(12) UK Patent Application (19) GB (11) 2520942 (13) A

(43) Date of A Publication 10.06.2015

(21) Application No: 1321307.9

(22) Date of Filing: 03.12.2013

(51) INT CL:
G06F 12/08 (2006.01)

(56) Documents Cited:
GB 2503437 A US 20090198955 A

(58) Field of Search:
INT CL G06F
Other: EPODOC, WPI, TXTE.

(71) Applicant(s):
International Business Machines Corporation
New Orchard Road, Armonk 10504, New York, United States of America

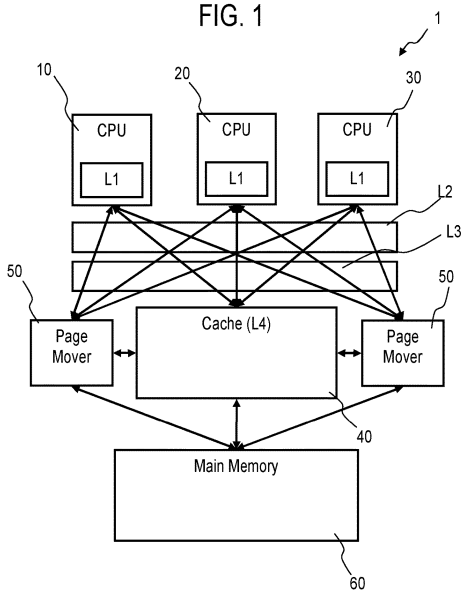
(72) Inventor(s):
Christian Jacobi
Matthias Pflanz
Kai Webber
Stefan Schuh
Jens Dittrich

(74) Agent and/or Address for Service:
IBM United Kingdom Limited
Intellectual Property Law, Hursley Park, WINCHESTER, Hampshire, SO21 2JN, United Kingdom

(54) Title of the Invention: **Data Processing system and method for data processing in a multiple processor system**
Abstract Title: **Multi-processor cache system with a page mover including a data processing engine.**

(57) Disclosed is a multi-processor system 1 with a multi-level cache L1, L2, L3, L4 structure between the processors 10, 20, 30 and the main memory 60. The memories of at least one of the cache levels is shared between the processors. A page mover 50 is positioned closer to the main memory and is connected to the cache memories of the shared cache level, to the main memory and to the processors. In response to a request from a processor the page mover fetches data of a storage area line-wise from one of the shared cache memories or the main memory, while maintaining cache memory access coherency. The page mover has a data processing engine that performs aggregation and filtering of the fetched data. The page mover moves processed data to the cache memories, the main memory or the requesting processor. The data processing engine may have a filter engine that filters data by comparing all elements of a fetched line from a source address of a fetched line from a source address of the shared cache level or main memory with filter arguments to create a bitmask buffer of the target storage area.

FIG. 1



GB 2520942 A

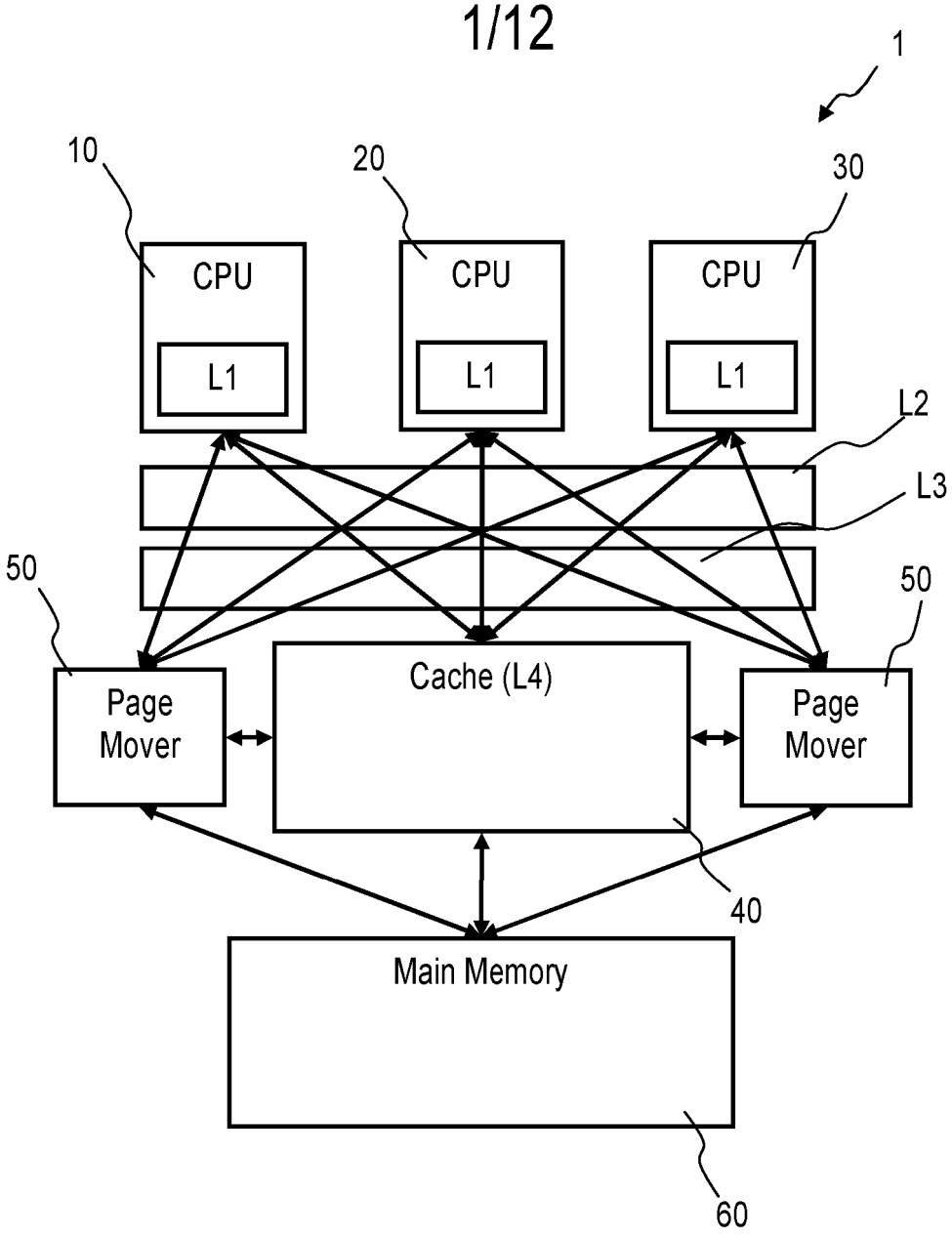

1
10
20
30
CPU
L1
CPU
L1
CPU
L1
L2
L3
50
50
Page Mover
Cache (L4)
Page Mover
40
Main Memory
60

FIG. 1

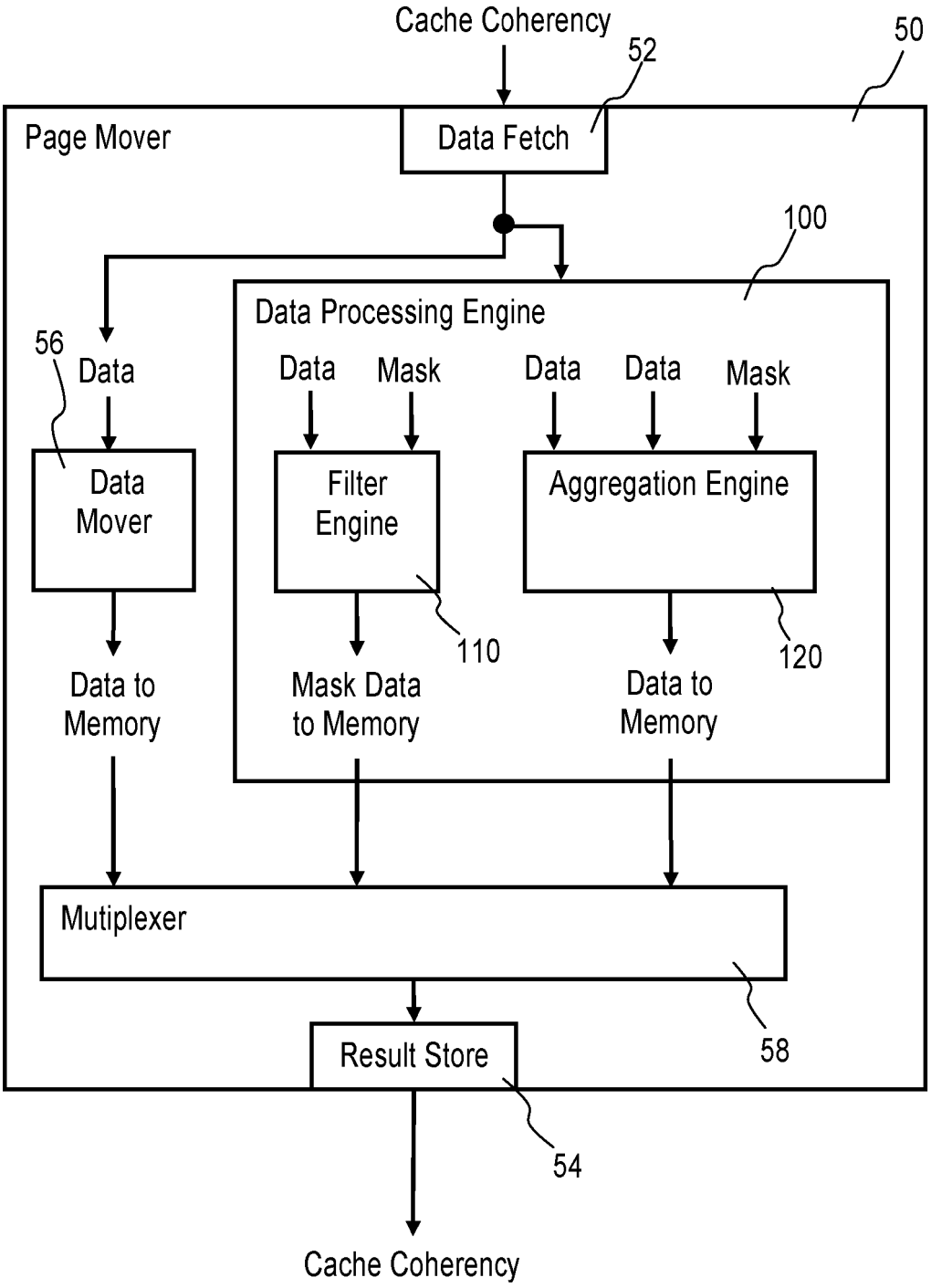

Cache Coherency
52
50
Page Mover
Data Fetch
100
Data Processing Engine
56
Data
Data
Mask
Data
Data
Mask
Data Mover
Filter Engine
Aggregation Engine
110
120
Data to Memory
Mask Data to Memory
Data to Memory
Mutiplexer
58
Result Store
54
Cache Coherency

FIG. 2

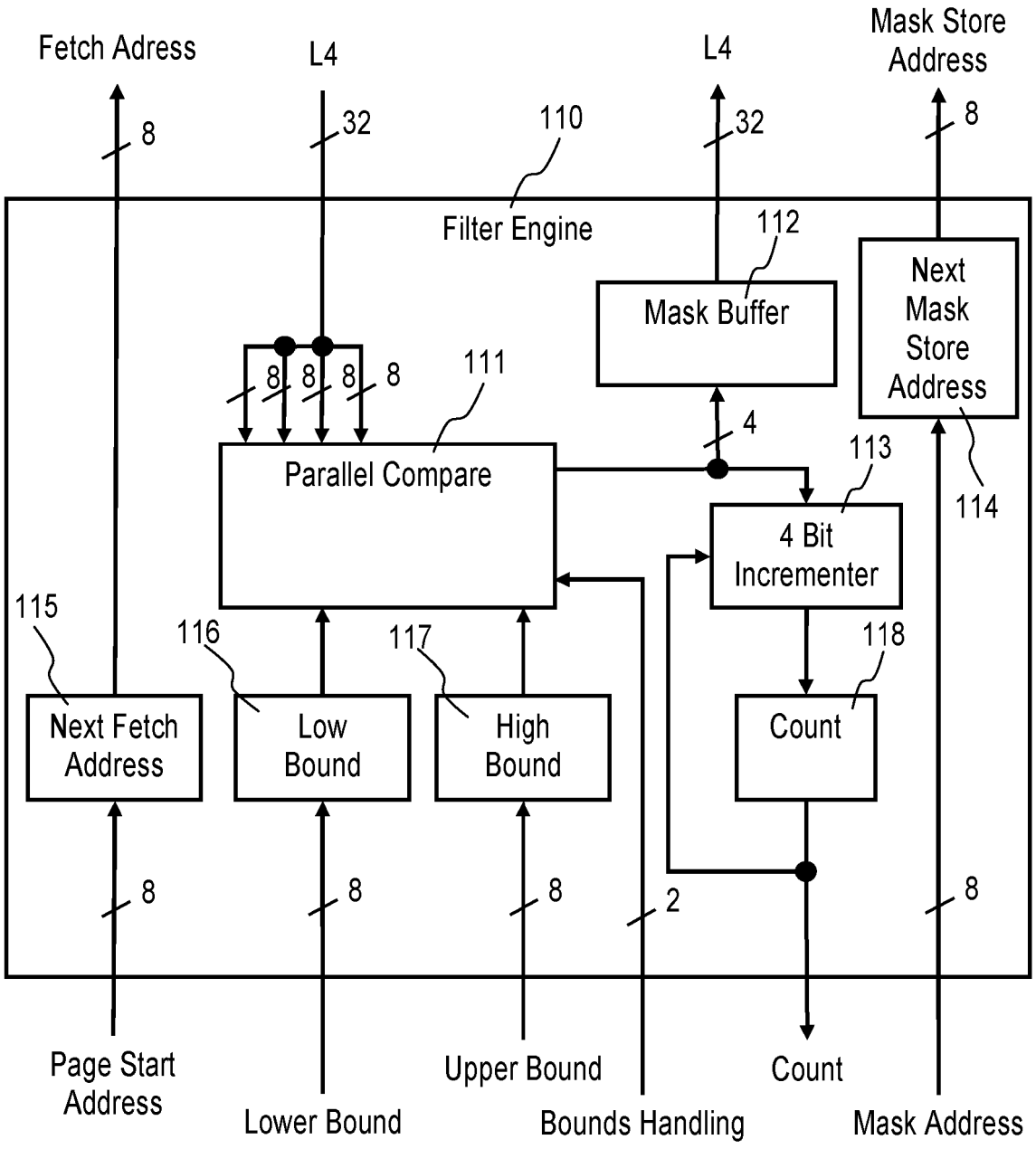

Fetch Adress
L4
L4
Mask Store Address
8
32
110
32
8
Filter Engine
112
Mask Buffer
Next Mask Store Address
8 8 8 8
111
4
Parallel Compare
113
114
4 Bit Incrementer
115
116
117
118
Next Fetch Address
Low Bound
High Bound
Count
8
8
8
2
8
Page Start Address
Lower Bound
Upper Bound
Count
Bounds Handling
Mask Address

FIG. 3

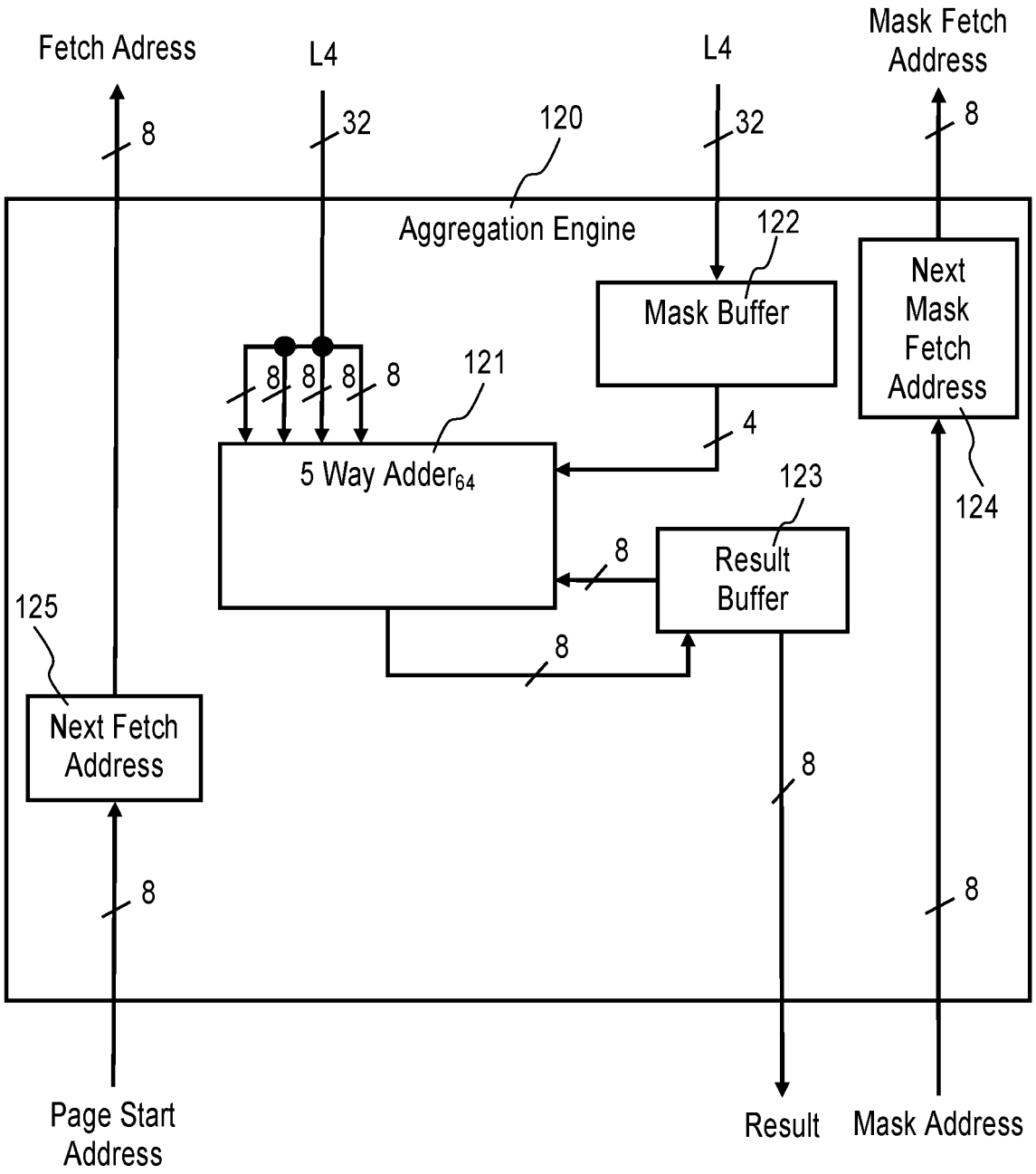

Fetch Adress
L4
L4
Mask Fetch Address
8
32
120
32
8
Aggregation Engine
122
Mask Buffer
Next Mask Fetch Address
8 8 8 8
121
4
5 Way Adder64
123
124
8
Result Buffer
125
8
Next Fetch Address
8
8
8
Page Start Address
Result
Mask Address

FIG. 4

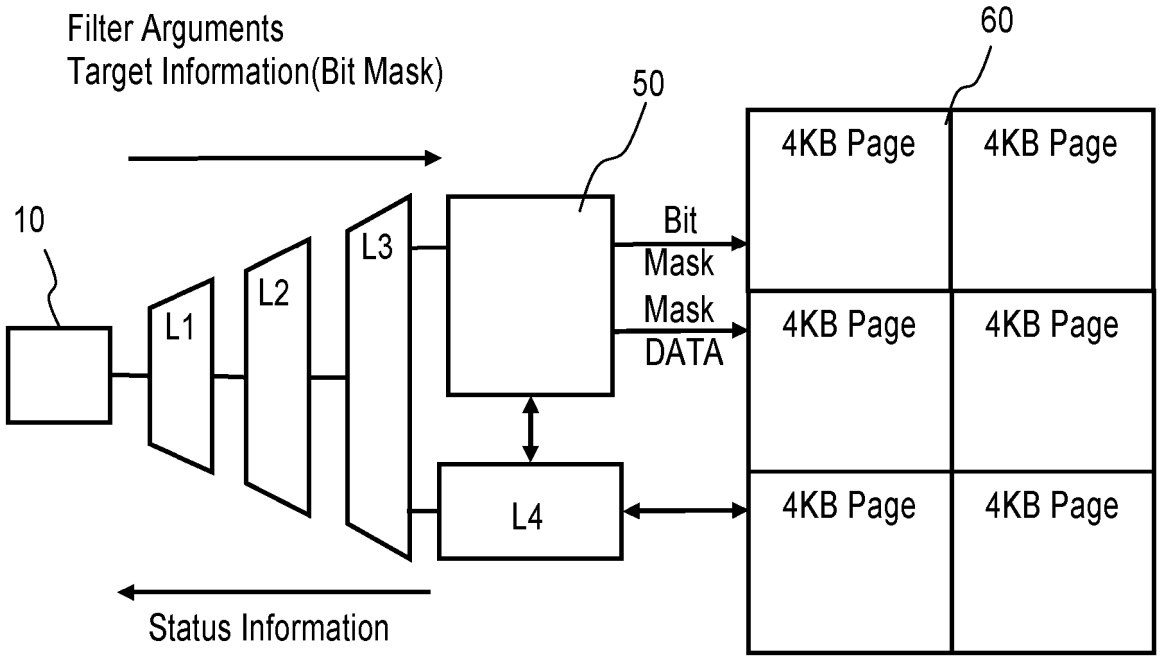

Filter Arguments
Target Information(Bit Mask)
60
50
10
L1
L2
L3
L4
Bit
Mask
Mask
DATA
4KB Page
4KB Page
4KB Page
4KB Page
4KB Page
4KB Page
Status Information

FIG. 5

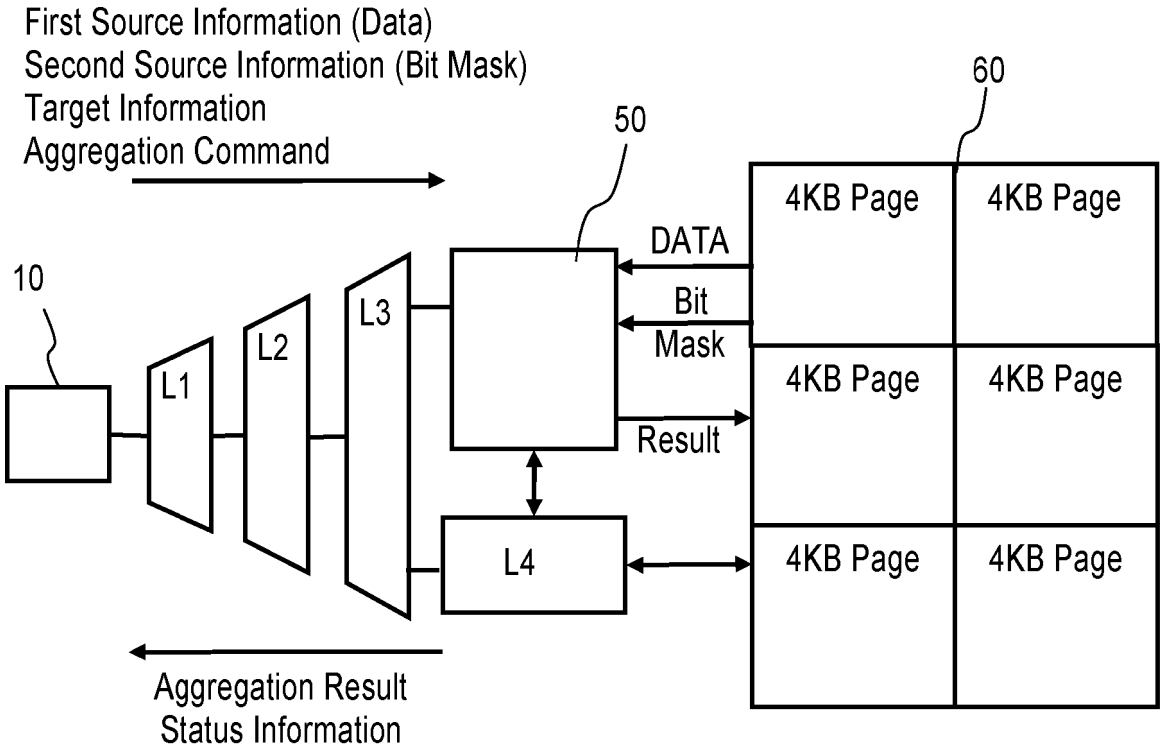

First Source Information (Data)
Second Source Information (Bit Mask)
Target Information
Aggregation Command
60
50
10
L1
L2
L3
L4
DATA
Bit
Mask
Result
4KB Page
4KB Page
4KB Page
4KB Page
4KB Page
4KB Page
Aggregation Result
Status Information

FIG. 6

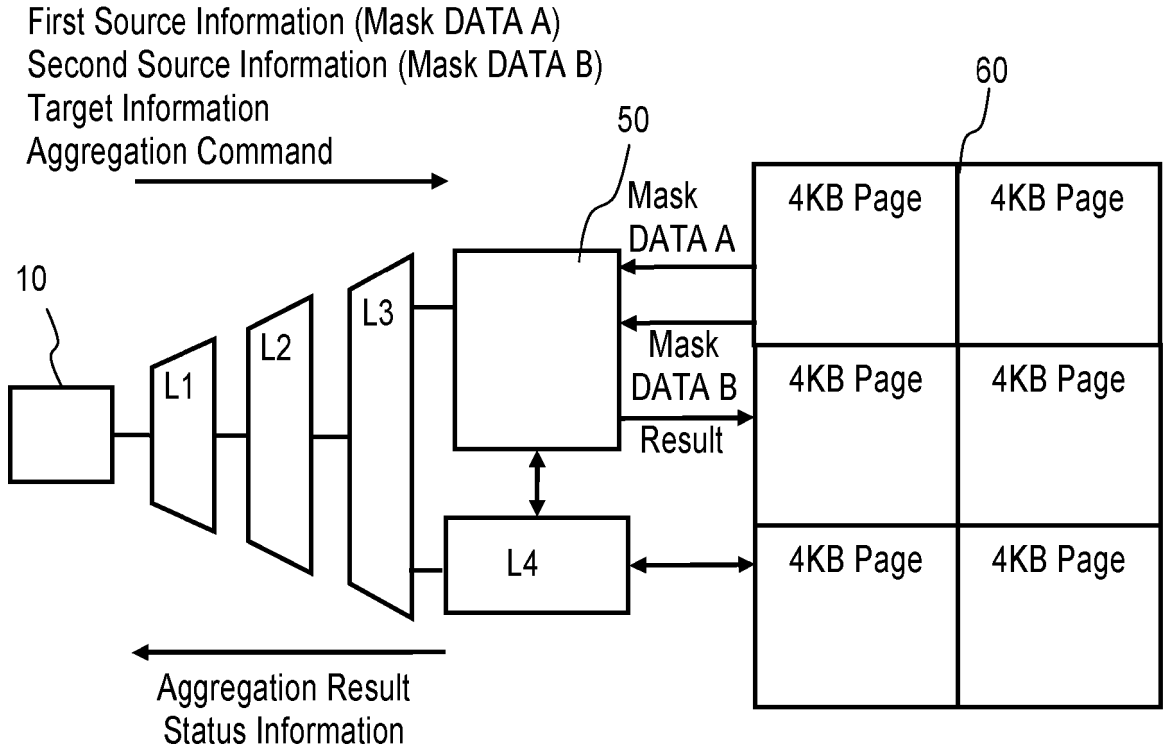

First Source Information (Mask DATA A)
Second Source Information (Mask DATA B)
Target Information
Aggregation Command
50
60
Mask
DATA A
Mask
DATA B
Result
10
L1
L2
L3
L4
4KB Page
4KB Page
4KB Page
4KB Page
4KB Page
4KB Page
Aggregation Result
Status Information

FIG. 7

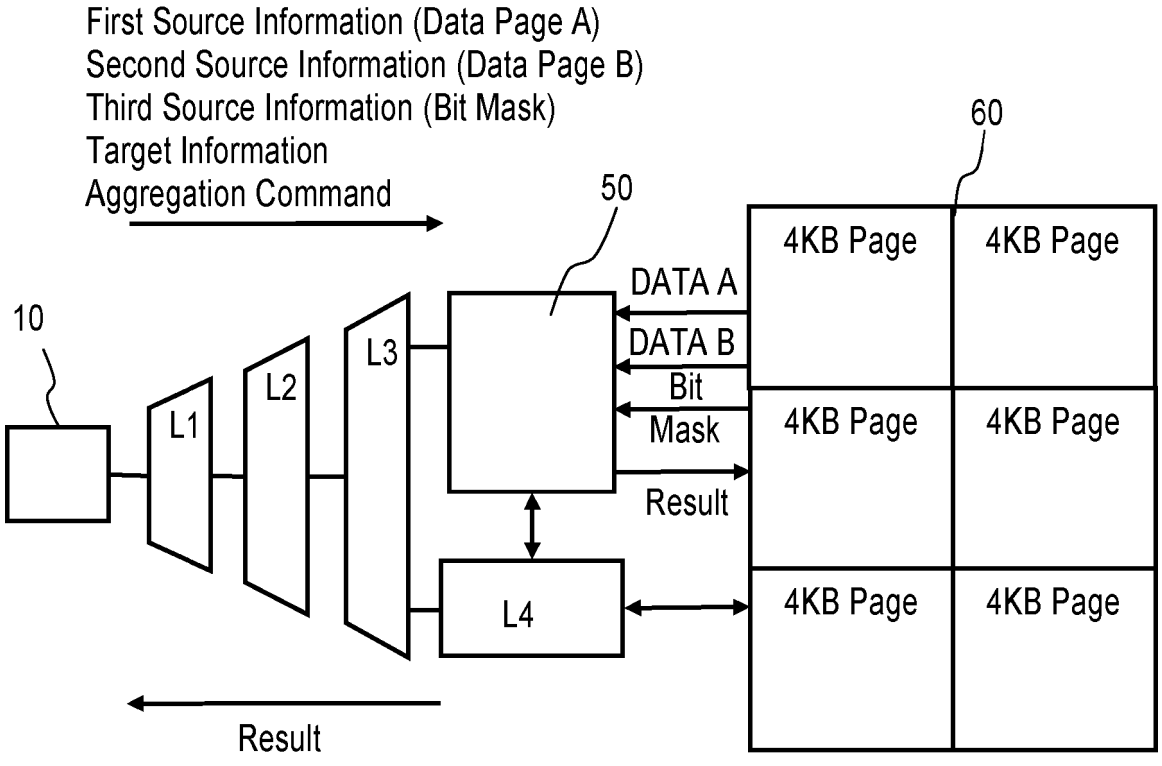

First Source Information (Data Page A)
Second Source Information (Data Page B)
Third Source Information (Bit Mask)
Target Information
Aggregation Command
50
60
DATA A
DATA B
Bit
Mask
Result
10
L1
L2
L3
L4
4KB Page
4KB Page
4KB Page
4KB Page
4KB Page
4KB Page
Result

FIG. 8

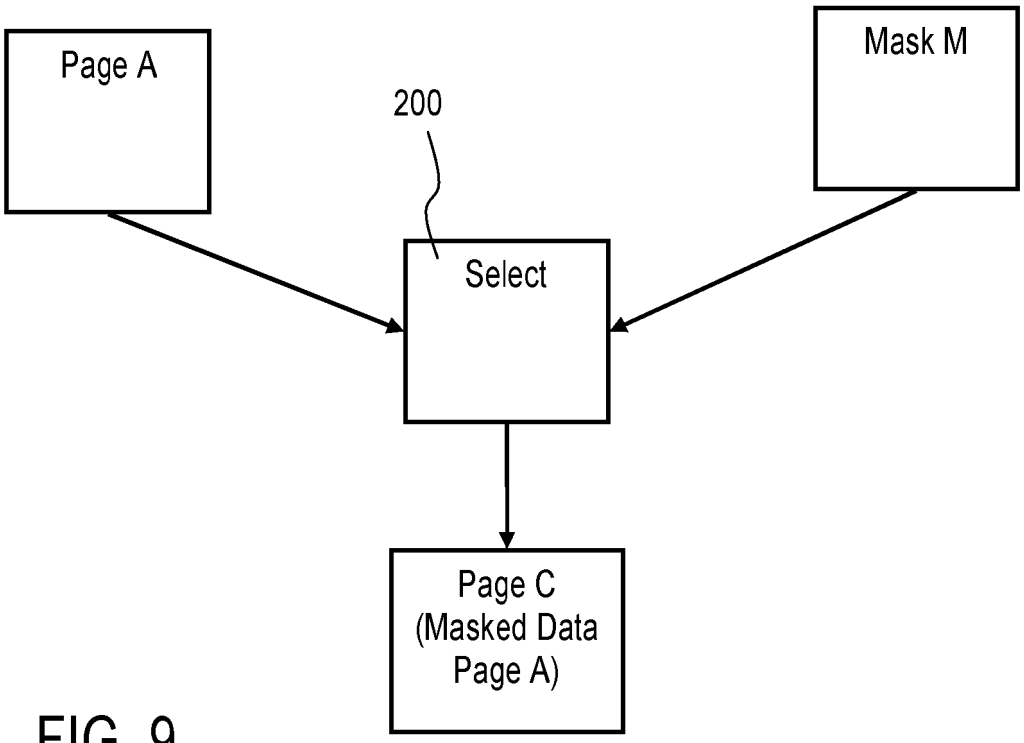

Page A
Mask M
200
Select
Page C
(Masked Data
Page A)

FIG. 9

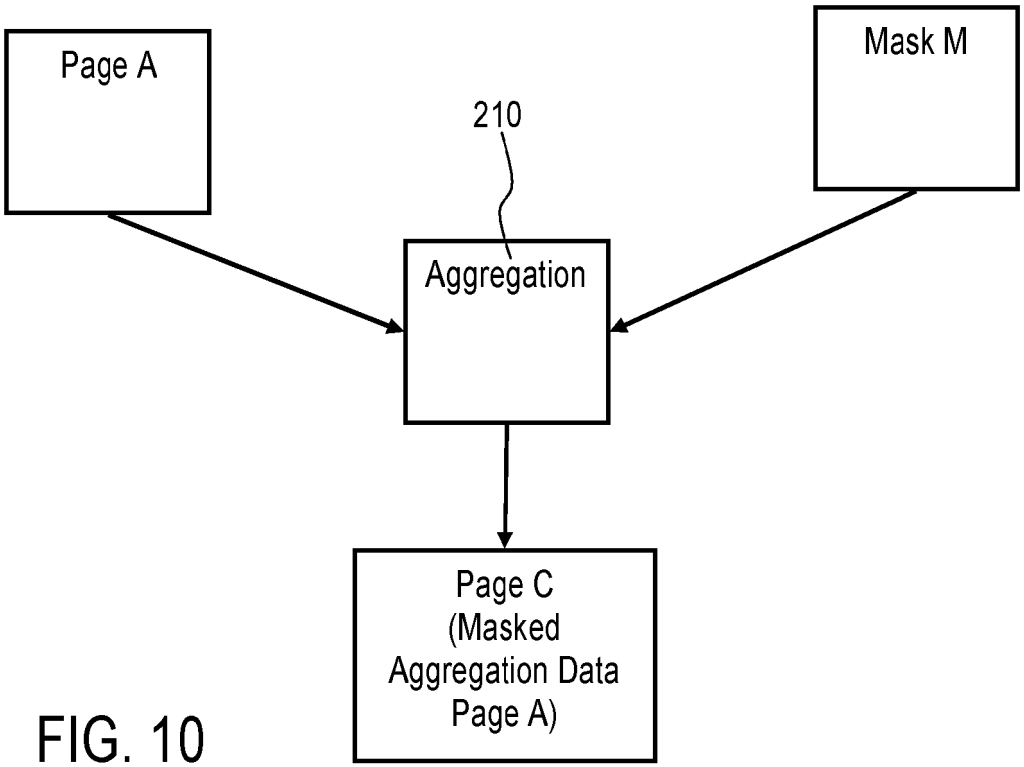

Page A
Mask M
210
Aggregation
Page C
(Masked
Aggregation Data
Page A)

FIG. 10

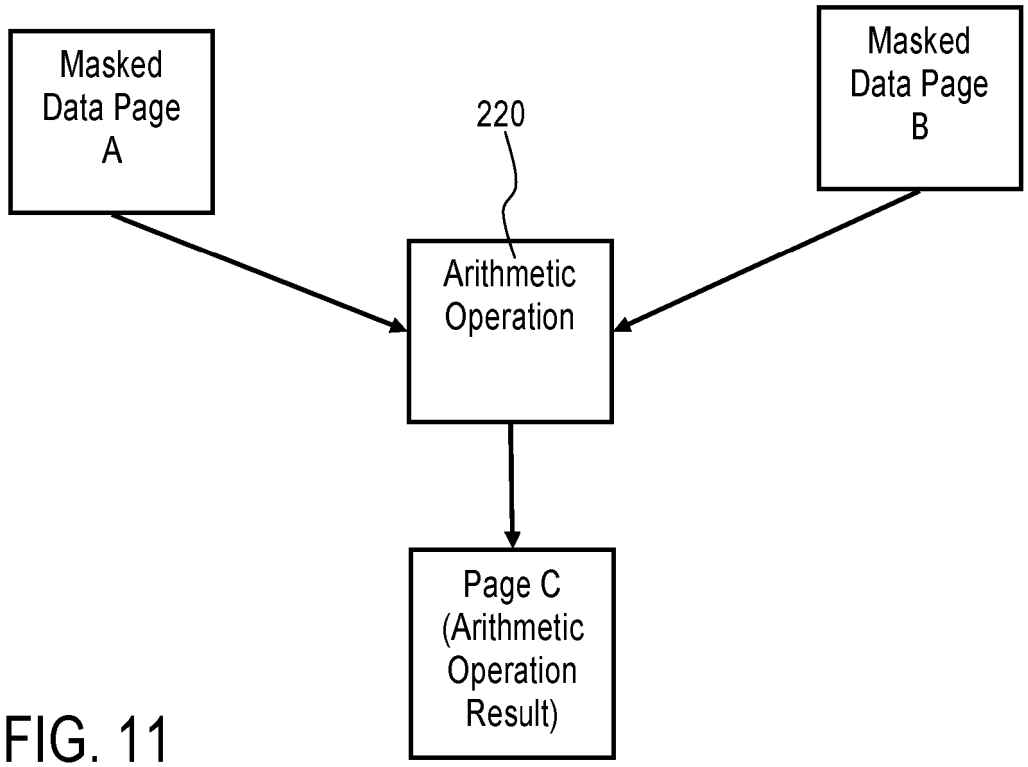

Masked
Data Page
A
Masked
Data Page
B
220
Arithmetic
Operation
Page C
(Arithmetic
Operation
Result)

FIG. 11

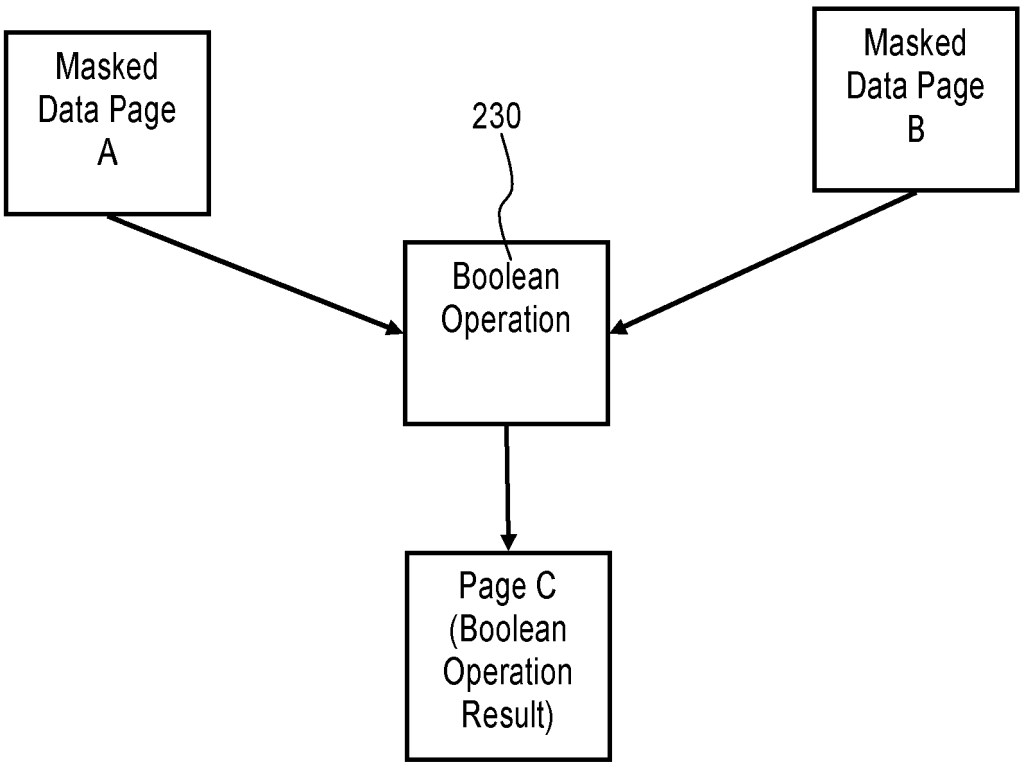

Masked
Data Page
A
Masked
Data Page
B
230
Boolean
Operation
Page C
(Boolean
Operation
Result)

FIG. 12

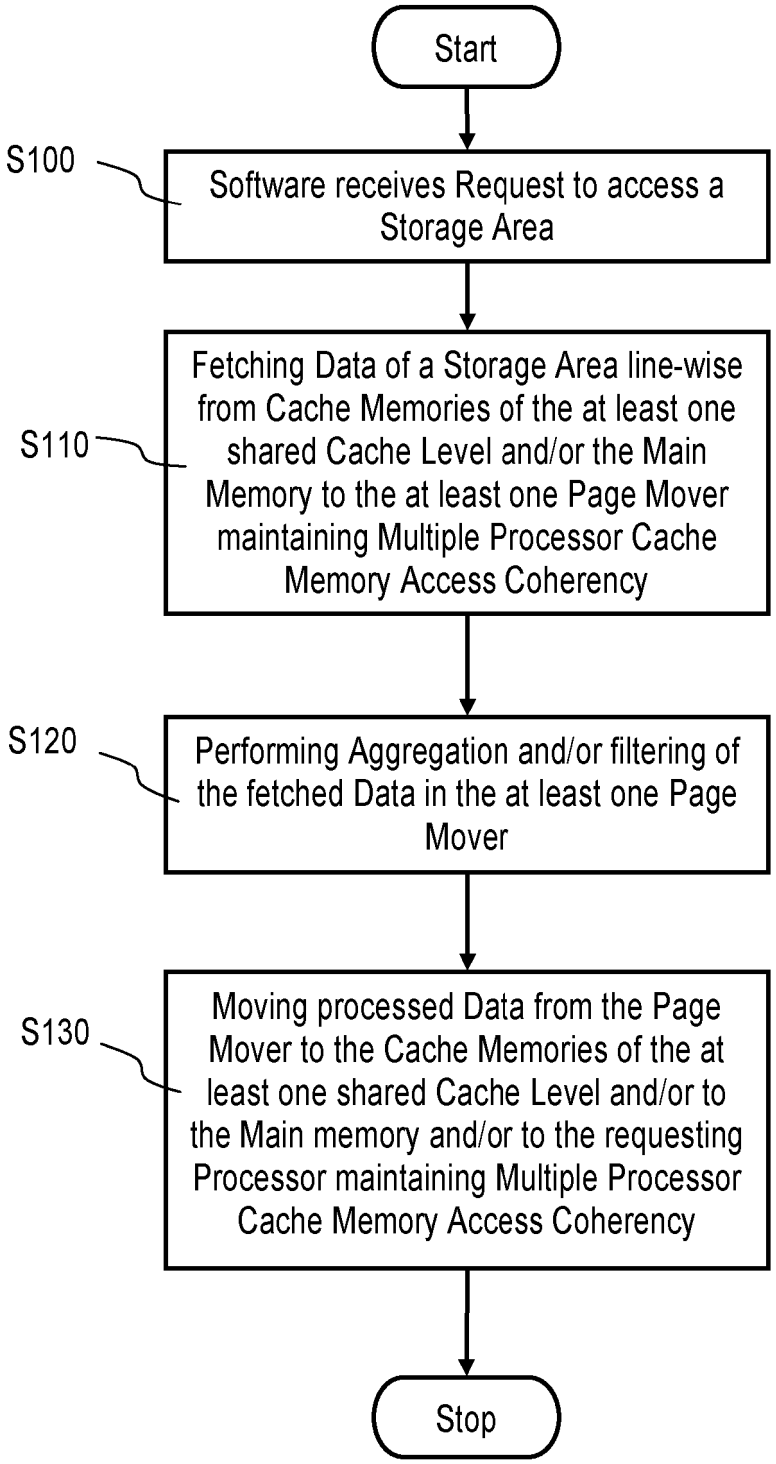

Start
S100
Software receives Request to access a Storage Area
Fetching Data of a Storage Area line-wise from Cache Memories of the at least one shared Cache Level and/or the Main Memory to the at least one Page Mover maintaining Multiple Processor Cache Memory Access Coherency
S110
S120
Performing Aggregation and/or filtering of the fetched Data in the at least one Page Mover
S130
Moving processed Data from the Page Mover to the Cache Memories of the at least one shared Cache Level and/or to the Main memory and/or to the requesting Processor maintaining Multiple Processor Cache Memory Access Coherency
Stop

FIG. 13

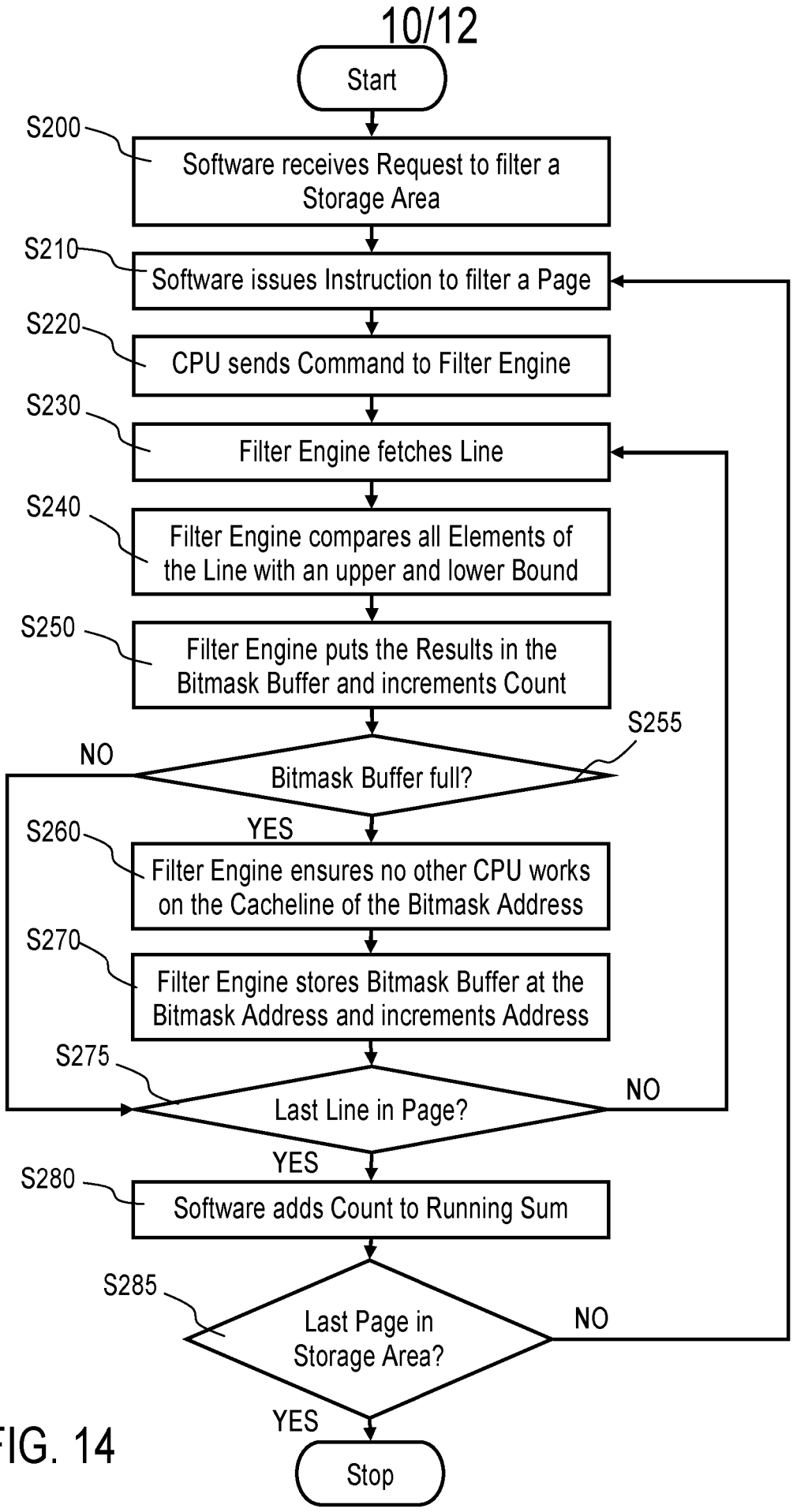

Start
S200
Software receives Request to filter a Storage Area
S210
Software issues Instruction to filter a Page
S220
CPU sends Command to Filter Engine
S230
Filter Engine fetches Line
S240
Filter Engine compares all Elements of the Line with an upper and lower Bound
S250
Filter Engine puts the Results in the Bitmask Buffer and increments Count
S255
Bitmask Buffer full?
NO
YES
S260
Filter Engine ensures no other CPU works on the Cacheline of the Bitmask Address
S270
Filter Engine stores Bitmask Buffer at the Bitmask Address and increments Address
S275
Last Line in Page?
NO
YES
S280
Software adds Count to Running Sum
S285
Last Page in Storage Area?
NO
YES
Stop

FIG. 14

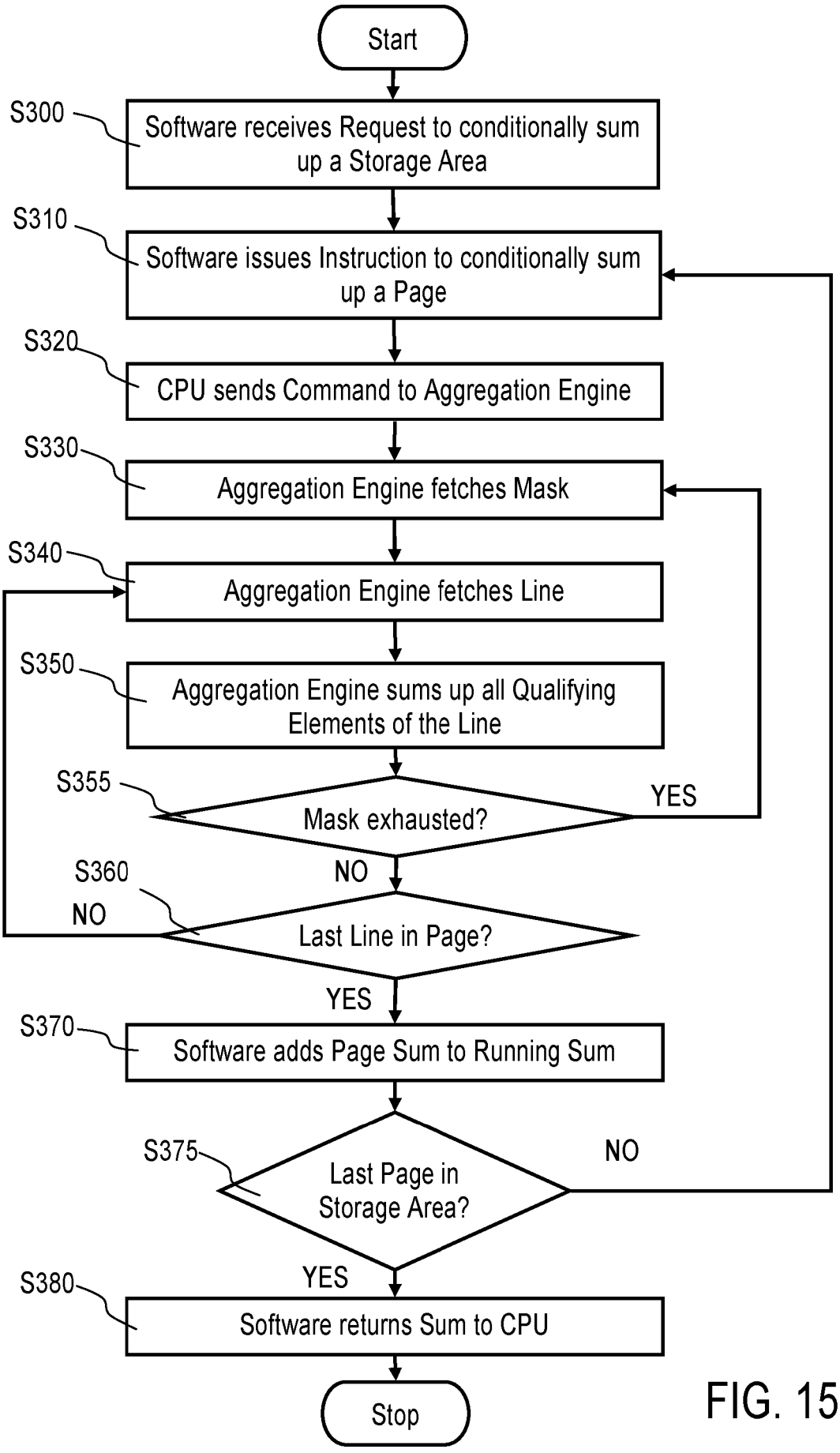

Start
S300
Software receives Request to conditionally sum up a Storage Area
S310
Software issues Instruction to conditionally sum up a Page
S320
CPU sends Command to Aggregation Engine
S330
Aggregation Engine fetches Mask
S340
Aggregation Engine fetches Line
S350
Aggregation Engine sums up all Qualifying Elements of the Line
S355
Mask exhausted?
YES
NO
S360
Last Line in Page?
NO
YES
S370
Software adds Page Sum to Running Sum
S375
Last Page in Storage Area?
NO
YES
S380
Software returns Sum to CPU
Stop

FIG. 15

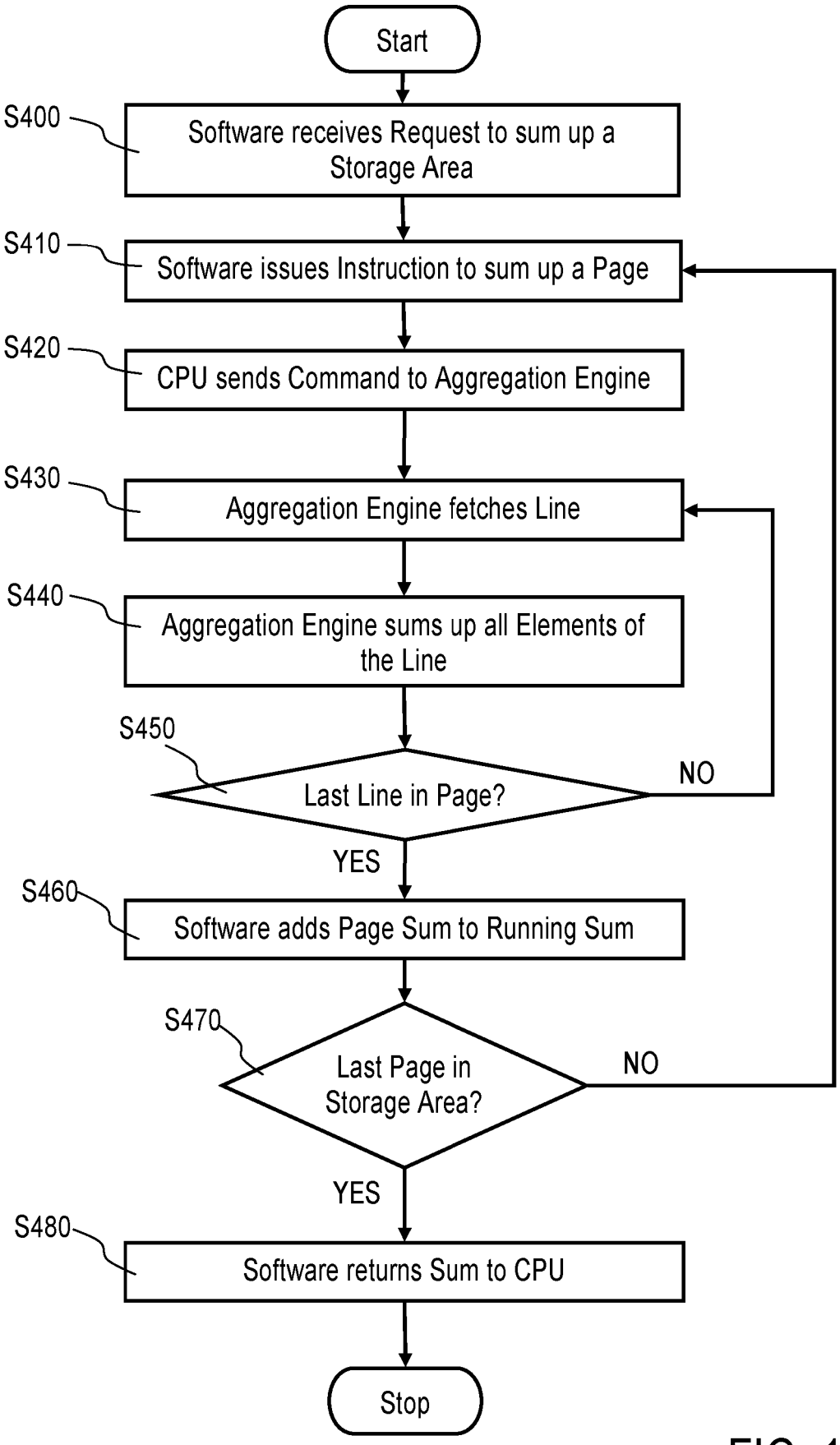

Start
S400
Software receives Request to sum up a Storage Area
S410
Software issues Instruction to sum up a Page
S420
CPU sends Command to Aggregation Engine
S430
Aggregation Engine fetches Line
S440
Aggregation Engine sums up all Elements of the Line
S450
Last Line in Page?
NO
YES
S460
Software adds Page Sum to Running Sum
S470
Last Page in Storage Area?
NO
YES
S480
Software returns Sum to CPU
Stop

FIG. 16

D E S C R I P T I O N

DATA PROCESSING SYSTEM AND METHOD FOR DATA PROCESSING IN A MULTIPLE PROCESSOR SYSTEM

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates in general to the field of multiprocessor system with hierarchical cache structure, and in particular to a data processing system comprising multiple processors with a hierarchical cache structure comprising multiple levels of cache between the processors and a main memory; and a method for data processing in a multiple processor system. Still more particularly, the present invention relates to a data processing program and a computer program product for data processing in a multiple processor system.

Description of the Related Art

Known data processing systems comprising multiple processors with hierarchical processor caches and a main memory share at least one processor cache between the processors. In such data processing systems the bandwidth between memory and processors is the new bottleneck. To increase the payload the known data processing systems comprise page mover functionality for moving data blocks from one memory location to another memory location without involving the corresponding processor that has initiated the data moving process.

In the Patent Application Publication US 2011/0320730 A1 “NON-BLOCKING DATA MOVE DESIGN” by Blake et al. a mechanism for data buffering is disclosed. A portion of a cache is allocated as buffer regions, and another portion of the cache is designated

as random access memory. One of the buffer regions is assigned to a processor. A data block is stored form the one of the buffer regions of the cache to the memory.

Summary of the Invention

The technical problem underlying the present invention is to provide a data processing system and a method for data processing in a multiple processor system, which are able to increase the payload of the system.

According to the present invention this problem is solved by providing a data processing system having the features of claim 1, a method for data processing in a multiple processor system having the features of claim 6, a data processing program for data processing in a multiple processor system having the features of claim 14, and a computer program product for data processing in a multiple processor system having the features of claim 15. Advantageous embodiments of the present invention are mentioned in the subclaims.

Accordingly, in an embodiment of the present invention a data processing system comprises multiple processors with a hierarchical cache structure comprising multiple levels of cache and a main memory. At least cache memories of one cache level are shared between the processors. Further, at least one page mover is positioned closer to the main memory and connected to the cache memories of the at least one shared cache level, the main memory and to the multiple processors to move data between the cache memories of the at least one shared cache level, the main memory and the processors. In response to a request from one of the processors the at least one page mover fetches data of a storage area line-wise from at least one of the following memories: The cache memories of the at least one shared cache level and the main memory maintaining multiple processor cache

memory access coherency; wherein the at least one page mover comprises a data processing engine which performs at least one of the following data processing operations: Aggregation and filtering of the fetched data. The page mover moves processed data to at least one of the following components: Cache memories of the at least one shared cache level, the main memory and the requesting processor maintaining multiple processor cache memory access coherency.

In further embodiments of the present invention, the data processing engine comprises at least one filter engine filtering data of a storage area line-wise by comparing all elements of a fetched line from a source address of the at least one shared cache level and/or the main memory with filter arguments to create a bitmask, and writing comparison results as bitmask data in a bitmask buffer of a target storage area located at a target address of the at least one shared cache level and/or the main memory based on a corresponding request from one of the processors containing a filter command with the filter arguments and source and target information.

In further embodiments of the present invention, the data processing engine comprises at least one filter engine moving data of a storage area corresponding with bitmask data of the bitmask buffer line-wise from a source address to a target address to create a bitmask data set based on a corresponding request from one of the processors containing a move command and bitmask and source and target information.

In further embodiments of the present invention, the data processing engine comprises at least one aggregation engine performing arithmetic or Boolean operations with data of at least one storage area fetched from a corresponding source address having a corresponding bitmask data set and sending a data processing result to a storage area at a target address of

the at least one shared cache level and/or the main memory or to a requesting processor based on a corresponding request from one of the processors containing a aggregation command and bitmask and source and target information.

In further embodiments of the present invention, at least one aggregation engine performs arithmetic or Boolean operations with data of a first storage area fetched from corresponding source addresses having a corresponding first bitmask data set, and data of a second storage area fetched from corresponding source addresses having the corresponding bitmask set, and sends a data processing result to a storage area at target addresses of the at least one shared cache level and/or the main memory or to a requesting processor based on a corresponding request from one of the processors containing an aggregation command and bitmask and source and target information.

In another embodiment of the present invention, a method for data processing in a multiple processor system with a hierarchical cache structure comprising multiple levels of cache between the processors and a main memory, wherein at least cache memories of one cache level are shared between the processors and at least one page mover is positioned closer to the main memory and connected to the cache memories of the at least one shared cache level, the main memory and to the multiple processors to move data between the cache memories of the at least one shared cache level, the main memory and the processors; wherein in response to a request from one of the processors the method performs the steps of: Fetching data of a storage area line-wise from at least one of the following memories: The cache memories of the at least one shared cache level and the main memory; to the at least one page mover maintaining multiple processor cache memory access coherency; performing at least one of the following data processing operations in the at least one page mover: Aggregation and

filtering of the fetched data; and moving processed data from the page mover to at least one of the following components: Cache memories of the at least one shared cache level, the main memory and the requesting processor (10, 20, 30) maintaining multiple processor cache memory access coherency.

In further embodiments of the present invention, data of a storage area is moved line-wise from a source address of the at least one shared cache level and/or the main memory to a target address of the at least one shared cache level and/or the main memory based on a corresponding request from one of the processors containing a move command and source and target information.

In further embodiments of the present invention, based on a corresponding request from one of the processors containing a filter command with the filter arguments and source and target information data of a storage area are filtered line-wise by comparing all elements of a fetched line from a source address of the at least one shared cache level and/or the main memory with filter arguments, wherein comparison results are written in a bitmask buffer located at a target address of the at least one shared cache level and/or the main memory.

In further embodiments of the present invention, based on a corresponding request from one of the processors containing a aggregation command and source and target information arithmetic or Boolean operations are performed with data of at least one storage area fetched from a corresponding source address of the at least one shared cache level and/or the main memory, wherein a data processing result is send to a storage area at a target address of the at least one shared cache level and/or the main memory or to a requesting processor.

In further embodiments of the present invention, at least one aggregation engine performs arithmetic or Boolean operations with data of a first storage area and a masked or unmasked second storage area fetched from corresponding source addresses of the at least one shared cache level and/or the main memory.

In another embodiment of the present invention, a data processing program for execution in a data processing system comprises software code portions for performing a method for data processing in a multiple processor system when the program is run on the data processing system.

In yet another embodiment of the present invention, a computer program product stored on a computer-usable medium, comprises computer-readable program means for causing a computer to perform a method for data processing in a multiple processor system when the program is run on the computer.

All in all, embodiments of the present invention are focused on a page mover functionality comprising a data processing engine connected to the shared processor cache and each processor to aggregate and/or filter data from the shared processor cache in response to a request from one of the processors.

The key of the innovation is to perform specific operations like filtering and/or aggregation operations closer to the memory. This is suitable to increase payload for online analytical processing (OLAP) in business environments.

Embodiments of the present invention interpret a page of memory data as a vector of 8, 16, 32 or 64 bit scalars, for example; and aggregate (sum, min, max) such a vector and return the result in a register. Further embodiments of the present invention create bit masks (filters) by comparing vector

elements against register contents passed to the page mover functionality.

Embodiments of the present invention read and interpret a page of memory as a vector of 8, 16, 32 or 64 bit scalars, read one or two 256 bytes cache lines and interpret them as a bit vector, and aggregate (sum, min, max) data under mask and return the result in a register

The above, as well as additional purposes, features, and advantages of the present invention will become apparent in the following detailed written description.

Brief Description of the Drawings

Preferred embodiments of the present invention, as described in detail below, are shown in the drawings, in which

FIG. 1 is a schematic block diagram of a data processing system, in accordance with an embodiment of the present invention;

FIG. 2 is a schematic block diagram of a page mover used in the data processing system shown in FIG. 1;

FIG. 3 is a schematic block diagram of a filter engine used in the page mover shown in FIG. 2;

FIG. 4 is a schematic block diagram of an aggregation engine used in the page mover shown in FIG. 2;

FIG. 5 to 8 are schematic block diagrams of the data processing system shown in FIG. 1, performing different data filtering and data aggregation operations, in accordance with an embodiment of the present invention;

FIG. 9 to 12 are schematic block diagrams showing different implementations of data filtering and data aggregation operations performed by the page mover shown in FIG. 2, in accordance with an embodiment of the present invention;

FIG. 13 is a schematic flow diagram of a method for data processing in a multiple processor system, in accordance with an embodiment of the present invention;

FIG. 14 is a schematic flow diagram of data filtering operation performed by the page mover shown in FIG. 2, in accordance with an embodiment of the present invention;

FIG. 15 is a more detailed flow diagram of a masked summing operation performed by the page mover shown in FIG. 2, in accordance with an embodiment of the present invention; and

FIG. 16 is a more detailed flow diagram of a summing operation performed by the page mover shown in FIG. 2, in accordance with an embodiment of the present invention.

Detailed Description of the Preferred Embodiments

As will be appreciated by one skilled in the art, aspects of the present invention may be embodied as a system, method or computer program product. Accordingly, aspects of the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a "circuit," "module" or "system." Furthermore, aspects of the present invention may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied thereon.

Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the context of this document, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device.

A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device.

Program code embodied on a computer readable medium may be transmitted using any appropriate medium, including but not

limited to wireless, wireline, optical fiber cable, RF, etc., or any suitable combination of the foregoing.

Computer program code for carrying out operations for aspects of the present invention may be written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

Aspects of the present invention are described below with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

These computer program instructions may also be stored in a computer readable medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer readable medium produce an article of manufacture including instructions which implement the function/act specified in the flowchart and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

FIG. 1 shows a data processing system 1, in accordance with an embodiment of the present invention; FIG. 2 shows a page mover 50 used in the data processing system 1; FIG. 3 shows a filter engine 110 used in the page mover shown in FIG. 2; and FIG. 4 shows an aggregation engine 120 used in the page mover shown in FIG. 2. FIG. 9 to 12 show different implementations of data filtering and data aggregation operations performed by the page mover 50, in accordance with an embodiment of the present invention.

Referring to FIG. 1 to 4, the shown embodiment of the data processing system 1 comprises multiple processors 10, 20, 30 with a hierarchical cache structure comprising multiple levels of cache L1, L2, L3, L4 between the processors 10, 20, 30, a main memory 60 and at least one page mover 50, here two page movers 50 are used. In the shown embodiment four levels of cache L1, L2, L3, L4 are used and cache memories of at a second cache

level L2, a third cache level L3 and a fourth cache level are shared between the processors 10, 20, 30. The at least one page mover 50 is connected to the cache memories of the different shared cache levels L2, L3, L4, to the main memory 60 and to the multiple processors 10, 20, 30 to move data between the cache memories of the shared cache levels L2, L3, L4, the main memory 60 and the processors 10, 20, 30. Each page mover 50 is positioned closer to the main memory 60 as the processors 10, 20, 30. In response to a request from one of the processors 10, 20, 30 the corresponding page mover 50 fetches data of a storage area line-wise from the cache memories of the shared cache levels L2, L3, L4 or the main memory 60 maintaining multiple processor cache memory access coherency. Each page mover 50 comprises a data processing engine 100 which performs at least one of the following data processing operations: Aggregation and filtering of the fetched data; and each page mover 50 moves processed data to the cache memories of the shared cache levels L2, L3, L4 and/or to the main memory 60 and/or to the requesting processor 10, 20, 30 maintaining multiple processor cache memory access coherency.

Referring to FIG. 2, each page mover 50 further comprises a data fetch functionality 52 performing the above mentioned data fetch process, a standard data moving functionality 56 for moving data blocks from one memory location to another memory location, a multiplexer 58 multiplexing the output data of the data processing engine 100 and the standard data moving functionality 56 to a result store functionality 42 performing the above mentioned data store process.

Referring to FIG. 2 to 4, the data processing engine 100 comprises at least one filter engine 110 and at least on aggregation engine 120. The at least one filter engine 110 comprises in the shown embodiment a parallel comparing unit 111, a mask buffer 112, holding data to be written to a corresponding

mask store address, an incrementing block 113, a first address buffer 114 holding the next mask store address, a second address buffer 115 holding the next fetch address, a low bound buffer 116, a high bound buffer 117, and a count buffer 118. The at least one filter engine 110 filters data of a storage area line-wise by comparing all elements of a fetched line from a source address of the at least one shared cache level L2, L3, L4 and/or the main memory 60 with filter arguments to create result, and writing comparison results to a target storage area located at a target address of the at least one shared cache level L2, L3, L4 and/or the main memory 60 based on a corresponding request from one of the processors 10, 20, 30 containing a filter command with the filter arguments and source and target information. The at least one aggregation engine 110 comprises in the shown embodiment an adder 121, a mask buffer 122, holding data read from a corresponding mask fetch address, a result buffer 123, a first address buffer 124 holding the next mask fetch address, and a second address buffer 115 holding the next fetch address. The at least one aggregation engine 110 performs arithmetic or Boolean operations with data of at least one storage area of the cache memories of the shared cache levels L2, L3, L4 and/or the main memory 60; and writes the result to at least one storage area of the cache memories of the shared cache levels L2, L3, L4 and/or to the main memory 60 and/or to the requesting processor 10, 20, 30.

Referring to FIG. 5, the at least one filter engine 110 of the data mover 50 filters data of a storage area line-wise by comparing all elements of a fetched line from a source address of the at least one shared cache level L2, L3, L4 and/or the main memory 60 with filter arguments to create a bitmask (Bit Mask), and writing comparison results as bitmask data (Mask DATA) in a bitmask buffer of a target storage area located at a target address of the at least one shared cache level L2, L3, L4 and/or the main memory 60 based on a corresponding request from

one of the processors 10, 20, 30 containing a filter command with the filter arguments and source and target information. In the shown embodiment, status information is provided additionally to the requesting processor 10.

Referring to FIG. 6, the at least one aggregation engine 120 of the data mover 50 performs aggregation operations with data (DATA) of at least one storage area of the at least one shared cache level L2, L3, L4 and/or the main memory 60 having a corresponding bitmask data set (Bit Mask) fetched from a corresponding source address. The data mover 50 sends the aggregation result (Result) to a storage area at a target address of the at least one shared cache level L2, L3, L4 and/or the main memory 60 or to a requesting processor 10 based on a corresponding request from one of the processors 10, 20, 30 containing a aggregation command and bitmask and source and target information. In the shown embodiment, status information is provided additionally to the requesting processor 10.

Referring to FIG. 7, the at least one aggregation engine 120 of the data mover 50 performs aggregation operations with data (Mask DATA A) of a first storage area of the at least one shared cache level L2, L3, L4 and/or the main memory 60 having a corresponding first bitmask data set fetched from corresponding source addresses, and data (Mask DATA B) of a second storage area of the at least one shared cache level L2, L3, L4 and/or the main memory 60 having the corresponding bitmask set fetched from corresponding source addresses. The data mover 50 sends the data aggregation result (Result) to a storage area at target addresses of the at least one shared cache level L2, L3, L4 and/or the main memory 60 or to a requesting processor 10 based on a corresponding request from one of the processors 10, 20, 30 containing an aggregation command and bitmask and source and target information. In the shown embodiment, status information is provided additionally to the requesting processor 10.

Referring to FIG. 8, the at least one aggregation engine 120 of the data mover 50 performs aggregation operations with data (DATA A) of a first storage area of the at least one shared cache level L2, L3, L4 and/or the main memory 60 fetched from corresponding source addresses, and data (DATA B) of a second storage area of the at least one shared cache level L2, L3, L4 and/or the main memory 60 fetched from corresponding source addresses, according to a corresponding bitmask set (Bit Mask) fetched from corresponding source addresses of a storage area of the at least one shared cache level L2, L3, L4 and/or the main memory 60. The data mover 50 sends the data aggregation result (Result) to a storage area at target addresses of the at least one shared cache level L2, L3, L4 and/or the main memory 60 or to a requesting processor 10 based on a corresponding request from one of the processors 10, 20, 30 containing an aggregation command and bitmask and source and target information. In the shown embodiment, status information is provided additionally to the requesting processor 10.

Referring to FIG. 9, the at least one filter engine 110 of the data mover 50 performs a select and move operation 200 and selects data (Page A) of a storage area corresponding with bitmask data (Mask M) of the bitmask buffer line-wise from a source address and moves the data to a target address (Page C) to create masked data of page A based on a corresponding request from one of the processors 10, 20, 30 containing a move command and bitmask and source and target information.

Referring to FIG. 10, the at least one aggregation engine 110 of the data mover 50 aggregates data (Page A) of a storage area from a source address according to bitmask data (Mask M) of the bitmask buffer line-wise to a target address (Page C) to create masked aggregation data of page A based on a corresponding request from one of the processors 10, 20, 30 containing a move

command, an aggregation command and bitmask and source and target information.

Referring to FIG. 11, the at least one aggregation engine 110 of the data mover 50 performs line-wise arithmetic operations 220 with masked data (Masked Data Page A) of a first page from a source address of a storage area and with masked data (Masked Data Page B) of a second page from a source address of a storage area and writes the result (Arithmetic Operation Result) of the arithmetic operations 220 to a storage area of a target address (Page C) based on a corresponding request from one of the processors 10, 20, 30 containing an aggregation command and source and target information.

Referring to FIG. 12, the at least one aggregation engine 110 of the data mover 50 performs line-wise Boolean operations 230 with masked data (Masked Data Page A) of a first page from a source address of a storage area and with masked data (Masked Data Page B) from a source address of a second page of a storage area and writes the result (Boolean Operation Result) of the Boolean operations 230 to a storage area of a target address (Page C) based on a corresponding request from one of the processors 10, 20, 30 containing an aggregation command and source and target information.

FIG. 13 shows a method for data processing in a multiple processor system 1, in accordance with an embodiment of the present invention; FIG. 14 shows a data filtering operation, in accordance with an embodiment of the present invention; FIG. 15 shows a masked summing operation, in accordance with an embodiment of the present invention; and FIG. 16 shows a summing operation, in accordance with an embodiment of the present invention.

Referring to FIG. 13, in step S100 software of the page mover 50 receives request to access a storage Area. In step S110 data of the storage area is fetched line-wise from cache memories of the at least one shared cache level L2, L3, L4 and/or the main memory 60 to the at least one page mover 50 maintaining multiple processor cache memory access coherency. In step S120 aggregation and/or filtering of the fetched data is performed in the at least one page mover 50. In step S130 the processed data is moved from the page mover to the cache memories of the at least one shared cache level L2, L3, L4 and/or the main memory 60 and/or to the requesting processor 10, 20, 30 maintaining multiple processor cache memory access coherency.

Referring to FIG. 14, in step S200 software of the data processing system 1 receives request to filter a storage area. In step S210, the software of the data processing system 1 issues instruction to filter a page to one of the processors 10, 20, 30. In step S220 the corresponding processor 10, 20, 30 sends a filter command to the filter engine 110 of the corresponding page mover 50. In step S230, the filter engine 110 fetches a line from the corresponding storage area. In step S240 the filter engine 110 compares all elements of the line with an upper and lower bound. In step S250 the filter engine 110 puts the results in the bitmask buffer 112 and increments the count unit 118. In step S255, it is checked, if the bitmask buffer 112 is full. If the bitmask buffer 112 is full, the filter engine 110 ensures in step S260 that no other processor 10, 20, 30 works on the cache-line of the bitmask address. In step S270 the filter engine 110 stores the content of the bitmask buffer 112 at the bitmask address and increments the address. If the bitmask buffer 112 is not full, the process continuous directly with step S275. In step S275 it is checked, if the last line of the page is reached. If not, the process repeats steps S230 to S275. If yes, the software adds the current count value to a running sum in step S280. In step S285 it is checked, if the

last page in the storage area is reached. If not, the process repeats steps S210 to S285. If yes, the process ends.

Referring to FIG. 15, in step S300 software of the data processing system 1 receives request to conditionally sum up a storage area. In step S310, the software of the data processing system 1 issues instruction to conditionally sum up a page to one of the processors 10, 20, 30. In step S320 the corresponding processor 10, 20, 30 sends a conditionally sum up command to the aggregation engine 120 of the corresponding page mover 50. In step S330, the aggregation engine 120 fetches a mask from the corresponding storage area to the bitmask buffer 122. In step S340 the aggregation engine 120 fetches a line of the corresponding page. In step S350 the aggregation engine 120 sums up all qualifying elements of the line based on the mask in the bitmask buffer 122. In step S355, it is checked, if the mask in the bitmask buffer 122 is exhausted. If the mask in the bitmask buffer 122 is exhausted, the aggregation engine 120 repeats steps S330 to S355. If not, the process continues with step S360. In step S360 it is checked, if the last line of the page is reached. If not, the process repeats steps S340 to S360. If yes, the software adds the current page sum to a running sum in step S370. In step S375 it is checked, if the last page in the storage area is reached. If not, the process repeats steps S310 to S375. If yes, the software returns the sum to the requesting processor 10, 20, 30 in step S380 and the process ends.

Referring to FIG. 16, in step S400 software of the data processing system 1 receives request to sum up a storage area. In step S410, the software of the data processing system 1 issues instruction to sum up a page to one of the processors 10, 20, 30. In step S420 the corresponding processor 10, 20, 30 sends a sum up command to the aggregation engine 120 of the corresponding page mover 50. In step S430, the aggregation engine 120 fetches a line of the corresponding page. In step

S440 the aggregation engine 120 sums up all elements of the line. In step S450, it is checked, if the last line of the page is reached. If not, the process repeats steps S430 to S450. If yes, the software adds the current page sum to a running sum in step S460. In step S470 it is checked, if the last page in the storage area is reached. If not, the process repeats steps S410 to S470. If yes, the software returns the sum to the requesting processor 10, 20, 30 in step S480 and the process ends.

The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

The descriptions of the various embodiments of the present invention have been presented for purposes of illustration, but are not intended to be exhaustive or limited to the embodiments disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the

scope and spirit of the described embodiments. The terminology used herein was chosen to best explain the principles of the embodiments, the practical application or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments disclosed herein.

C L A I M S

What is claimed is:

1. A data processing system comprising multiple processors (10, 20, 30) with a hierarchical cache structure comprising multiple levels of cache (L1, L2, L3, L4) between said processors (10, 20, 30) and a main memory (60),
wherein at least cache memories of one cache level (L2, L3, L4) are shared between said processors (10, 20, 30);
wherein at least one page mover (50) is positioned closer to said main memory (60) and is connected to said cache memories of said at least one shared cache level (L2, L3, L4), said main memory (60) and to said multiple processors (10, 20, 30) to move data between said cache memories of said at least one shared cache level (L2, L3, L4), said main memory (60) and said processors (10, 20, 30);
wherein in response to a request from one of said processors (10, 20, 30) said at least one page mover (50) fetches data of a storage area line-wise from at least one of the following memories: Said cache memories of said at least one shared cache level (L2, L3, L4) and said main memory (60) maintaining multiple processor cache memory access coherency;
wherein said at least one page mover (50) comprises a data processing engine (100) which performs at least one of the following data processing operations: Aggregation and filtering of said fetched data; and
wherein said page mover (50) moves processed data to at least one of the following components: Cache memories of said at least one shared cache level (L2, L3, L4), said main memory (60) and said requesting processor (10, 20,

30) maintaining multiple processor cache memory access coherency.

2. The system according to claim 1, wherein said data processing engine (100) comprises at least one filter engine (110) filtering data of a storage area line-wise by comparing all elements of a fetched line from a source address of said at least one shared cache level (L2, L3, L4) and/or said main memory (60) with filter arguments to create a bitmask, and writing comparison results as bitmask data in a bitmask buffer of a target storage area located at a target address of said at least one shared cache level (L2, L3, L4) and/or said main memory (60) based on a corresponding request from one of said processors (10, 20, 30) containing a filter command with said filter arguments and source and target information.

3. The system according to claims 1 or 2, wherein said data processing engine (100) comprises at least one filter engine (110) moving data of a storage area (Page A) corresponding with bitmask data (M) of said bitmask buffer line-wise from a source address to a target address to create a bitmask data set based on a corresponding request from one of said processors (10, 20, 30) containing a move command and bitmask and source and target information.

4. The system according to claim 3, wherein said data processing engine (100) comprises at least one aggregation engine (120) performing arithmetic or Boolean operations with data of at least one storage area fetched from a corresponding source address having a corresponding bitmask data set and sending a data processing result to a storage area at a target address of said at least one shared cache level (L2, L3, L4)

and/or said main memory (60) or to a requesting processor (10, 20, 30) based on a corresponding request from one of said processors (10, 20, 30) containing a aggregation command and bitmask and source and target information.

5. The system according to claim 3, wherein at least one aggregation engine (120) performs arithmetic or Boolean operations with data of a first storage area fetched from corresponding source addresses having a corresponding first bitmask data set, and data of a second storage area fetched from corresponding source addresses having said corresponding bitmask set, and sends a data processing result to a storage area at target addresses of said at least one shared cache level (L2, L3, L4) and/or said main memory (60) or to a requesting processor (10, 20, 30) based on a corresponding request from one of said processors (10, 20, 30) containing an aggregation command and bitmask and source and target information.

6. A method for data processing in a multiple processor system with a hierarchical cache structure comprising multiple levels of cache (L1, L2, L3, L4) between said processors (10, 20, 30) and a main memory (60), wherein at least cache memories of one cache level (L2, L3, L4) are shared between said processors (10, 20, 30) and at least one page mover (50) is positioned closer to said main memory (60) and connected to said cache memories of said at least one shared cache level (L2, L3, L4), said main memory (60) and to said multiple processors (10, 20, 30) to move data between said cache memories of said at least one shared cache level (L2, L3, L4), said main memory (60) and said processors (10, 20, 30); wherein in response to a request from one of said processors (10, 20, 30) said method performs the steps of:

Fetching data of a storage area line-wise from at least one of the following memories: Said cache memories of said at least one shared cache level (L2, L3, L4) and said main memory (60) to said at least one page mover (50) maintaining multiple processor cache memory access coherency;

performing at least one of the following data processing operations in said at least one page mover (50): Aggregation and filtering of said fetched data; and

moving processed data from said page mover (50) to at least one of the following components: Cache memories of said at least one shared cache level (L2, L3, L4, said main memory (60) and said requesting processor (10, 20, 30) maintaining multiple processor cache memory access coherency.

7. The method according to claim 6, wherein data of a storage area is moved line-wise from a source address of said at least one shared cache level (L2, L3, L4) and/or said main memory (60) to a target address of said at least one shared cache level (L2, L3, L4) and/or said main memory (60) based on a corresponding request from one of said processors (10, 20, 30) containing a move command and source and target information.

8. The method according to claim 6 or 7, wherein based on a corresponding request from one of said processors (10, 20, 30) containing a filter command with said filter arguments and source and target information data of a storage area are filtered line-wise by comparing all elements of a fetched line from a source address of said at least one shared cache level (L2, L3, L4) and/or said main memory (60) with filter arguments, wherein comparison results are written in a bitmask buffer

located at a target address of said at least one shared cache level (L2, L3, L4) and/or said main memory (60).

9. The method according to one of the preceding claims 6 to 8, wherein based on a corresponding request from one of said processors (10, 20, 30) containing a aggregation command and source and target information arithmetic or Boolean operations are performed with data of at least one storage area fetched from a corresponding source address of said at least one shared cache level (L2, L3, L4) and/or said main memory (60), wherein a data processing result is send to a storage area at a target address of said at least one shared cache level (L2, L3, L4) and/or said main memory (60) or to a requesting processor (10, 20, 30).

10. The method according to claim 9, wherein at least one aggregation engine (120) performs arithmetic or Boolean operations with data of a first storage area and a masked or unmasked second storage area fetched from corresponding source addresses of said at least one shared cache level (L2, L3, L4) and/or said main memory (60).

11. A data processing program for execution in a data processing system comprising software code portions for performing a method for data processing in a multiple processor system according to one of the preceding claims 6 to 10 when said program is run on said data processing system.

12. A computer program product stored on a computer-usable medium, comprising computer-readable program means for causing a computer to perform a method for data processing in a multiple processor system according to

one of the preceding claims 6 to 10 when said program is run on said computer.

Intellectual Property Office

Application No: GB1321307.9 **Examiner:** Mr David Maskery

Claims searched: 1 - 12 **Date of search:** 20 May 2014

Patents Act 1977: Search Report under Section 17

Documents considered to be relevant:

Category	Relevant to claims	Identity of document and passage or figure of particular relevance
A,E	-	GB 2503437 A (IBM) See accelerator part 30 fig 3.
A	-	US 2009/0198955 A (ARIMILLI et AL) See AMM mover part 220 fig 2.

Categories:

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

Field of Search:

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC^{X} :

Worldwide search of patent documents classified in the following areas of the IPC

G06F

The following online and other databases have been used in the preparation of this search report

EPODOC, WPI, TXTE.

International Classification:

Subclass	Subgroup	Valid From
G06F	0012/08	01/01/2006