple non-centralized fetchers, senders, orchestrators, or agents. Those non centralized requests, accesses and controls contend at the shared resources using variations of methods and logic implementation as described in this invention depending on the topologies and structures of those shared resources. By way of example, the resources of the engines and their read/write ports need to be efficiently shared by the cores. Additionally, the finite number of global buses and local buses comprise resources that need to be efficiently shared. In the Figure 3 embodiment, the non-centralized access process is implemented through reservation adders and threshold limiters. In one embodiment, at each contested resource, a reservation adder tree and a threshold limiter control access to that contested resources. As used herein, the term contested resource refers to read write ports of a load store buffer, memory/cache fragment, register file segment or L2 cache, a global buses reservation, or local buses reservation.

[050] A reservation adder and a threshold limiter control access to each contested resource. As described above, to access a resource, a core needs to reserve the necessary bus and reserve the necessary port. During each cycle, orchestrators attempt to reserve the resources necessary to execute their pending instruction. For example, for an orchestrator scheduling an instruction I1 shown in Figure 3, that orchestrator will set a flag, or a bit, in the reservation adder of its needed resource. In this case a bit is set in register file 1 and in register file 3. Other orchestrators will similarly set bits in the reservation adders of their needed resource. For example, a different orchestrator for instruction I2 sets two bits for register file 2. As the orchestrators request their needed resources the reservation adders sum the requests until they reach the threshold limiter. In the Figure 4 embodiment, there are four ports for each of the resources. Hence, the reservation adders will accept flags from reservation requests until the four ports are all reserved. No other flags will be accepted.

[051] An orchestrator will not receive confirmation to execute its instruction unless all of its flags necessary to execute the instruction are set. Hence, the orchestrator will receive confirmation to execute the instruction if the flags for the necessary buses are set and the flags for the necessary read write ports are set. If a cancel signal is received for any of the flags, all flags for that orchestrator's request are cleared, and the request is queued until the next cycle.

[052] In this manner, each of the orchestrators contends with each other for the resources on a cycle by cycle basis. Requests that are canceled are queued and given priority in the next cycle. This ensures that one particular core is not locked out of resource access for large number of cycles. It should be noted that the resources in the proposed implementations get assigned automatically to the resources, for example if the request succeed in obtaining a resource (e.g., it is not canceled by the adder and threshold logic) then the adder sum output corresponding to that request represent the resource number assigned to that request, thus completing the resource assignment without requiring any further participation from the orchestrators. This reservation and allocation adder and threshold limiters fairly balance access to contested resources in a decentralized manner (e.g., there is no need for requestors/orchestrators to actively

participate in any centralized arbitration). Each remote orchestrator sends its requests to the shared resources, those requests that succeed will be granted resources/buses automatically.

[053] Figure 4 shows the interconnect and the ports into the memory fragments in accordance with one embodiment of the present invention. As depicted in Figure 4, each memory fragment is shown with four read write ports that provide read/write access to the load store buffer, the L1 cache, and the L2 cache. The load store buffer includes a plurality of entries and the L1 cache includes a plurality of ways.

[054] As described above, embodiments of the present invention implement a non-centralized access process for using the interconnects and the local interconnects. The finite number of global buses and local buses comprise resources which must be efficiently shared by the cores. Thus, a reservation adder and a threshold limiter control access to each contested resource, in this case, the ports into each fragment. As described above, to access a resource, a core needs to reserve the necessary bus and reserve the necessary port.

[055] Figure 5 shows the interconnect and the ports into the segments in accordance with one embodiment of the present invention. As depicted in Figure 5, each segment is shown with 4 read write ports that provide read/write access to the operand/result buffer, threaded register file, and common partition or scheduler. The Figure 5 embodiment is shown as including a common partition or scheduler in each of the segments. In this embodiment, the common partition scheduler is configured to function in cooperation with the global front end fetch and scheduler shown in Figure 1B.

[056] The non-centralized access process for using the interconnects and the local interconnects employ the reservation adder and a threshold limiter control access to each contested resource, in this case, the ports into each segment. As described above, to access a resource, a core needs to reserve the necessary bus and reserve the necessary port.

[057] Figure 6 shows a diagram depicting a segmented interconnect 601 in accordance with one embodiment of the present invention. As shown in Figure 6, an interconnect 601 is shown connecting resources 1-4 to consumers 1-4. The interconnect 601 is also shown as comprising segments 1, 2, and 3.

[058] Figure 6 shows an example of a fetch model interconnect structure. In the Figure 6 embodiment, there are no orchestrators. In this embodiment, the resources are contended for by the consumers, as they attempt to fetch the necessary resources to support consumption (e.g., execution units). The consumers send the necessary fetch requests to the reservation adders and threshold limiters.

[059] The interconnect structure comprises a plurality of global segmented buses. The local interconnect structure comprises a plurality of locally connected engine to engine buses. Accordingly, to balance costs in both performance and fabrication, there are a finite number of global buses and a finite number of local buses. In the Figure 6 embodiment, four globally segmented buses are shown.

[060] In one embodiment, the global buses can be segmented into 3 portions. The segmentation allows the overall length of the global buses to be adjusted in accordance with the distance of the global access. For example, an access by consumer 1 to resource 4 would span the entire bus, and thus not be segmented. However, an access by consumer 1 to resource 3 would not span the entire bus, and thus the global bus can be segmented between resource 3 and resource 4.

[061] In the Figure 6 embodiment, the interconnect 601 is shown as having 4 buses. The segmentation can be implemented via, for example, a tri-state buffer. The segmentation results in faster and more power efficient transmission characteristics of the bus. In the Figure 6 embodiment, the buses each include one directional tri-state buffers (e.g., buffer 602) and bidirectional tri-state buffers (e.g., buffer 603). The bidirectional tri-state buffers are shaded in the Figure 6 diagram. The buffers enable the interconnect to be segmented to improve its signal transmission characteristics. These segments also comprise resources which must be contested for an allocated for by the resource consumers. This process is illustrated in the Figure 7 diagram below.

[062] Figure 7 shows a table that illustrates the manner which requests for segments of the interconnect 601 are contested for and allocated in accordance with one embodiment of the present invention. The left-hand side of the Figure 7 table shows how requests are ordered as they are received within the cycle. In this case, eight requests are shown. When a request from a resource consumer wants to reserve a segment, that consumer places a one in the requested segment's reservation table. For example, for request 1, consumer 1 wants to reserve segment 1 and segment 2 in order to access resource 3. Thus, consumer 1 sets a flag, or a bit, in the request column for segment 1 and segment 2, while the column for segment 3 remains zero. In this manner, requests are added within the columns. Requests are allocated until they exceed the number of global buses, in this case four. When the requests exceed the number of global buses, they are canceled. This is shown by request number 6 and request number 7 having been canceled because they exceed the limit.

[063] Figure 8 shows a table that illustrates the manner in which requests for a point-to-point bus are handled in accordance with one embodiment of the present invention. As opposed to the table of Figure 7, the table of Figure 8 shows how only one consumer and only one resource can use a point-to-point bus (e.g., the interconnect illustrated in Figure 2). The requests come from the multiple orchestrators that want to route resources through the point-to-point buses. In this case, the point-to-point bus shows the number of possible consumer resource pairs (e.g., the six columns proceeding from left to right) and a number of requests 1-8 proceeding from top to bottom. Because only one resource consumer pair can use a bus at any given time, the column can only have one request flag before all of the requests are canceled as exceeding the limit. Thus, in each column, the first request is granted while all subsequent requests are canceled as exceeding the limit. Since there are six global point-to-point buses, there are six columns which can accommodate six different requests in each cycle.

[064] Figure 9 shows a diagram of an exemplary logic implementation that implements the functionality of the table of Figure 7 in accordance with one embodiment of the present invention. As described above, the table of Figure 7 illustrates the manner which requests for segments of the interconnect 601 are contested for and allocated in accordance with one embodiment of the present invention.

Specifically, Figure 9 shows the logic for allocating the column associated with bus segment 2 from the table of Figure 7.

[065] The Figure 9 embodiment shows a plurality of parallel adders 901-905. Both requests are canceled if the limit is exceeded. As described above, there are 4 buses which can be used to implement segment 2. The first four requests can be processed and granted because even if they are all flagged, by marking request with a logical one, they will not exceed the limit. The remaining requests need to be checked whether they will exceed the limit. This is done by the parallel adders 901-905. Each adder after the first three rows adds itself and all previous rows and checks against the limit. If the adder exceeds the limit, the request is canceled, as shown. The adder sum output also determines which particular bus segment is allocated to each request. In the Figure 9 embodiment, this is by bus segment number as shown.

[066] Figure 10 shows a diagram of an exemplary logic implementation that implements the functionality of the manner in which requests for a point-to-point bus are handled in accordance with one embodiment of the present invention. The table of Figure 8 shows how only one consumer and only one resource can use a point-to-point bus. Specifically, Figure 10 shows the logic for allocating the column associated with bus column 2-4 from the table of Figure 8.

[067] The Figure 10 embodiment shows a plurality of multi-input OR gates coupled to AND gates, as shown. As described above, one consumer and only one resource can use a point-to-point bus. Because only one resource/consumer pair can use a bus at any given time, the column can only have one request flag before all of the subsequent requests are canceled as exceeding the limit. Thus, in each column, the first request is granted while all subsequent requests are canceled as exceeding the limit. In the Figure 10 embodiment, each row of the column is logically combined through an OR operation with all of the previous rows of the column and then is logically combined through an AND operation with itself. Thus, if any previous row reserves the column, all subsequent requests are canceled, as shown.

[068] Figure 11 shows a diagram of an interconnect 1101 in accordance with one embodiment of the present invention. The interconnect 1101 comprises five shared interconnect structures that are shared by each of the senders and each of the receivers.

[069] The Figure 11 embodiment shows an example of a send model interconnect structure. For example, the senders comprise the execution units of the engines. The receivers comprise the memory fragments and the register segments of the engines. In this model, the senders issue the necessary requests to the reservation adders and the threshold limiters to reserve resources to implement their transfers. These resources include ports into the receivers and a plurality of shared buses of the interconnect 1101.

[070] Figure 12 shows a table illustrating the manner in which the sender model interconnect structure of Figure 11 functions in accordance with one embodiment of the present invention. The table shows the requests as they are received from all of the senders. The right hand side of the table shows the interconnect allocation. Since the interconnect 1101 comprises five shared buses, the first five requests are granted, and any further requests are canceled as exceeding the limit. Thus, request 1, request 3, request 4, request 5, and request 6 are granted. However, request 7 is canceled as having exceeded the limit.

[071] Figure 13 shows a diagram of an exemplary logic implementation that implements the functionality of the manner in which requests for shared bus interconnect structure are handled in accordance with one embodiment of the present invention.

[072] Figure 13 shows how the allocation of the interconnect buses is handled by the adders 901-905. This logic implements the table of Figure 12. As requests are received, corresponding flags are set. The adders add their respective flag with all prior flags. Flags will be granted along with their bus number by the adder so long as they do not exceed the limit, which is five in this case. As described above, any requests that exceed the limit are canceled.

[073] It should be noted that the sender model and the fetch model of an interconnect can be simultaneously supported using a common interconnect structure and a common contesting mechanism. This is shown by the similarity of the diagram of Figure 13 to the diagram of Figure 9.

[074] It should be noted that current presentations in the current invention of different models of communications (Sender, Fetch, Orchestrator, etc.) and different interconnect topologies (point to point busses, multi-bus, and segmented busses, etc.) should not be interpreted as the only communication modes or the only interconnect topologies applicable to the current invention. To the contrary, one skilled in the art can easily mix and match the different contestation, reservation and allocation techniques of the current invention with any communication mode or bus topology.

[075] It should be further noted that the described embodiments of the current invention present interconnects alongside the resources. This should be understood as a generalized illustration meant to show a broader set of possibilities for implementing the current invention, but it should be noted that the meaning of interconnects as used in the current invention is not limited to data interconnects between different cores or compute engines or between register files or memory fragments, but refers also to the control interconnects that carry the requests to the resources and the physical interconnects that carry data from structures (i.e., register file ports, memory ports, array decoder busses, etc.). This broader meaning is illustrated in Figure 3, for example, which shows the interconnects only as the ports coming out of each register file.

[076] Figure 14 shows a diagram of an exemplary microprocessor pipeline 1400 in accordance with one embodiment of the present invention. The microprocessor pipeline 1400 includes a fetch module 1401 that implements the functionality of the process for identifying and extracting the instructions comprising an execution, as described above. In the Figure 14 embodiment, the fetch module is followed by a decode module 1402, an allocation module 1403, a dispatch module 1404, an execution module 1405 and a retirement module 1406. It should be noted that the microprocessor pipeline 1400 is just one example of the pipeline that implements the functionality of embodiments of the present invention described above. One skilled in the art would

recognize that other microprocessor pipelines can be implemented that include the functionality of the decode module described above.

[077] For purposes of explanation, the foregoing description refers to specific embodiments that are not intended to be exhaustive or to limit the current invention. Many modifications and variations are possible consistent with the above teachings. Embodiments were chosen and described in order to best explain the principles of the invention and its practical applications, so as to enable others skilled in the art to best utilize the invention and its various embodiments with various modifications as may be suited to their particular uses.

CLAIMS

What is claimed is:

1. An interconnect system, comprising

a plurality of resources having data for supporting the execution of multiple code sequences

a plurality of engines for implementing the execution of the multiple code sequences;

a plurality of resource consumers within each of the plurality of engines

an interconnect structure for coupling the plurality of resource consumers with the plurality of resources to access the data and execute the multiple code sequences, wherein the resource consumers access the resources through a per cycle utilization of the interconnect structure.

2. The interconnect system of claim 1, wherein the resource consumers comprise execution units of the engines.

3. The interconnect system of claim 1, wherein the resources comprise memory fragments.

4. The interconnect system of claim 1, wherein the resources comprise register file segments.

5. The interconnect system of claim 1, wherein the interconnect structure comprises a plurality of point-to-point buses that resource consumers access to resources through the per cycle utilization.

6. The interconnect system of claim 1, wherein the interconnect structure comprises a plurality of segmented buses that resource consumers access to resources through the per cycle utilization.

7. The interconnect system of claim 1, wherein the interconnect structure comprises a memory interconnect structure and an execution interconnect structure.

8. The interconnect system of claim 1, wherein the system further includes a plurality of local interconnect structures that enable adjacent engines to directly access data from adjacent resources.

9. A microprocessor, comprising

a plurality of resources having data for supporting the execution of multiple code sequences

a plurality of engines for implementing the execution of the multiple code sequences

a plurality of resource consumers within each of the plurality of engines

an interconnect structure for coupling the plurality of resource consumers with the plurality of resources to access the data and execute the multiple code sequences, wherein the resource consumers access the resources through a per cycle utilization of the interconnect structure.

10. The microprocessor of claim 9, wherein the resource consumers comprise execution units of the engines.

11. The microprocessor of claim 9, wherein the resources comprise memory fragments.

12. The microprocessor of claim 9, wherein the resources comprise register file segments.

13. The microprocessor of claim 9, wherein the interconnect structure comprises a plurality of point-to-point buses that resource consumers access to resources through the per cycle utilization.

14. The microprocessor of claim 9, wherein the interconnect structure comprises a plurality of segmented buses that resource consumers access to resources through the per cycle utilization.

15. The microprocessor of claim 9, wherein the interconnect structure comprises a memory interconnect structure and an execution interconnect structure.

16. The microprocessor of claim 9, wherein the system further includes a plurality of local interconnect structures that enable adjacent engines to directly access data from adjacent resources.

17. A computer system having a microprocessor coupled to a computer readable memory, wherein the microprocessor comprises:

a plurality of resources having data for supporting the execution of multiple code sequences

a plurality of engines for implementing the execution of the multiple code sequences

a plurality of resource consumers within each of the plurality of engines

an interconnect structure for coupling the plurality of resource consumers with the plurality of resources to access the data and execute the multiple code sequences, wherein the resource consumers access the resources through a per cycle utilization of the interconnect structure, and wherein the resource consumers comprise execution units of the engines.

18. The computer system of claim 17, wherein the resources comprise memory fragments.

19. The computer system of claim 17, wherein the resources comprise register file segments.

20. The computer system of claim 17, wherein the interconnect structure comprises a plurality of point-to-point buses that resource consumers access to resources through the per cycle utilization.

21. The computer system of claim 17, wherein the interconnect structure comprises a plurality of segmented buses that resource consumers access to resources through the per cycle utilization.

22. The computer system of claim 17, wherein the interconnect structure comprises a memory interconnect structure and an execution interconnect structure.

23. The computer system of claim 17, wherein the system further includes a plurality of local interconnect structures that enable adjacent engines to directly access data from adjacent resources.

24. The computer system of claim 17, wherein interconnect structure comprises a fetch model interconnect structure.

25. The computer system of claim 17, wherein the interconnect structure comprises a send model interconnect structure.

26. The computer system of claim 17, wherein the interconnect structure comprises an orchestrate model interconnect structure.

27. The computer system of claim 17, further comprising:

an adder structure that sums requests for access to the plurality of resources and that particularly assigns a unique port for each successful request in accordance with adder structure output sums.

28. The computer system of claim 17, further comprising:

an adder structure that sums requests for access to the plurality of resources and that particularly assigns a unique bus and a unique port for each successful request in

accordance with adder structure output sums to arbitrate and allocate a plurality of resources on a per cycle basis.

29. The computer system of claim 17, further comprising:

an adder structure that functions in accordance with a fetch model, a send model, or an orchestrate model, and that sums requests for access to the plurality of resources and that particularly assigns a unique bus and a unique port for each successful request in accordance with adder structure output sums to arbitrate and allocate a plurality of resource on a per cycle basis, and in accordance with said fetch model, said send model, or said orchestrate model.

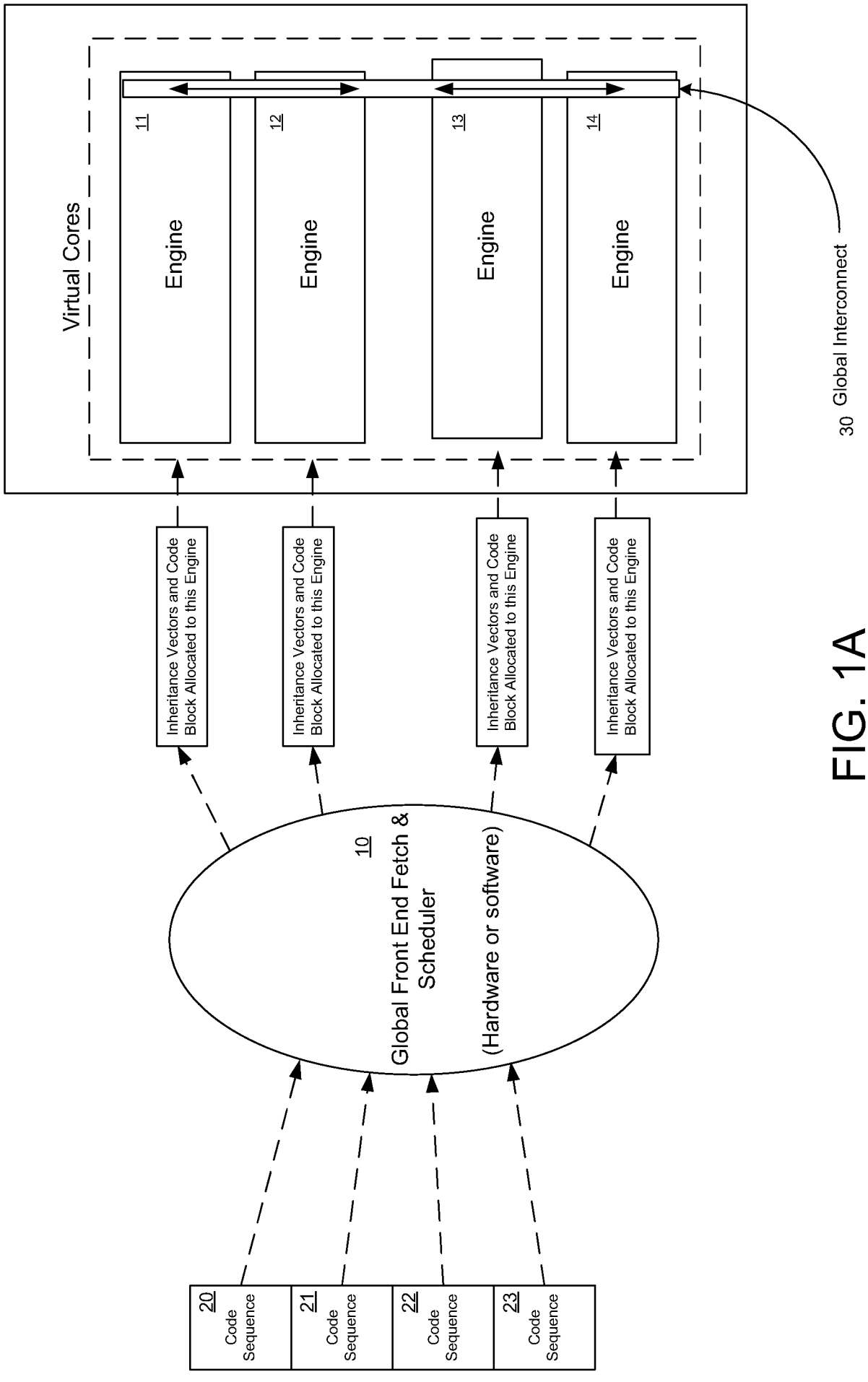
Virtual Cores
11
Engine
12
Engine
13
Engine
14
Engine
30 Global Interconnect
Inheritance Vectors and Code Block Allocated to this Engine
Inheritance Vectors and Code Block Allocated to this Engine
Inheritance Vectors and Code Block Allocated to this Engine
Inheritance Vectors and Code Block Allocated to this Engine
10
Global Front End Fetch & Scheduler
(Hardware or software)
20 Code Sequence
21 Code Sequence
22 Code Sequence
23 Code Sequence

FIG. 1A

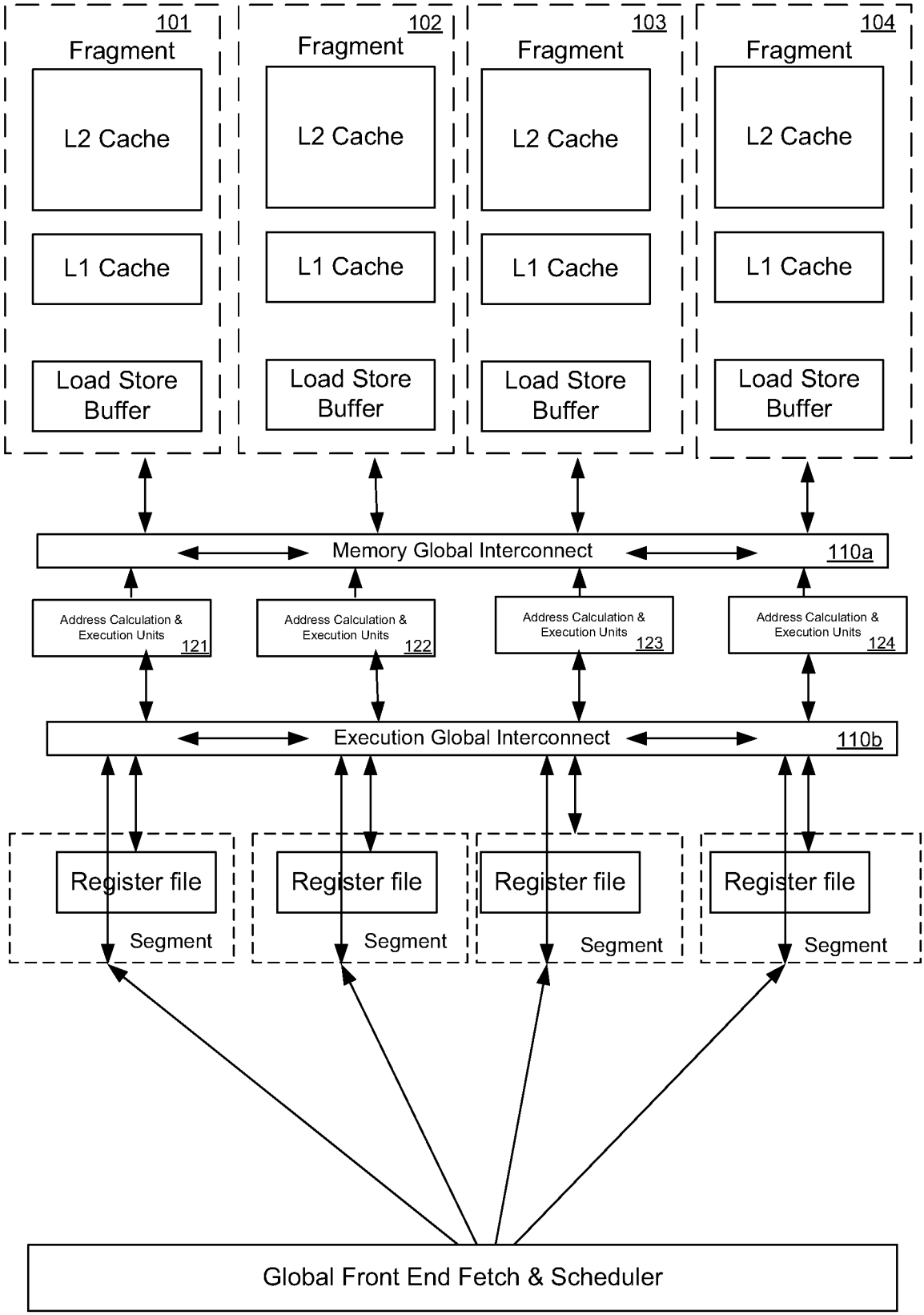
101
Fragment
L2 Cache
L1 Cache
Load Store Buffer
102
Fragment
L2 Cache
L1 Cache
Load Store Buffer
103
Fragment
L2 Cache
L1 Cache
Load Store Buffer
104
Fragment
L2 Cache
L1 Cache
Load Store Buffer
Memory Global Interconnect
110a
Address Calculation & Execution Units
121
Address Calculation & Execution Units
122
Address Calculation & Execution Units
123
Address Calculation & Execution Units
124
Execution Global Interconnect
110b
Register file
Segment
Register file
Segment
Register file
Segment
Register file
Segment
Global Front End Fetch & Scheduler

FIG. 1B

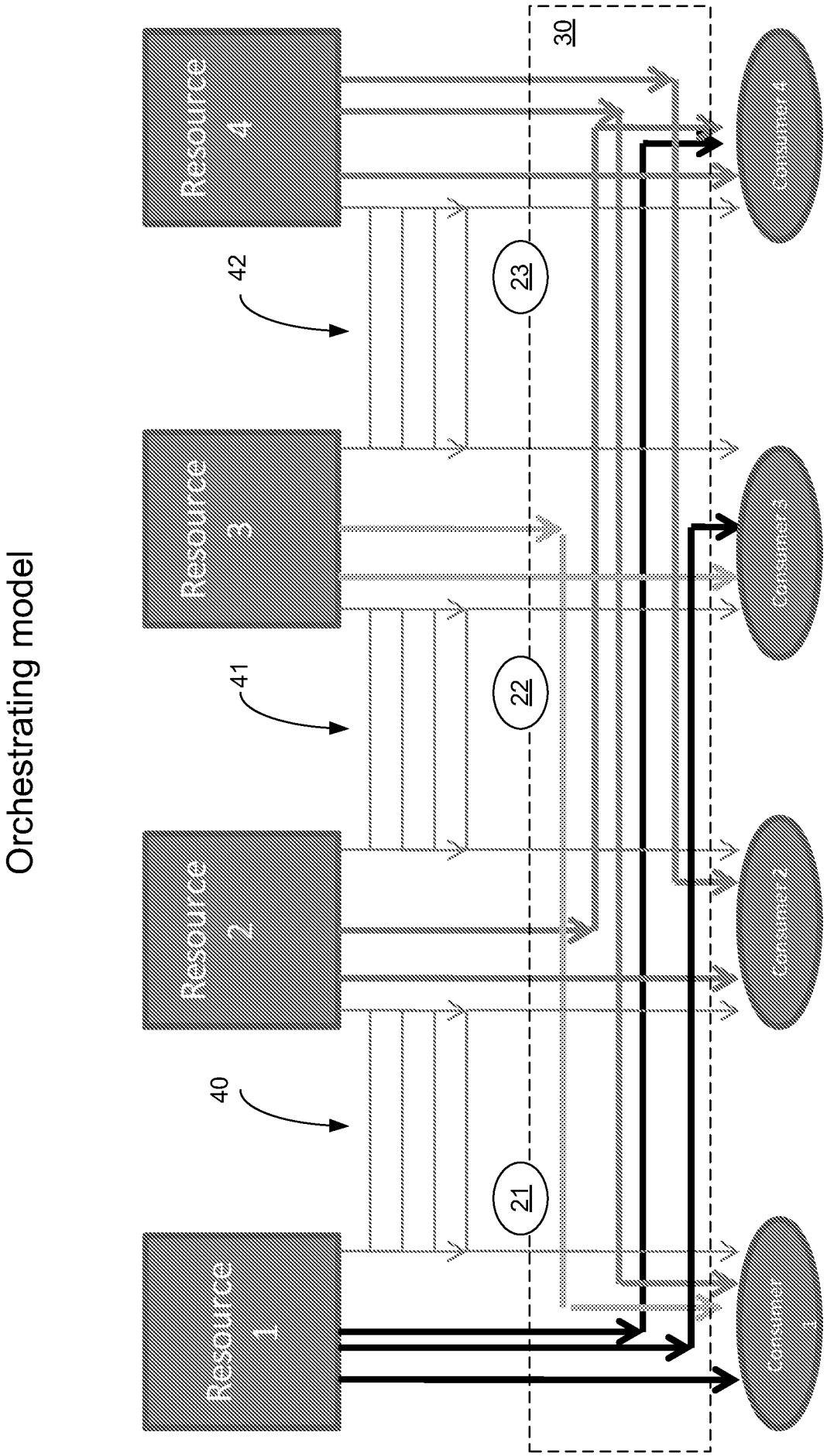
Orchestrating model
Resource 1
Resource 2
Resource 3
Resource 4
Consumer 1
Consumer 2
Consumer 3
Consumer 4
30
40
41
42
21
22
23

FIG. 2

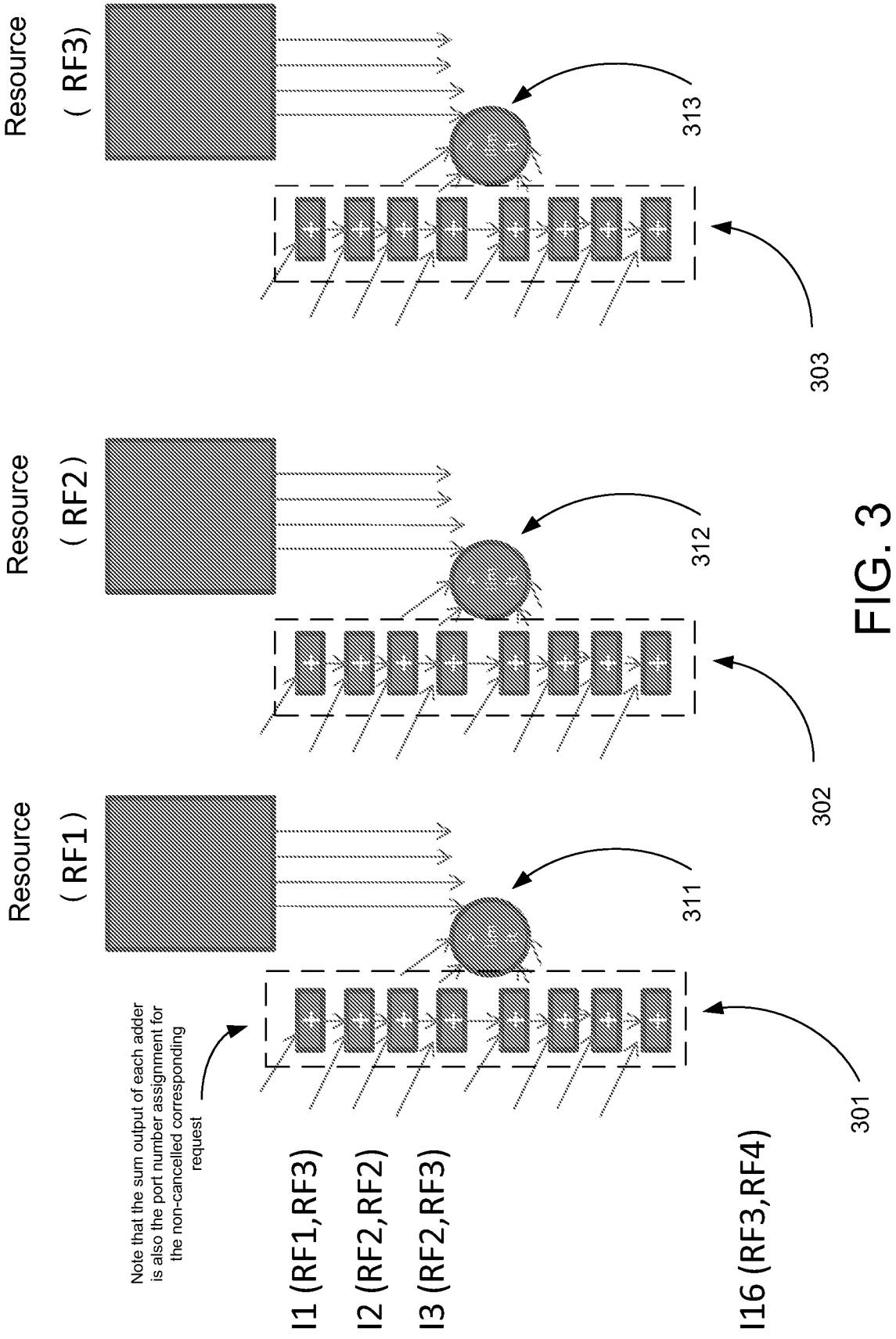
Resource
(RF3)
313
303
Resource
(RF2)
312
302
Resource
(RF1)
311
301
Note that the sum output of each adder is also the port number assignment for the non-cancelled corresponding request
I1 (RF1,RF3)
I2 (RF2,RF2)
I3 (RF2,RF3)
I16 (RF3,RF4)

FIG. 3

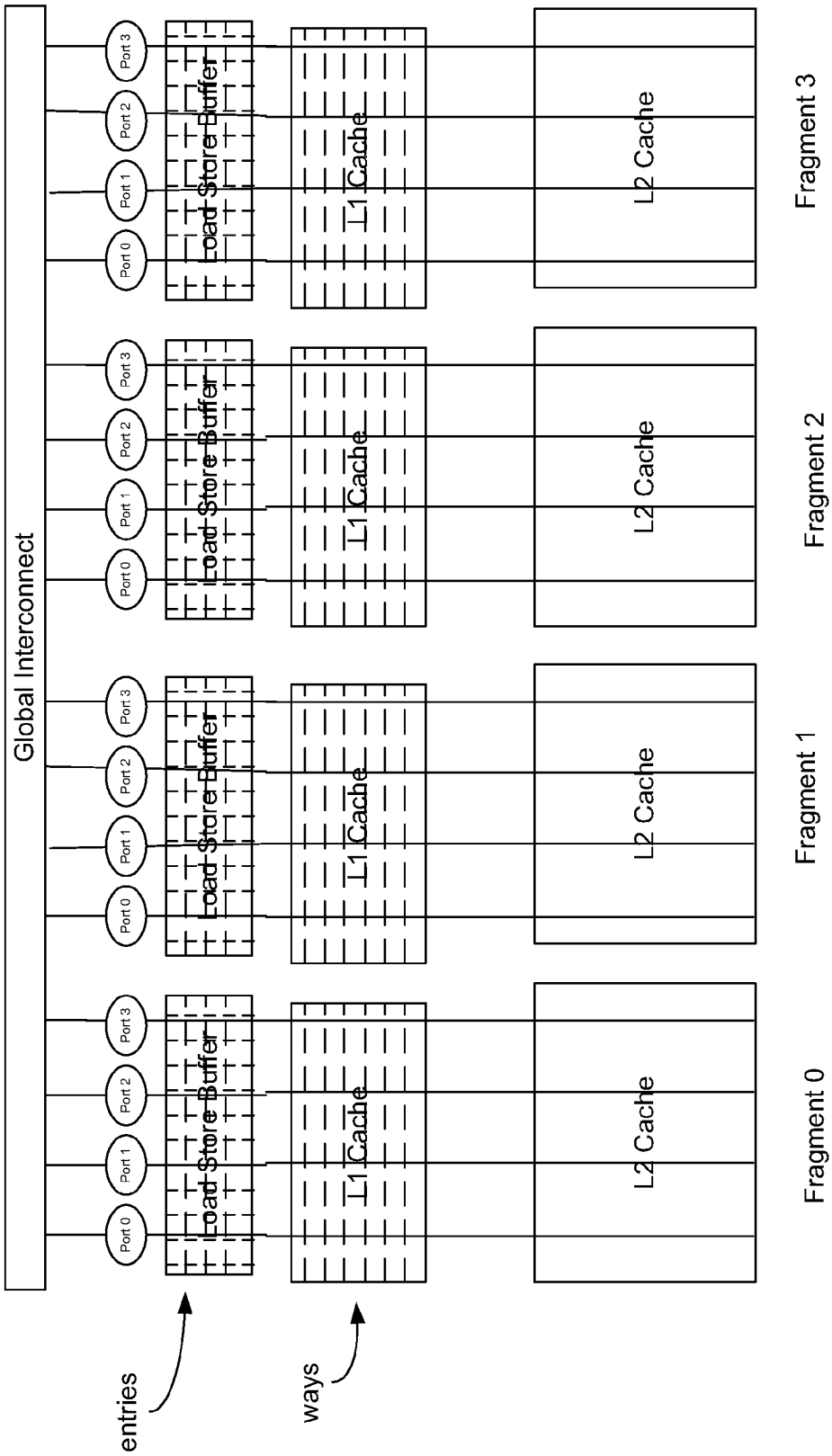
Global Interconnect
Port 0
Port 1
Port 2
Port 3
Load Store Buffer
L1 Cache
L2 Cache
Fragment 0
Fragment 1
Fragment 2
Fragment 3
entries
ways

FIG. 4

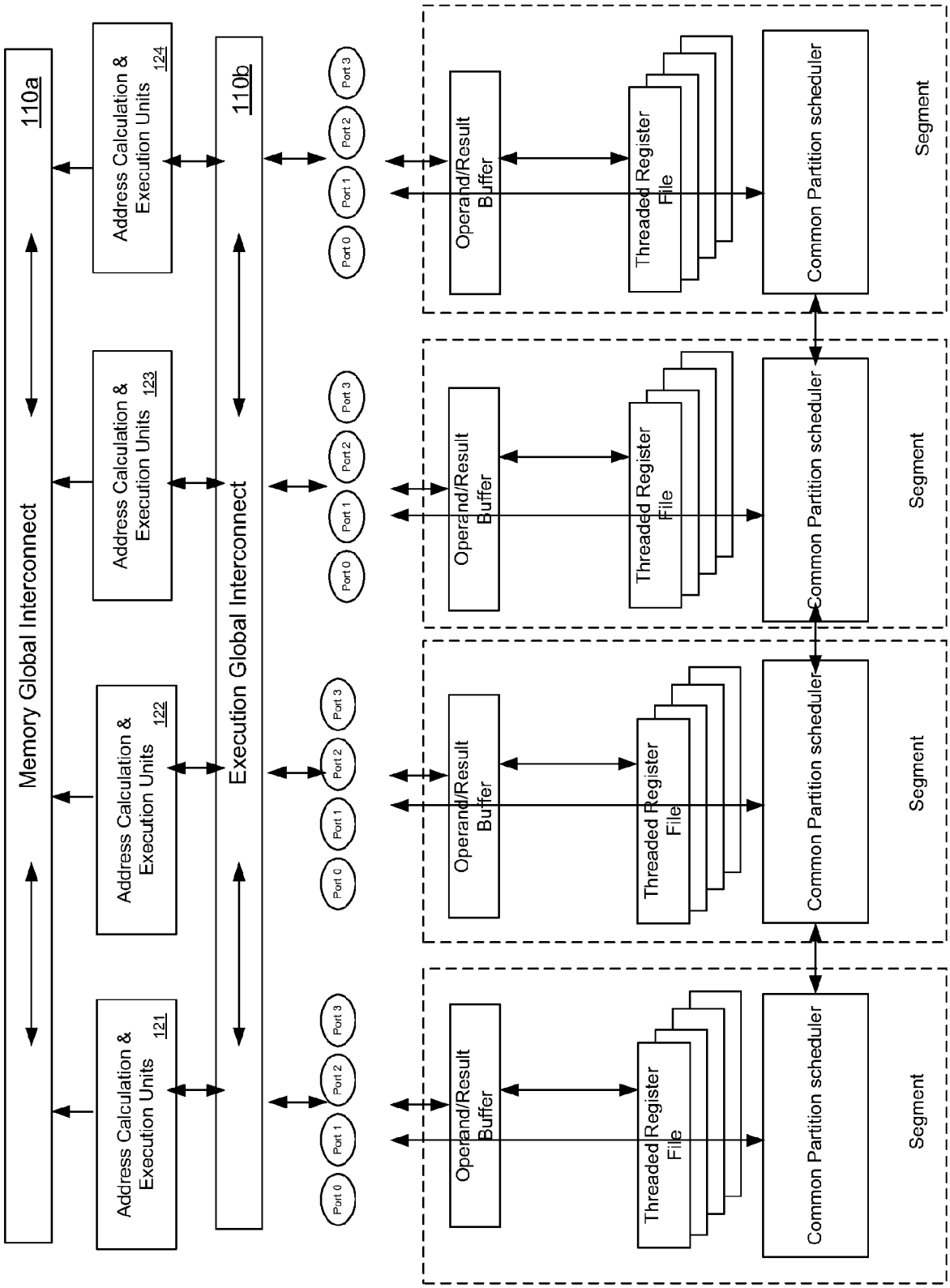
Memory Global Interconnect
110a
Execution Global Interconnect
110b
Address Calculation & Execution Units
121
Address Calculation & Execution Units
122
Address Calculation & Execution Units
123
Address Calculation & Execution Units
124
Port 0
Port 1
Port 2
Port 3
Operand/Result Buffer
Threaded Register File
Common Partition scheduler
Segment

FIG. 5

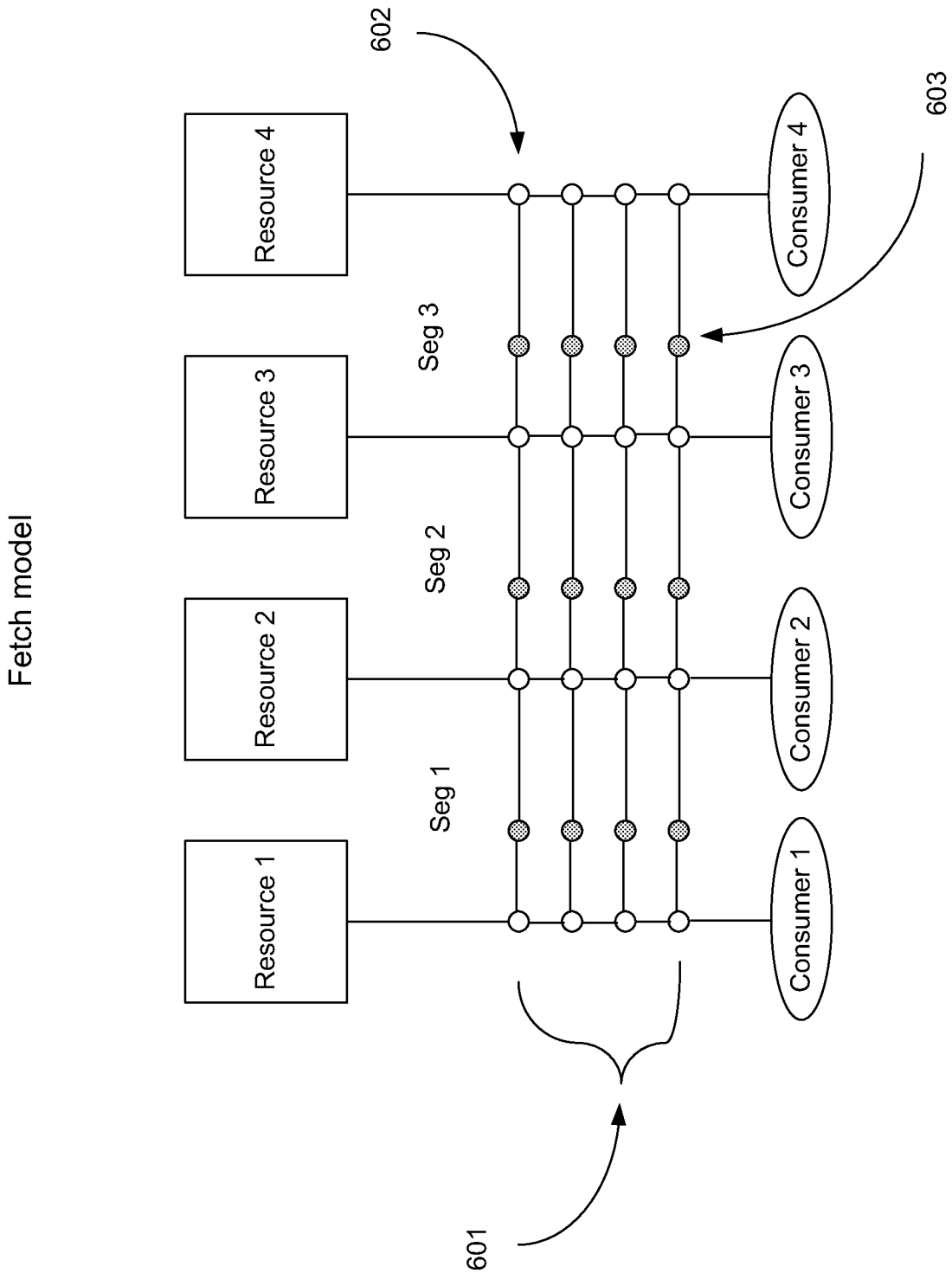
Fetch model
Resource 1
Resource 2
Resource 3
Resource 4
Seg 1
Seg 2
Seg 3
Consumer 1
Consumer 2
Consumer 3
Consumer 4
601
602
603

FIG. 6

	Segment 1	Segment 2	Segment 3
Request 1	1	1	0
Request 2	0	0	1
Request 3	0	1	0
Request 4	0	1	1
Request 5	1	1	0
Request 6 (cancelled)	1	1 (exceed limit)	0
Request 7 (cancelled)	1	1 (exceed limit)	1
Request 8	0	0	1

FIG. 7

Requests come from multiple orchestrators that want to route through the point to point busses

Only one pair uses of the bus, point to point

	1-3	3-1	4-1	**1-4**	2-4	4-2
Request 1	0	1	0	0	0	0
Request 2	0	0	1	0	0	1
Request 3	0	0	0	0	1	0
Request 4 (cancelled)	0	0	0	0	1 (exceed limit)	0
Request 5	1	0	0	0	0	0
Request 6 (cancelled)	0	1 (exceed limit)	0	1	0	0
Request 7 (cancelled)	1 (exceed limit)	1 (exceed limit)	0	0	0	0
Request 8	0	0	0	0	0	0

FIG. 8

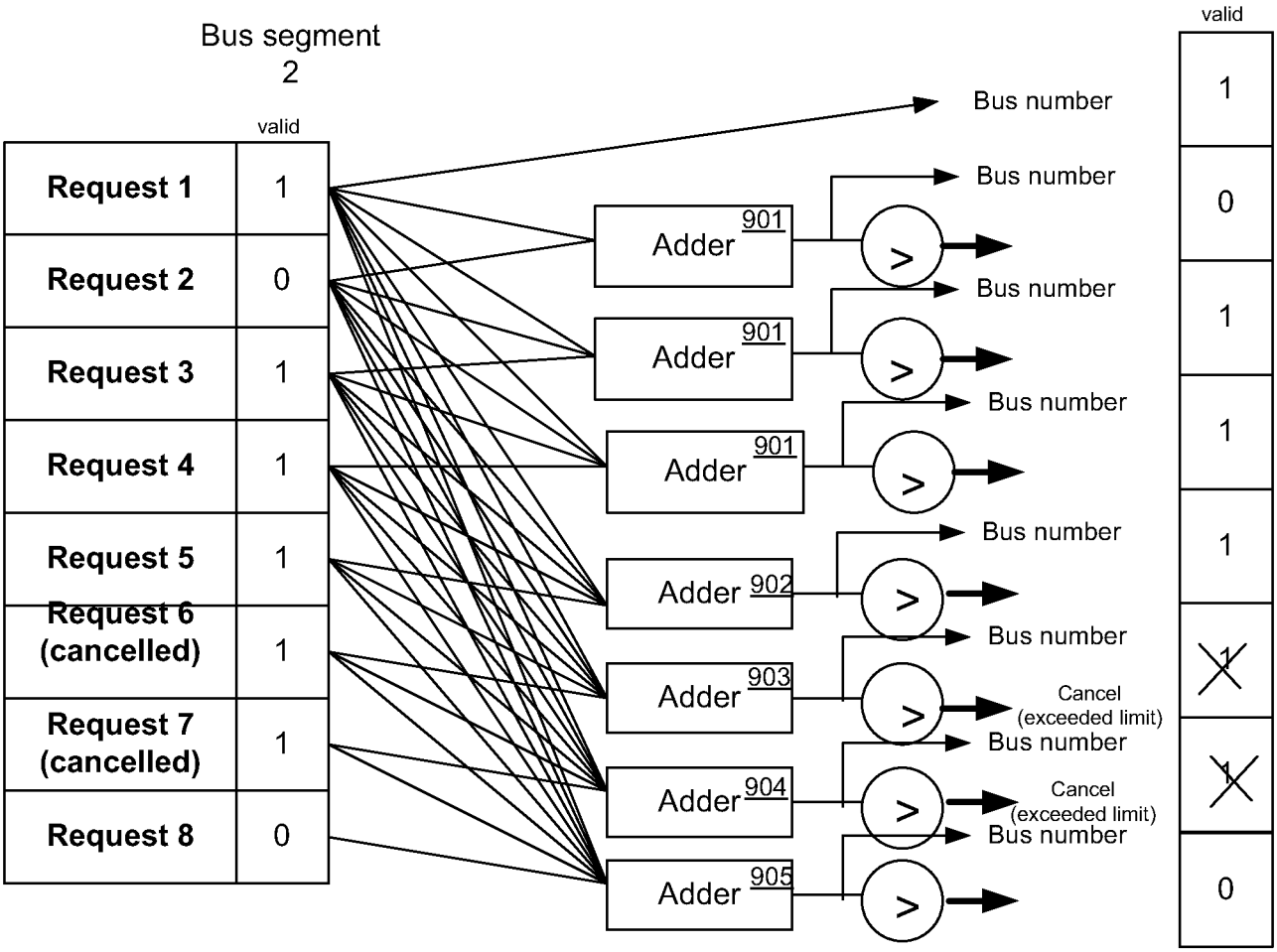
Bus segment 2
valid
Request 1
1
Request 2
0
Request 3
1
Request 4
1
Request 5
1
Request 6 (cancelled)
1
Request 7 (cancelled)
1
Request 8
0
Adder 901
Adder 901
Adder 901
Adder 902
Adder 903
Adder 904
Adder 905
>
Bus number
Bus number
Bus number
Bus number
Bus number
Cancel (exceeded limit)
Bus number
Cancel (exceeded limit)
Bus number
valid
1
0
1
1
1
1
0

FIG. 9

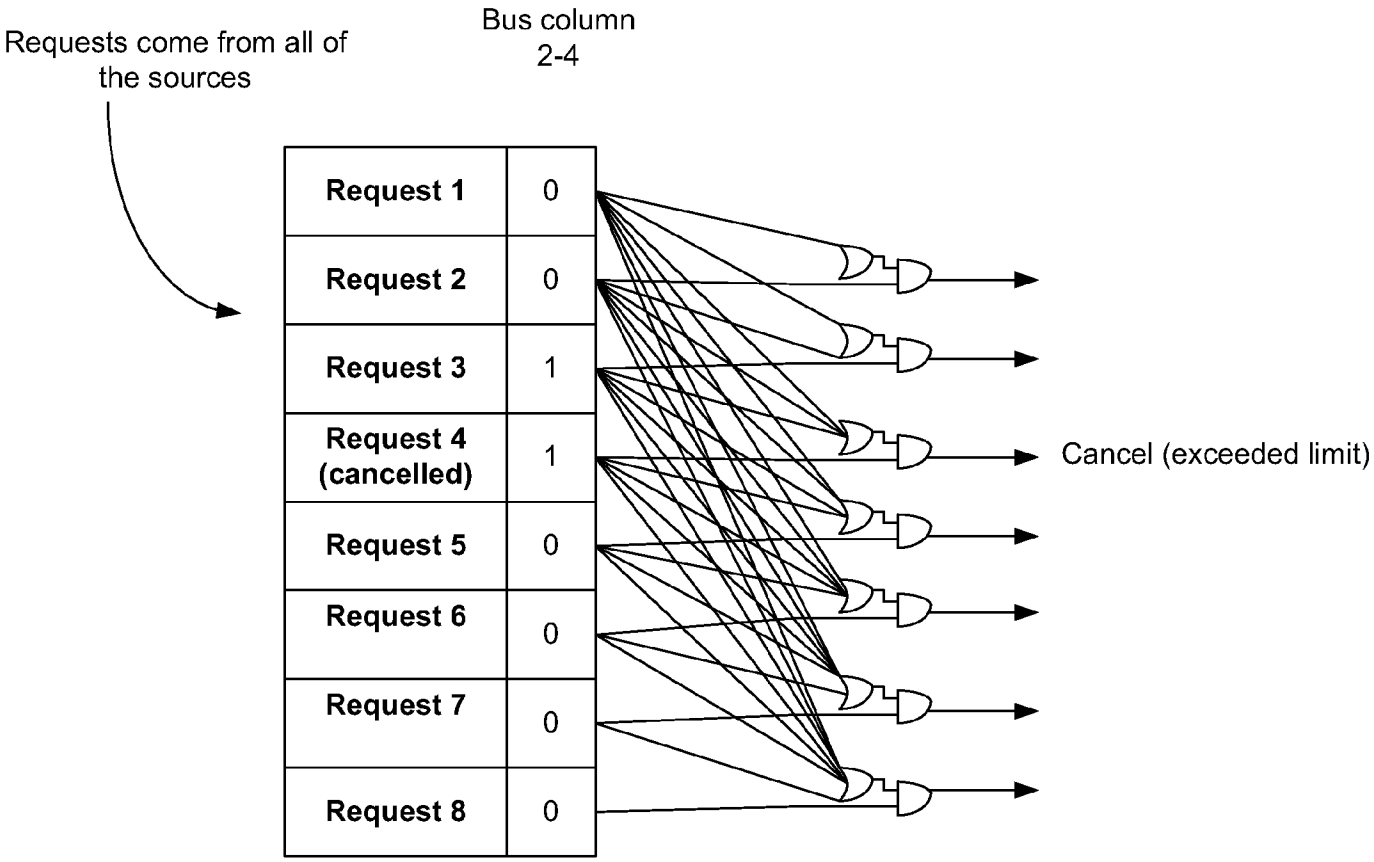
Requests come from all of the sources
Bus column 2-4
Request 1
0
Request 2
0
Request 3
1
Request 4 (cancelled)
1
Request 5
0
Request 6
0
Request 7
0
Request 8
0
Cancel (exceeded limit)

FIG. 10

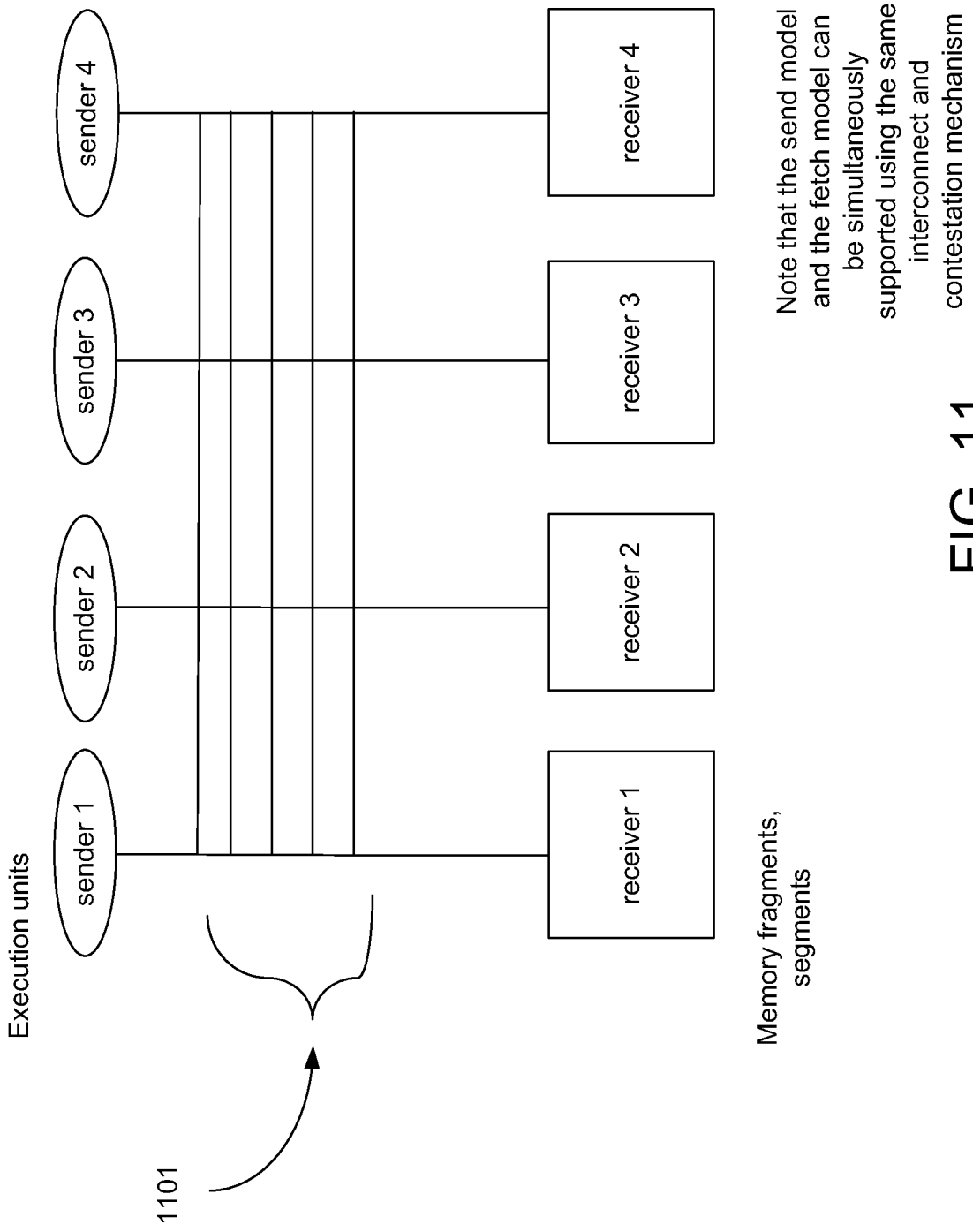
Send Model
Execution units
sender 1
sender 2
sender 3
sender 4
receiver 1
receiver 2
receiver 3
receiver 4
1101
Memory fragments, segments
Note that the send model and the fetch model can be simultaneously supported using the same interconnect and contestation mechanism

FIG. 11

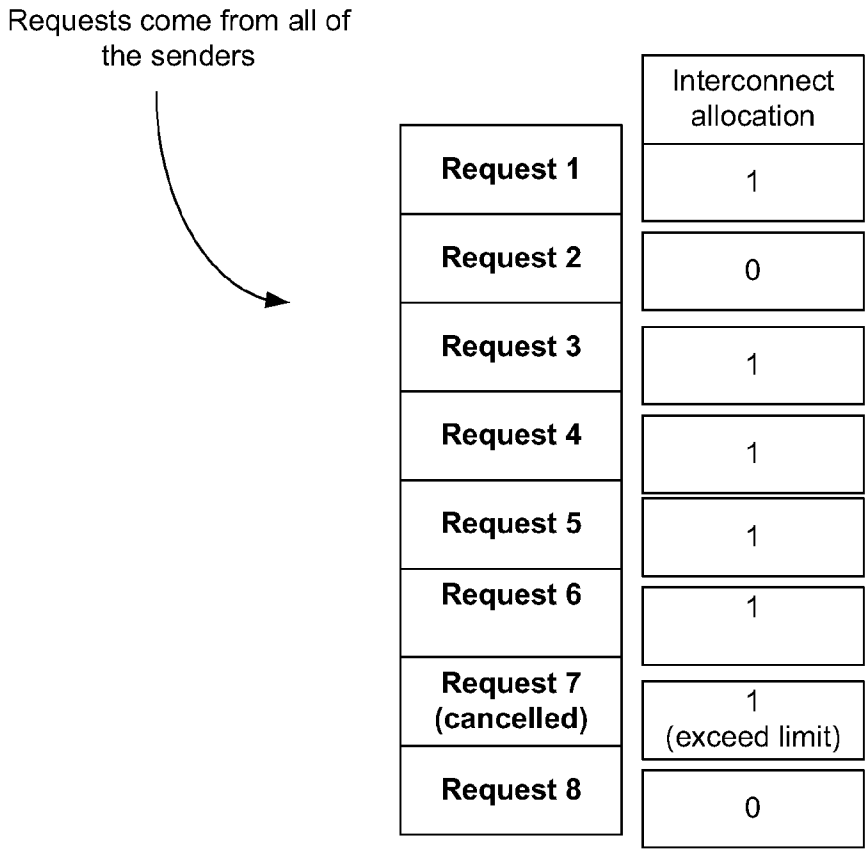
Requests come from all of the senders
Interconnect allocation
Request 1
1
Request 2
0
Request 3
1
Request 4
1
Request 5
1
Request 6
1
Request 7 (cancelled)
1 (exceed limit)
Request 8
0

FIG. 12

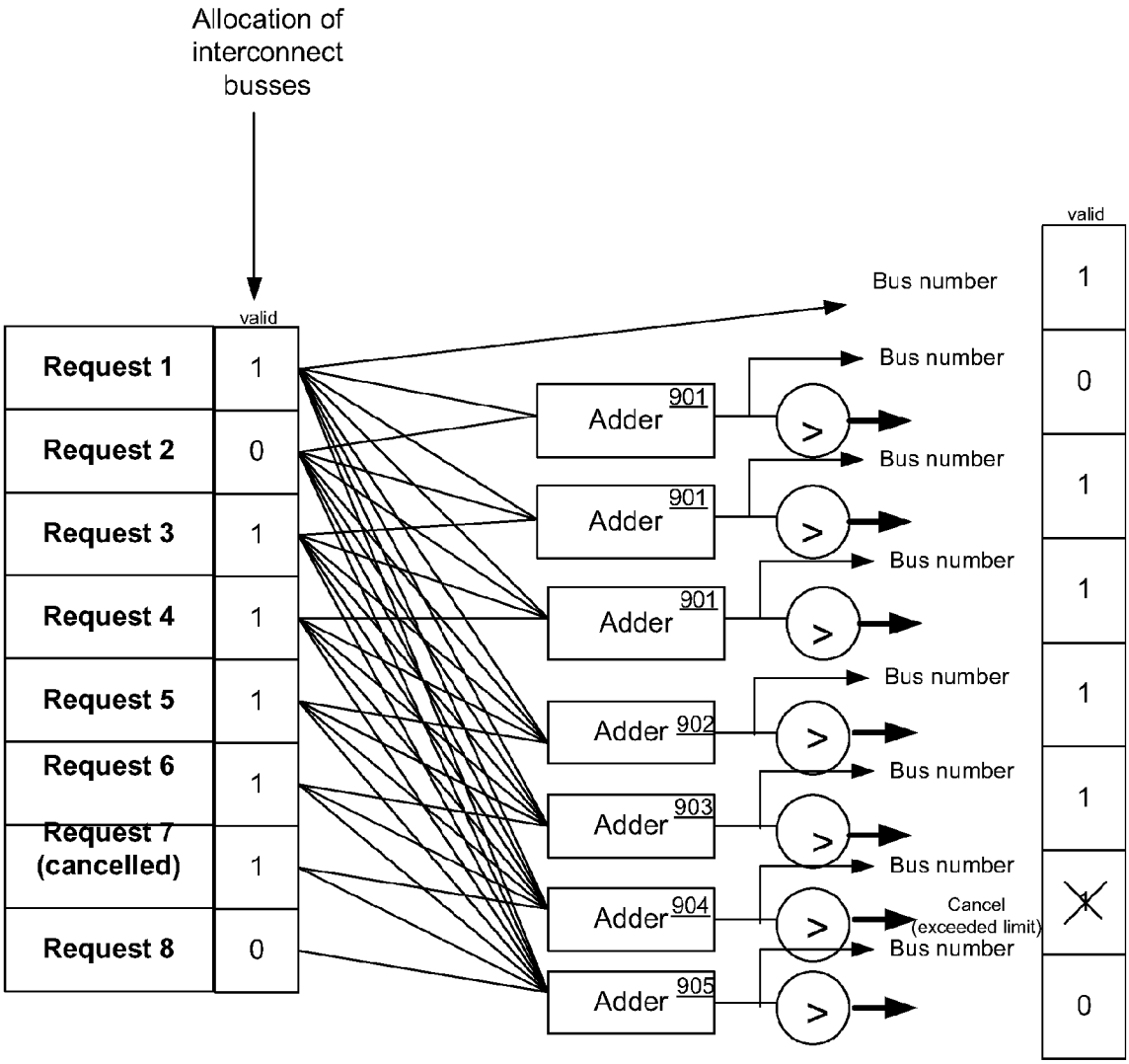
Allocation of interconnect busses
valid
Request 1
Request 2
Request 3
Request 4
Request 5
Request 6
Request 7 (cancelled)
Request 8
1
0
1
1
1
1
1
0
Adder 901
Adder 901
Adder 901
Adder 902
Adder 903
Adder 904
Adder 905
>
Bus number
Bus number
Bus number
Bus number
Bus number
Bus number
Bus number
Cancel (exceeded limit)
valid
1
0
1
1
1
1
1
0

FIG. 13

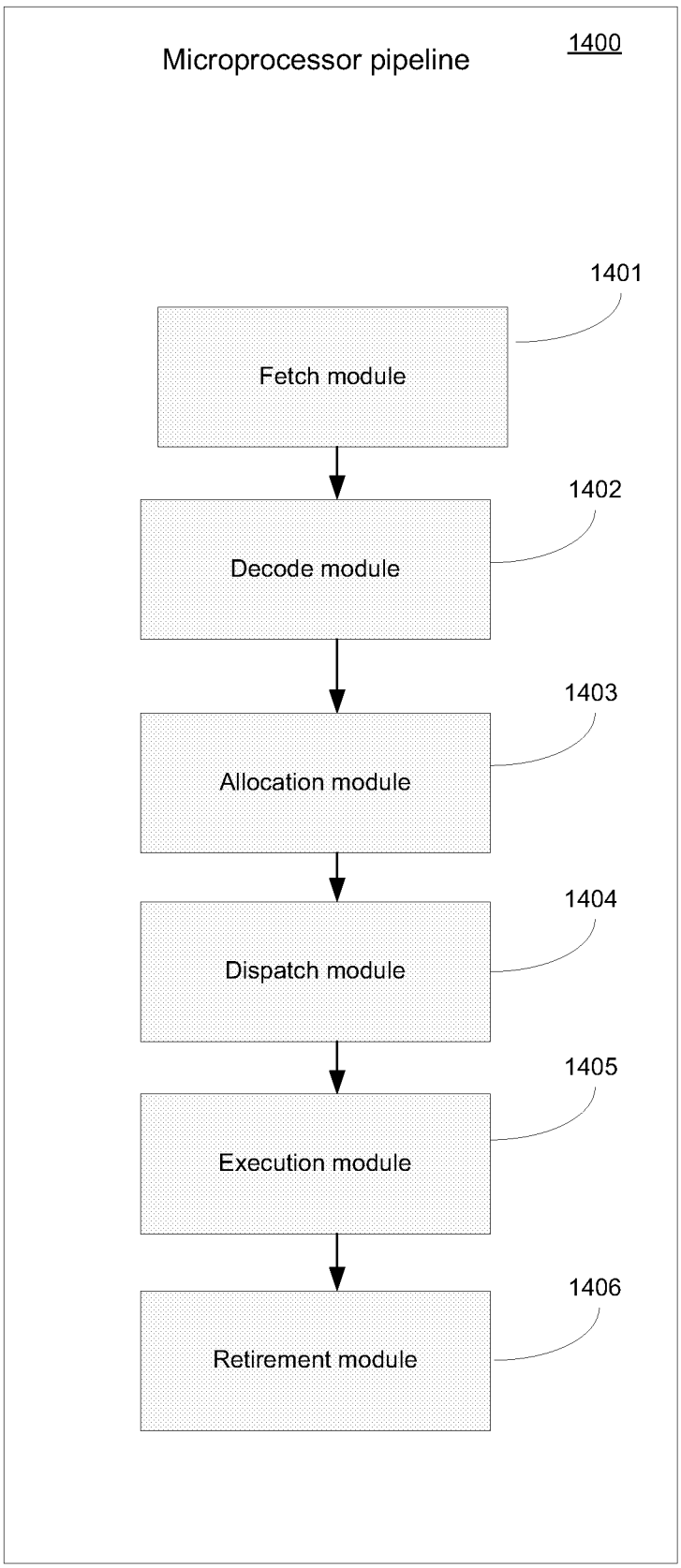
Microprocessor pipeline
1400
1401
Fetch module
1402
Decode module
1403
Allocation module
1404
Dispatch module
1405
Execution module
1406
Retirement module

FIG. 14

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2012/038713

A. CLASSIFICATION OF SUBJECT MATTER

G06F 13/14(2006.01)i, G06F 13/38(2006.01)i, G06F 9/06(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)

IPC : G06F

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)

multi resources, multi consumers, multi engines

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2006-0143390 A1 (SAILESH KOTTAPALLI) 29 June 2006 See the paragraphs [0020]-[0030] and figure 2.	1-29
A	US 2005-0044547 A1 (STEPHAN, KURT GIPP) 24 February 2005 See the paragraphs [0019]-[0025] and figures 1-2.	1-29
A	US 2005-0132145 A1 (GERALD DYBSETTER et al.) 16 June 2005 See the paragraphs [0023]-[0028], claim 1, and figure 1.	1-29
A	US 7150021 B1 (VARAPRASAD VAJJHALA et al.) 12 December 2006 See the column 13, line 63- column 16, line 39, claim 1, and figure 14.	1-29

☐ 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 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
26 OCTOBER 2012 (26.10.2012)	**29 OCTOBER 2012 (29.10.2012)**

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

lee Jung Eun

Telephone No. 82-42-481-5391

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2012/038713

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2006-0143390 A1	29.06.2006	CN 1848095 A0	18.10.2006
		US 7996644 B2	09.08.2011
US 2005-0044547 A1	24.02.2005	EP 1656614 A2	17.05.2006
		US 8296771 B2	23.10.2012
		WO 2005-017763 A2	24.02.2005
US 2005-0132145 A1	16.06.2005	None	
US 7150021 B1	12.12.2006	None	