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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 March 2020 (26.03.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/059156 A1

(51) International Patent Classification:
*G06F 9/38* (2006.01) *G06F 17/16* (2006.01)

(21) International Application Number:
PCT/JP2018/035246

(22) International Filing Date:
18 September 2018 (18.09.2018)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **NEC CORPORATION** [JP/JP]; 7-1, Shiba 5-chome, Minato-ku, Tokyo 1088001 (JP).

(72) Inventor: **SUN, Heming**; c/o NEC Corporation, 7-1, Shiba 5-chome, Minato-ku, Tokyo 1088001 (JP).

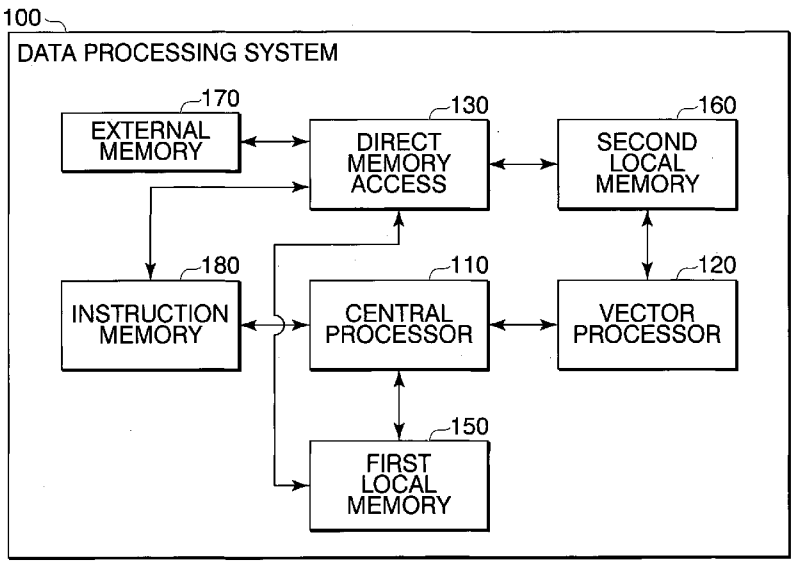
(74) Agent: **TANAI, Sumio** et al.; 1-9-2, Marunouchi, Chiyoda-ku, Tokyo 1006620 (JP).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ,

(54) Title: DATA PROCESSING SYSTEM, METHOD, AND PROGRAM

*FIG. 15*




(57) Abstract: The present disclosure provides a data processing system including a central processor; a vector processor electronically connected to the central processor and configured to perform operations based on instructions received from the central processor; an instruction memory unit electronically connected to the central processor and configured to store instructions; an external memory unit; a first local memory unit electronically connected to the central processor and configured to store one-dimensional systolic data; a second local memory unit electronically connected the vector processor and configured to store matrix data; and a direct memory access unit electronically connected to the first local memory unit, the second local memory unit, the instruction memory unit, and the external memory unit and configured to access data in the external memory unit, wherein data is transferred via the direct memory access unit at a timing based on a predetermined selection priority.

WO 2020/059156 A1

*[Continued on next page]*

TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Published:**

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

[DESCRIPTION]

[Title of the Invention]

DATA PROCESSING SYSTEM, METHOD, AND PROGRAM

[Technical Field]

[0001]

The present invention generally relates to data processing system, method, and program based a one-dimensional array architecture, and, in particular, to be used for continuous matrix processing including matrix transpose calculations.

[Background Art]

[0002]

Machine learning has become very popular in the recent years due to its high performance in many research fields. With more and more applications for machine learning being developed, the computation complexity has been greatly increasing. Therefore, efficient data processing is very important. In order to improve calculation efficiency, increased parallelism in vector processing is highly beneficial and therefore preferred.

[0003]

In vector processing, the main concept is to process data in a vector pattern. In order to process the matrix data, vector lanes are used in which each vector lane should include an arithmetic logic unit (ALU). In order to store the calculation data, each vector lane should include a local memory. In addition, in order to transfer the data between on-chip local memory and off-chip external memory, direct memory access (DMA) may be used. Finally, for the overall control logic, a central control unit is preferably used in the system. Based on the above concepts, some vector processor

designs can be found in Patent Literature 1, Non-Patent Literature 1, and Non-Patent Literature 2.

[0004]

In machine learning, one of the most common computations is matrix computation. General matrix multiply (GEMM) is used in both forward and backward propagation. When performing the GEMM function, all the source data has to be taken from the memory, and in order to fully utilize the fetched data from each element, two-dimensional processing elements are used. However, a disadvantage of the two-dimensional array is that if the actual matrix size in the calculation is much smaller than the size of the supported two-dimensional array, the computation resource in the unused processing elements are wasted. As an alternative, a one-dimensional array can also be used due to its higher flexibility.

[Citation List]

[Patent Literature]

[Patent Literature 1] United States Patent No. 005600843A

[Non-Patent Literature 1] "Application-Specific Soft-Core Vector Processor for Advanced Driver Assistance Systems" written by Stephan Nolting et al., published in Sept. 2017 at International Conference on Field Programmable Logic and Applications.

[Non-Patent Literature 2] "Fully Pipelined Soft Vector Processor as a CPU Accelerator" written by Yeyong Pang et al., published in Nov. 2017 at Chinese Journal of Electronics, vol. 26, no. 6, pp. 1198-1205.

[Disclosure of Invention]

[Technical Problem]

[Problem to be solved by the Invention]

[0005]

A first problem with conventional technology is that a matrix transpose is not generally supported in vector processors. In machine learning, a matrix transpose is used in several cases such as backward propagation for a fully-connected layer. In the referenced literatures, a way to efficiently transpose matrices inside the processor is not disclosed. Therefore, if a specific instruction for a matrix transposition is not supported by the vector processor, the transposition must be performed outside of the vector processor. In order to do that, the matrix to be transposed should first be transferred from the local memory inside the vector processor to an external memory outside the processor. After the transposed results are prepared in the external memory, the transposed matrix should be transferred from the external memory outside of vector processor to the vector processor, which wastes a lot of time on data transfer.

[0006]

A second problem is that there are many data transfers mainly composed of two types. One is a transfer between the local memories themselves, and the other is a transfer between local memory and external memory. Since the external memory is not always available for the current vector processor, the data transfer between the local memory and the external memory usually have bubble cycles. Moreover, when performing the data transfer between the local memory and the external memory, even though there are several bubble cycles, other transfer requests cannot be processed since the requests are submitted one by one in a serial manner which reduces the throughput performance of the whole computation system.

[0007]

One exemplary objective of the present invention is to provide a matrix transposition device that is capable of solving the first problem (identified above) in

which matrix transposition is not supported in the conventional vector processing engines.

[0008]

Another exemplary objective of the present invention is to provide a parallel system that is capable of solving the second problem (identified above) in which multiple requests for one local memory cannot be accepted at the same time.

[Means for Solving the Problem]

[0009]

A first aspect of the present disclosure provides a data processing system including a central processor; a vector processor electronically connected to the central processor and configured to perform operations based on instructions received from the central processor; an instruction memory unit electronically connected to the central processor and configured to store instructions; an external memory unit; a first local memory unit electronically connected to the central processor and configured to store one-dimensional systolic data; a second local memory unit electronically connected the vector processor and configured to store a matrix data; and a direct memory access unit electronically connected to the first local memory unit, the second local memory unit, the instruction memory unit, and the external memory unit and configured to access data in the external memory unit, wherein data is transferred via the direct memory access unit at a timing based on a predetermined selection priority.

[0010]

A second aspect of the present disclosure provides a method for a data processing system. The data processing system includes: a central processor; a vector processor electronically connected to the central processor; an instruction memory unit electronically connected to the central processor; an external memory unit; a first local

memory unit electronically connected to the central processor; a second local memory unit electronically connected the vector processor; a direct memory access unit electronically connected to the first local memory unit, the second local memory unit, the instruction memory unit, and the external memory unit. The method includes: performing, by the vector processor, operations based on instructions received from the central processor; storing, by the instruction memory unit, instructions; storing, by the first local memory unit, one-dimensional systolic data; storing, by the second local memory unit, matrix data; and accessing, by the direct memory access unit, data in the external memory unit. Data is transferred via the direct memory access unit at a timing based on a predetermined selection priority.

[0011]

A third aspect of the present disclosure provides a program for a data processing system. The data processing system includes: a central processor; a vector processor electronically connected to the central processor; an instruction memory unit electronically connected to the central processor; an external memory unit; a first local memory unit electronically connected to the central processor; a second local memory unit electronically connected the vector processor; a direct memory access unit electronically connected to the first local memory unit, the second local memory unit, the instruction memory unit, and the external memory unit. The program causes: the vector processor to perform operations based on instructions received from the central processor; the instruction memory unit to store instructions; the first local memory unit to store one-dimensional systolic data; the second local memory unit to store matrix data; and the direct memory access unit to access data in the external memory unit. Data is transferred via the direct memory access unit at a timing based on a predetermined selection priority.

[0012]

One effect of the present invention is that a matrix transpose may be performed inside the local memories. The reason for the effect is that a specific instruction for the matrix transpose can be made.

[0013]

When the vector processor receives this instruction, it will start to transfer the data from one local memory to the other. During the transfer, the mapped address for the source and destination memory is calculated thus each data item from the source matrix will be put in a transpose manner in the destination matrix.

[0014]

A second effect is that data between two local memories and external memory can be transferred within the same period.

[0015]

The reason for the effect is that the data transfer with the lower priority can be executed during the bubble cycles of the data transfer with the higher priority due to a priority request selection scheme being implemented.

[BRIEF DESCRIPTION OF THE DRAWINGS]

[FIG. 1]

A block diagram illustrating the structure of a first example embodiment of the present invention.

[FIG. 2]

A block diagram illustrating the structure of a local memory of the first example embodiment.

[FIG. 3]

A block diagram illustrating the structure of a local memory of the first example embodiment.

[FIG. 4]

A block diagram illustrating the structure of the processing element of the first example embodiment.

[FIG. 5]

A block diagram illustrating the structure of the matrix transfer of the first example embodiment.

[FIG. 6]

A block diagram illustrating the data mapping for a local memory of the first example embodiment.

[FIG. 7]

A block diagram illustrating the data mapping for a local memory of the first example embodiment.

[FIG. 8]

A block diagram illustrating the priority of different memory access requests for a local memory of the first example embodiment.

[FIG. 9]

A block diagram illustrating the priority of the different memory access requests for a local memory of the first example embodiment.

[FIG. 10]

A flow diagram illustrating the procedures of two continuous GEMM calculations without a matrix transposition by a conventional method.

[FIG. 11]

A flow diagram illustrating the procedures of two continuous GEMM calculations without a matrix transposition by the method of the first example embodiment.

[FIG. 12]

A flow diagram illustrating the procedures of two continuous GEMM calculations with a matrix transposition by a conventional method.

[FIG. 13]

A flow diagram illustrating the procedures of two continuous GEMM calculations with a matrix transposition by the method of the first example embodiment.

[FIG. 14]

A block diagram illustrating the mechanism of two requests working in the same period with the priority selection of the first example embodiment.

[FIG. 15]

A block diagram illustrating the structure of another example embodiment of the present invention.

[EXAMPLE EMBODIMENTS]

[Explanation of Structure]

[0016]

First, a first example embodiment of the invention is elaborated below referring to the accompanying drawings.

[0017]

Referring to FIG. 1, in the first example embodiment of the present invention, a data processing system 100 contains central processor (CP) 110, vector processing (VP) engine 120, DMA 130, matrix transfer device 140, a first local memory 150 for data

storage, a second local memory 160 for data storage, an instruction memory 180 for instruction storage, and an external memory 170.

[0018]

The CP 110 may be a MIPS processor, or a similar architecture processor, which supports basic instructions such as arithmetic computation and store/load to/from external memory 170 with a general purpose register inside the central processor 110. The central processor 110 controls the first and second local memories 150, 160 to fetch the calculation data from external memory 170, and also prepare all of the instructions stored in the instruction memory 180. The central processor 110 then sends the vector instruction and the matrix data to the vector processor 120. The vector processor 120 receives the instruction and starts to do the computation. The calculation of the vector processor 120 is based on a one-dimensional systolic pattern. One input is fetched from the first local memory 150, the other input is fetched from the second local memory 160. After calculation, the results are stored into the second local memory 160. There is a path between the first local memory 150 and the second local memory 160, and data can be transferred between the first local memory 150 and the second local memory 160 by way of a matrix transfer device 140. The transfer between first and second local memories 150, 160 can be a normal transfer or a transposed transfer. If the calculation finishes, the result in the second local memory 160 may be transferred to the external memory 170 through the direct memory access 130.

[0019]

The detail architecture inside the vector processor 120 is shown in FIG.4. Each of a plurality of processors 121 is connected to each other. There is a connection between the central processor 110 and the first processing element 121, which is used to transfer the data value and the instruction information. Between neighboring processing

elements 121, there is a connection channel which is used to broadcast the information to each processing element 121. Inside each processing element 121, in order to calculate the multiplication and addition efficiently, a digital signal processor (DSP) can be used to achieve lower power and higher frequency. In addition, there is a dedicated register 125 in each processing element 121 in order to store intermediate results. Each processing element 121 has one dedicated register 125.

[0020]

The details of the architecture of the local memory 150 are shown in FIG. 2. There are 16 two-port RAM banks 151, and one first data selection unit 153. The two-port RAM 151 is used to store data. The first data selection unit 153 selects the data transfer with the external memory 170 or the second local memory 160. There is also an output as the systolic data input for the vector processor 120. It should be noted that the number of two-port RAM banks for the first local memory 150 in this exemplary embodiment is 16, but could also be 8, 32, etc.

[0021]

The details of the architecture of the second local memory 160 are shown in FIG. 3. There are two-port RAM banks 161, a second data selection unit 162, and a data-ring 163. The number of RAM banks 161 is equal to the number of processing elements 121, one processing element 121 is corresponding to one RAM bank 161.

[0022]

The details of the architecture of the matrix transfer device 140 are shown in FIG. 5. Inside the matrix transfer device 140, there is an address generator 141 and bank number generator 142. Because the organization of the first local memory 150 and the second local memory 160 are different, no matter what kind of data is transferred from either local memory to the other, the bank and the address in each bank in the

destination bank would be different from the bank and the address in the source bank. Therefore, the address generator 141 and the bank number generator 142 are used.

[Description of Operation]

[0023]

Next, referring to flowcharts in FIGS. 10 to 13, the general operation of the present example embodiment is described in detail.

[0024]

First, the direct memory access unit (DMA) transfers the instruction from the external memory 170 to the instruction memory 180. For each application, the application could be compiled to assembly code. Therefore, the assembly codes are made for each application and stored in the instruction memory 180.

[0025]

After that, the central processor 110 fetches the instructions one by one from the instruction memory 180. The initial data is stored in the first local memory 150 and the second local memory 160 before starting calculation. When performing the general matrix multiply (GEMM) function, the left matrix is stored in the first local memory 150, and the right matrix is stored in the second local memory 160. For each matrix M*K, the data mapping in the first local memory 150 is explained with reference to FIG. 6. When there are 16 banks of RAM used for the first local memory 150, D1[0,0] (D1[m,k] represents the m-th row and k-th column element in the matrix) is stored in the address0 of bank0 of the first local memory 150 in order to store the matrix. The data D1[0,1], for example, is stored in the address0 of bank1 of the first local memory 150, and the data D1[0,15] is stored in the address0 of bank15 of the first local memory 150, for example. Starting from D1[0,16], the data is stored in the address1 of bank0 of the first local memory 150.

[0026]

Therefore, the address generator 141 and the corresponding bank are calculated by the following equations.

ADDR=(m*K+k)/16

BANK=(m*K+k)%16

[0027]

About the data mapping for the second local memory 160, since each processing element 121 is corresponding to one RAM bank 161, the number of RAM bank 161 is equal to the number of processing elements 121. In an example where there are 256 processing elements 121, a data mapping method is given with reference to FIG. 7. D2[0,0] is stored in the address 0 of the bank0 of the second local memory 160, D2[0,1] is stored in the address 0 of the bank1 of the second local memory 160, and so on. For D2[1,0], the data is stored in the address 1 of the bank0 of the second local memory 160. For D2[1,1], the data is stored in the address 1 of bank1 of the second local memory 160. For each matrix K*N, if N is smaller than the number of PEs 121, the unused column will be filled with zeroes. If N is larger than the number of PEs 121, the matrix will be cut by column according to the number of PEs 121, and then mapped to the second local memory 160. Therefore, for the element D2[k,n] (k represents the row, while n represents the column), the address generator and the corresponding bank is obtained by the following equations.

ADDR=k

BANK=n

[0028]

In order to store the matrix in the second local memory 160, a data-ring 163 is used to pass the data through all of the two-port RAM banks, and write/read the data

to/from the corresponding two-port RAM banks. For example, when the cache line is 64 bytes and single precision floating point which is 32 bit is used, there are 16 data items in one cache line. Each data item should be stored in the corresponding two-port RAM bank 161.

[0029]

After storing the left and right matrices in the first local memory 150 and the second local memory 160, the computation is executed in a one-dimensional systolic manner. For the GEMM calculation, given that the left matrix is D1(M,K), and the right matrix is D2(K,N), the multiplication result is RES(M,N). For each element of RES[m,n], the following calculation is conducted.

[Math. 1]

$$\mathrm{RES[m,n]} = \sum_{k=0}^{K-1} D1[m,k] * D2[k,n]$$

D1[0,0] is read from the local memory 150 and transferred to processing elements 121. D2[0,0] is read from the second local memory 160 and will be the other input operand for the processing element 121. After the calculation, the multiplication of D1[0,0] and D2[0,0] will be stored in a dedicated register. D1[0,0] will be transferred to the second processing element 121 in a systolic manner. D2[0,1] is read from the second local memory 160. D1[0,0] and D2[0,1] are two operands for the multiplication. The result of D1[0,0]*D2[0,1] is stored in the dedicated register 125. For the latter processing elements 121, D1[0,0] is always transferred in a systolic way. Finally, D1[0,0] will transfer through all the processing elements 121. In fact, the dataflow for each processing element 121 is completely the same. Therefore, the data flow of one processing element 121 will simply be explained hereinafter.

[0030]

After D1[0,0] is transferred through all the PEs 121, the next data item will be read from the first local memory 150. The left matrix is read in row-major order. Therefore, D1[0,1], for example, is read from the first local memory 150 and sent to the first processing element 121, D2[1,0] is read from the second local memory 160 in the first processing 121. Thereafter, D1[0,1]*D2[1,0] is calculated. The previous result of D1[0,0]*D2[0,0] is read from the dedicated register 125 and added to the result of D1[0,1]*D2[1,0]. The sum is stored in the dedicated register 125 again. By doing so iteratively, the first element RES[0,0] in the first processing element 121 can be obtained. Similarly, all of the elements in the first processing element 121 can be calculated. After calculating the elements, the results are sent to the two-port RAM bank 161 in the second local memory 160.

[0031]

Hereinafter, the data transfer between the first local memory 150 and the second local memory 160 will be explained. There are four cases. The first case is a normal transfer from the first local memory 150 to the second local memory 160. Supposing that the matrix size stored in the second local memory 160 is KxN, N processing elements 121 are used and the depth of each RAM bank is K. As described above, if N is smaller than the number of processing elements 121 (e.g. 256), the unused columns are filled with zeroes. Therefore, D2[0,0] in the second local memory 160 is mapped to the first element of the first RAM bank in the local memory 150. D2[0,1] in the second local memory 160 is mapped to the first element of the second RAM in the first local memory 150. However, for D2[0,n] in the second local memory 160, if n is larger than the number of banks supported by the first local memory 150, the address and bank in the first local memory 150 and the second local memory 160 are different. For the address and bank mapping of the other elements, the calculation method is shown in the

following equation where DST_ADDR and DST_BANK are the destination address and bank respectively. In this case, the destination address and the bank are in the first local memory 150. SRC_ADDR and SRC_BANK represent the source address and the source bank, in this case, the source address and the source bank are in the second local memory 160. SRC_NUM_BANK is the number of two-port RAM banks 161 in the second local memory 160, and DST_NUM_BANK is the number of two-port RAM banks 151 supported in the first local memory 150.

DST_ADDR=(SRC_ADDR*SRC_NUM_BANK+SRC_BANK)/DST_NUM_BANK

DST_BANK=(SRC_ADDR*SRC_NUM_BANK+SRC_BANK)%DST_NUM_BANK

[0032]

In the real situation, in each clock cycle, 512-bit data are transferred supposed that the cache line is 512-bit. For the 32-bit single precision float case, one data item is 32-bits, so 16 data items are transferred in one clock cycle. Supposing there are 256 processing elements 121, for the data transfer between the first local memory 150 and the second local memory 160, in the first clock cycle, D2[0,0], D2[0,1]...D2[0,15] are taken out from the RAM 161 of each processing element 121 and transferred to the first local memory 150 through data-ring. Since D2[0,0], D2[0,1] ... D2[0,15] are stored in the different banks of the first local memory 150, so these data items can be written to the first local memory 150 in the same clock cycle. In the second clock cycle, D2[0,16], D2[0,17], D2[0,18]...D2[0,31] are taken out from the RAM bank 161 of the corresponding processing element 121 and transferred to the first local memory 150 through data-ring 163. Since D2[0,16], D2[0,17]...D2[0,31] are stored in different banks of the local memory 150, so these data items can be written to the first local memory 150 in the same clock cycle. Similarly, in each clock cycle, 16 data items are

read from the RAM bank 161 of the corresponding processing element 121, and then transferred through the data-ring 163, and finally stored in the first local memory 150.

[0033]

The second case is a normal transfer from the first local memory 150 to the second local memory 160. Similarly, the address and the bank generator is the same as the above, the only difference is that source becomes the first local memory 150, while the destination becomes the second local memory 160.

[0034]

The third case is that the transposed matrix is transferred from the first local memory 150 to the second local memory 160. Supposing that the matrix [N,M] is stored in the first local memory 150, this matrix is transposed to the size of [M,N] and stored in the second local memory 160. For example, D1[0,0] in the first local memory 150 is mapped to address 0 in bank 0 in the second local memory 160. D1[0,1] in the second local memory 160 is mapped to the address 1 in bank 0 in the second local memory 160. For the address and bank mapping of the other elements, the calculation method is shown in the following equations.

DST_ADDR=(SRC_ADDR*SRC_NUM_BANK+SRC_BANK)%DST_NUM_BANK

DST_BANK=(SRC_ADDR*SRC_NUM_BANK+SRC_BANK)/DST_NUM_BANK

[0035]

The transfer is still based on the 16x16 block. However, this 16x16 block is in the second local memory 160. In order to generate the 16x16 block for the side of the second local memory 160 in each 16 clock cycles, therefore, we have to find the corresponding elements of the D2[0,0], D2[1,1], D2[2,2]...D2[15,15] in the first local memory 150. The corresponding address for D2[0,0] is address 0 in bank 0, the corresponding address for D2[1,1] is address (N+1)/16 in bank 1, and the corresponding

address for D2[2,2] is address (Nx2+2)/16 in bank 2. Similarly, the corresponding position for D2[15,15] is address (Nx15+15)/16 in bank 15. Therefore, 16 data items can be fetched in one clock cycle. In the second clock cycle, the corresponding elements of the D2[1,0], D2[2,1], D2[3,2]...D2[0,15] are found in the first local memory 150. The corresponding address for D2[1,0] is address N/16 in bank0, the corresponding address for D2[2,1] is address (Nx2+1)/16 in bank1. Similarly, the corresponding position for D2[0,15] is address 0 in bank15. By these mapping methods, the address of the first local memory 150 for 16x16 blocks of the local memory 160 can be known. In each clock cycle, the 16 elements fetched from the first local memory 150 are transposed and stored in the second local memory 160. For example, in the first cycle, D2[0,0], D2[1,1], D2[2,2]...D2[15,15] are the diagonal, so the transposed results are the same. For the second local memory 160, the address of the bank0 is 0, the address of the bank1 is 1 and so on. In the second clock cycle, D2[1,0], D2[2,1], D2[3,2]...D2[0,15] are fetched from the first local memory 150. For D2[1,0], the transposed result should be stored in the address 0 of bank1, D2[2,1] should be stored in the address 1 of bank2, D2[3,2] should be stored in the address 2 of bank3. Similarly, all the data can be stored in the 16 banks of the second local memory 160 in one clock cycle.

[0036]

The address generator for the first local memory 150 can be represented as the following equation where CNT is the count of each 16 clock cycles for a 16x16 block, SRC_BANK means which bank in the first local memory 150.

SRC_ADDR_IN_B16=(N*(SRC_BANK+CNT)+SRC_BANK)/16 if SRC_BANK +CNT <=15

SRC_ADDR_IN_B16=(N*(SRC_BANK+CNT-16)+SRC_BANK)/16 if SRC_BANK +CNT >15

[0037]

The basic address for the 16x16 block is composed of two parts, one part is corresponding to the vertical movements of the 16x16 block, marked as delta_v, and the other part is corresponding to the horizontal movements of the 16x16 block, marked as delta_h. B16_vertical_num represents the vertical address of the 16x16 block in the second local memory 160. Here, a vertical scan of the 16x16 block in the second local memory 160 is used.

B16_vertical_num=(B16_vertical_num<M/16)?B16_vertical_num+1:0

Delta_v=(B16_vertical_num<M/16)?(Delta_v+N):0

Delta_h=(B16_vertical_num==0)?(Delta_h+1):Delta_h

SRC_B16_ADDR=Delta_v+Delta_h

[0038]

Therefore, the final address for 16 banks of the first local memory 150 can be obtained by the following equation.

SRC_ADDR=SRC_ADDR_IN_B16+SRC_B16_ADDR

[0039]

The address generator for the first local memory 150 has been given in the above. Next, an explanation of the address and bank generator for the second local memory 160 is given. Since the scan of the source matrix is the vertical scan in the second local memory 160, the scan of the transposed matrix is a horizontal scan in the second local memory 160. Therefore, the basic address and basic bank can be calculated by the following.

DST_ADDR_IN_B16 = (DST_BANK%16-CNT) if DST_BANK%16-CNT >= 0

DST_ADDR_IN_B16 = (DST_BANK%16-CNT+16) if DST_BANK%16-CNT < 0

DST_B16_ADDR=( DST_B16_BANK <N/16)? DST_B16_ADDR:(DST_B16_ADDR +16)

DST_ADDR=DST_ADDR_IN_B16+DST_B16_ADDR

[0040]

The fourth case is that the transposed matrix will be transferred from the second local memory 160 to the first local memory 150. Supposing that the matrix [M,N] is stored in the second local memory 160, this matrix needs to be transposed to [N,M] and stored in the first local memory 150. For example, D2[0,0] in the second local memory 160 is still mapped to address 0 in bank 0 in the first local memory 150. D2[0,1] in the second local memory 160 is mapped to the address 1 in bank 0 in the first local memory 150. For the address and bank mapping of the other elements, the calculation method is shown in the following equation.

DST_ADDR=(SRC_BANK*M+SRC_ADDR)/DST_NUM_BANK

DST_BANK=(SRC_BANK*M+SRC_ADDR)%DST_NUM_BANK

[0041]

If the data of D2[0,0], D2[0,1]...D2[0,15] is taken from the RAM 161 of each processing element 121, the 16 data items are stored in the same RAM bank of the first local memory 150. Therefore, we cannot store these 16 data elements into the first local memory 150 in one clock cycle. In order to avoid this problem, a cyclic data mapping method is used. The transfer is based on a 16x16 block, which means that after the data of one 16x16 block is accessed; the 16x16 block is shifted to the next 16x16 block. The top left 16x16 block is transferred first which takes 16 clock cycles. In the first clock cycle, D2[0,0], D2[1,1], D2[2,2]...D2[15,15] are read from the RAM 161 of the second local memory 160. These elements are stored in different banks of the first local

memory 150. D2[0,0] is stored in the first RAM of the first local memory 150, D2[1,1] is stored in the second RAM of the local memory, and so on. Therefore, we can store these 16 data items in the first local memory 150 in one clock cycle. In the second clock cycle, D2[0,1], D2[1,2], D2[2,3]...D2[14,15] and D2[15,0] are read from the RAM bank 161 of the second local memory 160. These elements are also stored in different banks of the first local memory 150. Therefore, we can store these 16 data in the first local memory 150 in one clock cycle. The address generator is the same as that of the third case, while the only difference is that the third case is to transfer from the first local memory 150 to the second local memory 160, while this fourth case is to transfer from the second local memory 160 to the first local memory 150. Therefore, the address of the source and destination is swapped.

[0042]

Regarding the second local memory 160, transfers between external memory 170 and the second local memory 160 and transfers between the first local memory 150 and the second local memory 160 are possible. If more than one transfer request is received at the same clock cycle, priority is determined by the data selection unit 162. An explanation of this operation is described below with reference to the flowcharts in FIG. 8. The first priority is data transfer from the external memory 170 to the second local memory 160. This is because the data has to been taken from the external memory 170 if the data is valid. The second priority is transfer from the second local memory 160 to the external memory 170. The third priority is transfer from the first local memory 150 to the second local memory 160, and the final priority is transfer from the second local memory 160 to the first local memory 150.

[0043]

Regarding the first local memory 150, a first data selection unit 153 is used. If more than one transfer request is received at the same clock cycle, a selection is performed based on priority, as shown in FIG. 9. The first priority is data transfer from the external memory 170 to the first local memory 150. The second priority is transfer from the first local memory 150 to the external memory 170. The third priority is transfer from the second local memory 160 to the first local memory 150, and the final priority is transfer from the first local memory 150 to the second local memory 160.

[0044]

For the continuous GEMMs (e.g. the first GEMM is A*B=C, while the second GEMM is C*D=E). Hereinafter, an explanation is given with reference to FIG. 10. The first GEMM is A*B=C, matrix A is stored in the first local memory 150, and matrix B is stored in the second local memory 160. After storing the matrix A in the first local memory 150 and the matrix B in the local memory 160, the computation is executed and the results are stored in the second local memory 160. If the second GEMM is C*D=E, matrix C should be stored in the first local memory 150 as the systolic one-dimensional input. In order to store the matrix C in the first local memory 150, the results have to be sent to the external memory 170 first, and then transferred from the external memory to the first local memory 150. After that, matrix D is stored in the second local memory 160 and then the computation is started. Finally, the calculation results are obtained and transferred to the external memory 170.

[0045]

However, in our methods, we can directly transfer the matrix C from the second local memory 160 to the first local memory 150. Therefore, we can save the transfer of matrix C to the external memory 170. The processing procedures are shown in FIG. 11.

Calculating the first GEMM can be parallelized when storing the matrix D of the second GEMM.

[0046]

Similarly, if the second GEMM is C.T*D=E where C.T is the transposed matrix of C, in the conventional method, the procedures are shown in FIG. 12. For calculating the first GEMM, the procedures are same those shown in FIG. 10. After storing C in the external memory, the central processor 110 will transpose the matrix C in the external memory 170. After that, the transposed matrix of C is transferred to the first local memory 150, matrix D is transferred to the second local memory 160 and the second GEMM can begin operation.

[0047]

However, in methods of the present disclosure, not only can direct transfer the matrix C from the second local memory 160 to the first local memory 150 be performed, but the transpose can be finished during the transfer. Therefore, the transfer of matrix C to the external memory 170 can be performed in fewer operations. In addition, transfer time to the central processor can be reduced. The processing procedures are shown in FIG. 13.

[0048]

In the above example, there are no simultaneous requests for the same local memory (i.e., the first local memory 150 or the second local memory160). However, in some implementations, these processes may occur. For example, after calculating the first GEMM A*B=C, the results of matrix C are should be transferred to the external memory 170. Meanwhile, the results should also be transferred to the first local memory 150 for the next GEMM C*D=E. In this case, transfer between the second local memory 160 and the external memory 170 has higher priority. As shown in FIG.

14, since the data transfer from/to the external memory 170 has some bubble cycles, these bubble cycles can be utilized to perform the transfer between the first local memory 150 and the second local memory 160.

[Description of Effect]

[0049]

Next, the effect of the present example embodiment is described.

[0050]

As the present example embodiment is configured in such a manner that the mapping address is given in a cyclic manner for two local memories, it is possible to transfer the data in a transposed manner through a data-ring 163 inside the accelerator, thus there is no need to perform the transpose on the host side.

[0051]

In addition, the example embodiment is configured in such a manner that the data transfer and the calculation use two different channels, which enable a continuous matrix computation and can be executed in a parallel way. Further, the data transfer of the previous matrix computation can be performed simultaneously with the computation of the next matrix computation.

[0052]

Moreover, more than one write/read requests can be executed in the same period for each local memory, which can utilize the bubble cycles of the communication between the local memory and the external memory.

[0053]

Referring to FIG. 15, in another example embodiment of the present invention, a data processing system 100 contains central processor (CP) 110, vector processing (VP)

engine 120, DMA 130, a first local memory 150 for data storage, a second local memory 160 for data storage, an instruction memory 180 for instruction storage, and an external memory 170.

[0054]

The above-mentioned program may be for partially carrying out the above-mentioned functions. The above-mentioned program may be a so-called difference file (difference program) that is combined with a program that is already recorded in the computer system in order to carry out the above-mentioned functions.

[0055]

All or some of the functions of the above-mentioned data processing system may be carried out by utilizing hardware such as an ASIC (Application Specific Integrated Circuit), a PLD (Programmable Logic Device), an FPGA (Field-Programmable Gate Array) or the like.

[0056]

In addition thereto, the features in the above-mentioned exemplary embodiments may be appropriately replaced with well-known features, within a range not departing from the scope of the present invention. Additionally, the technical scope of the invention is not limited to the above-mentioned exemplary embodiments, and various modifications may be made within a range not departing from the scope of the present invention.

[Industrial Applicability]

[0057]

The present invention is applicable to a data processing apparatus that involves a large amount of vector or matrix computation, such as image or video processing platforms, and deep learning platforms.

[Reference Signs List]

[0058]

100 Data Processing System

110 Central Processor (CP)

120 Vector Processor (VP)

121 Processing Element (PE)

125 Dedicated Register

130 Direct Memory Access (DMA)

140 Matrix Transfer Device

141 Address Generator

142 Bank Number Generator

150 First Local Memory

153 First Data Selection Unit

160 Second Local Memory

161 RAM Bank

162 Second Data Selection Unit

163 Data-Ring

170 External Memory

180 Instruction Memory

[CLAIMS]

[Claim 1]

1. A data processing system comprising:

a central processor;

a vector processor electronically connected to the central processor and configured to perform operations based on instructions received from the central processor;

an instruction memory unit electronically connected to the central processor and configured to store instructions;

an external memory unit;

a first local memory unit electronically connected to the central processor and configured to store one-dimensional systolic data;

a second local memory unit electronically connected the vector processor and configured to store matrix data; and

a direct memory access unit electronically connected to the first local memory unit, the second local memory unit, the instruction memory unit, and the external memory unit and configured to access data in the external memory unit, wherein

data is transferred via the direct memory access unit at a timing based on a predetermined selection priority.

[Claim 2]

2. The data processing system of claim 1, further comprising

a matrix transfer device electronically connected between the first and second local memory units and configured to transfer data between the first and second local memory units.

[Claim 3]

3. The data processing system of claim 2, wherein

the matrix transfer unit is able to perform matrix transposition on data when transferring the data between the first and second local memory units.

[Claim 4]

4. The data processing system of claim 3, wherein

the matrix transfer unit includes an address generator configured to generate memory addresses for a source memory and a destination memory, the source memory being one of the first local memory unit and the second local memory unit, and the destination memory being the other of the first local memory unit and the second local memory unit.

[Claim 5]

5. The data processing system of claims 1 to 4, wherein

the second local memory unit has a data ring configured to ring-broadcast data, the data ring using, when multiple memory requests are received, the predetermined selection priority to transfer data to the first local memory unit and the external memory unit via the direct memory access unit.

[Claim 6]

6. The data processing system of claims 1 to 5, wherein

the first and second local memory units each include a plurality of two-port RAM banks on which data is stored, the number of two-port RAM banks of the first local

memory unit being equal to the number of two-port RAM banks of the second local memory unit.

[Claim 7]

7. A method for a data processing system, the data processing system comprising: a central processor; a vector processor electronically connected to the central processor; an instruction memory unit electronically connected to the central processor; an external memory unit; a first local memory unit electronically connected to the central processor; a second local memory unit electronically connected the vector processor; a direct memory access unit electronically connected to the first local memory unit, the second local memory unit, the instruction memory unit, and the external memory unit, the method comprising:

performing, by the vector processor, operations based on instructions received from the central processor;

storing, by the instruction memory unit, instructions;

storing, by the first local memory unit, one-dimensional systolic data;

storing, by the second local memory unit, matrix data; and

accessing, by the direct memory access unit, data in the external memory unit, wherein

data is transferred via the direct memory access unit at a timing based on a predetermined selection priority.

[Claim 8]

8. A program for a data processing system, the data processing system comprising: a central processor; a vector processor electronically connected to the central

processor; an instruction memory unit electronically connected to the central processor; an external memory unit; a first local memory unit electronically connected to the central processor; a second local memory unit electronically connected the vector processor; a direct memory access unit electronically connected to the first local memory unit, the second local memory unit, the instruction memory unit, and the external memory unit, the program causing:

the vector processor to perform operations based on instructions received from the central processor;

the instruction memory unit to store instructions;

the first local memory unit to store one-dimensional systolic data;

the second local memory unit to store matrix data; and

the direct memory access unit to access data in the external memory unit, wherein

data is transferred via the direct memory access unit at a timing based on a predetermined selection priority.

*FIG. 1*

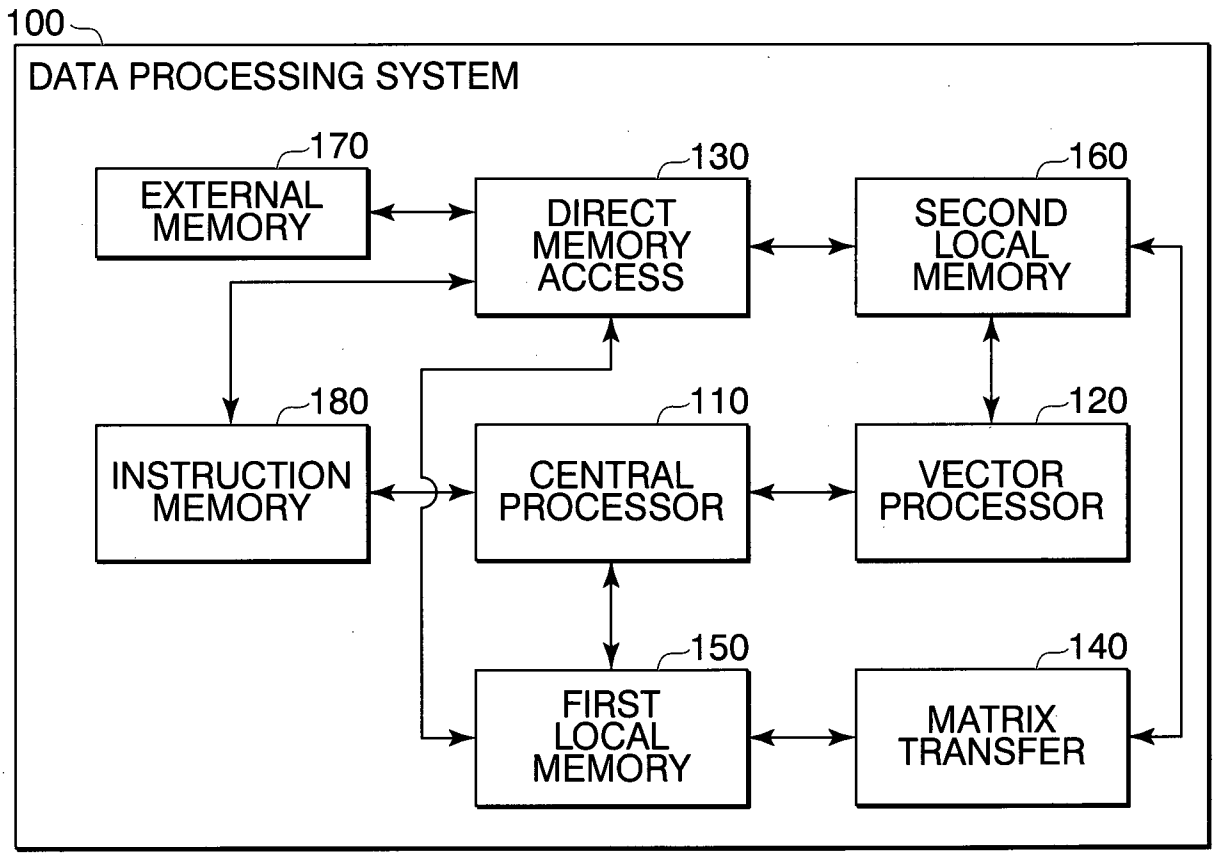

100
DATA PROCESSING SYSTEM
170
EXTERNAL MEMORY
130
DIRECT MEMORY ACCESS
160
SECOND LOCAL MEMORY
180
INSTRUCTION MEMORY
110
CENTRAL PROCESSOR
120
VECTOR PROCESSOR
150
FIRST LOCAL MEMORY
140
MATRIX TRANSFER

FIG. 2

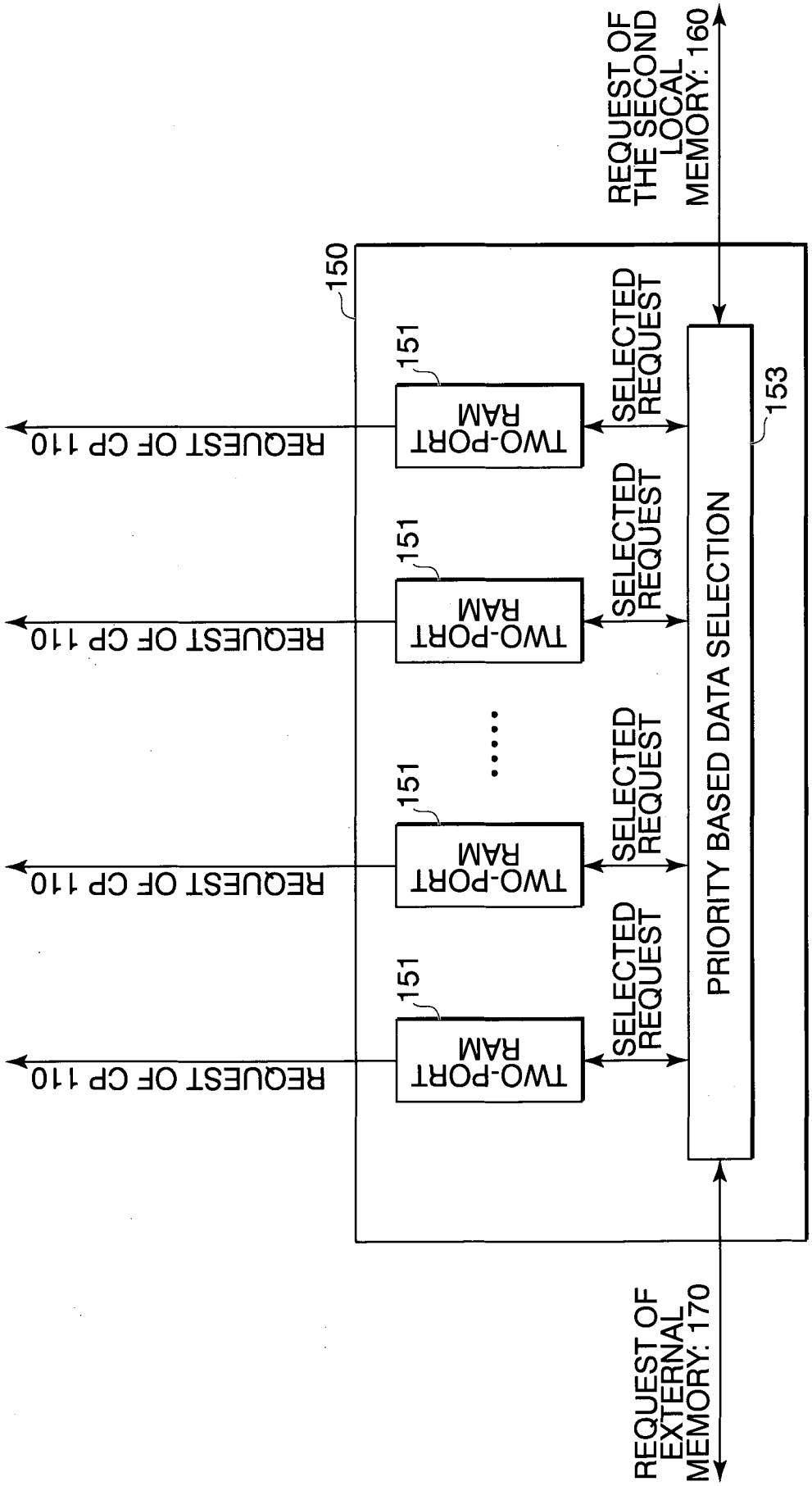

REQUEST OF THE SECOND LOCAL MEMORY: 160
150
151
TWO-PORT RAM
SELECTED REQUEST
153
PRIORITY BASED DATA SELECTION
REQUEST OF CP 110
REQUEST OF EXTERNAL MEMORY: 170

FIG. 3

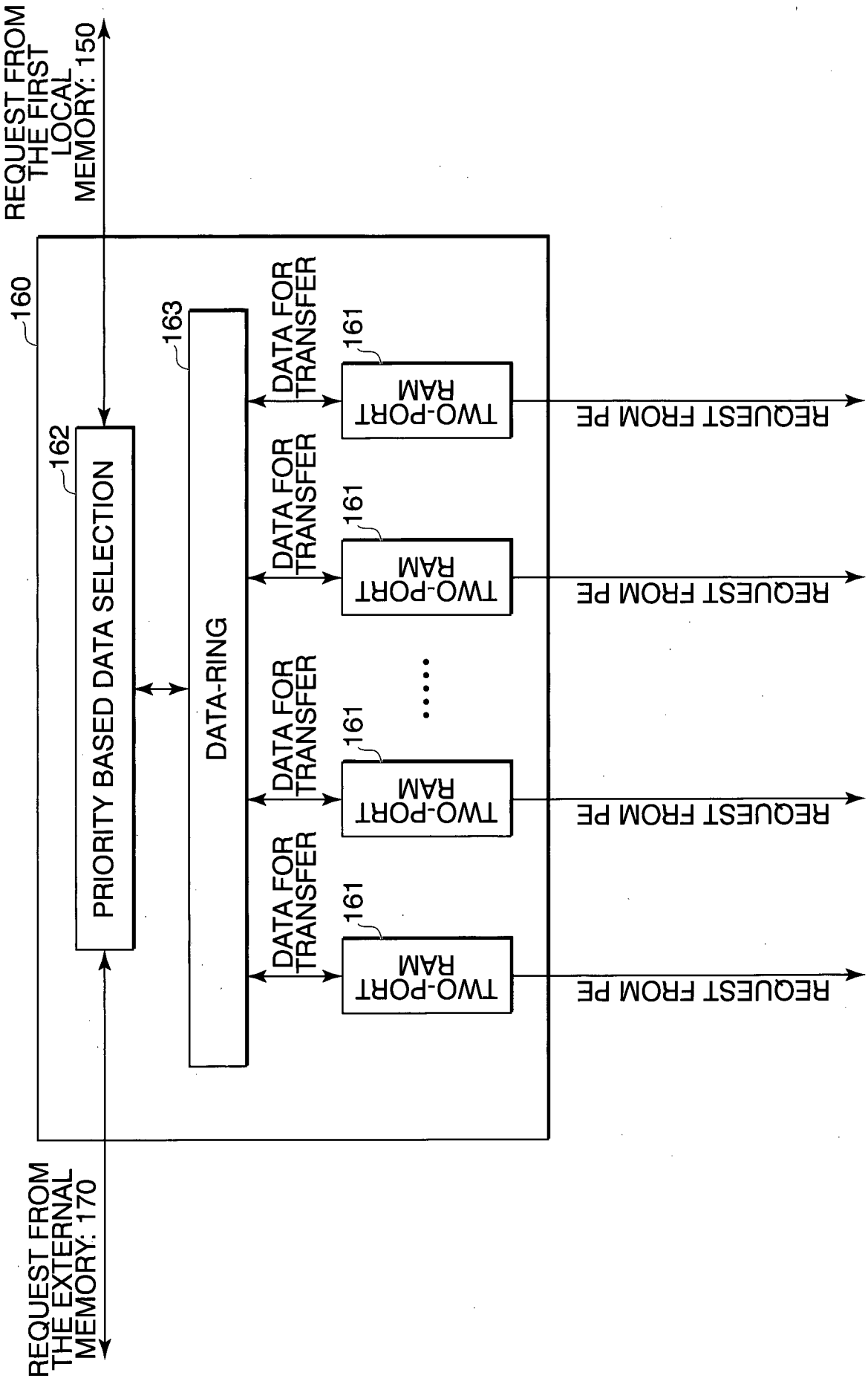

REQUEST FROM THE FIRST LOCAL MEMORY: 150
160
162
PRIORITY BASED DATA SELECTION
163
DATA-RING
DATA FOR TRANSFER
161
TWO-PORT RAM
REQUEST FROM PE
REQUEST FROM THE EXTERNAL MEMORY: 170

FIG. 4

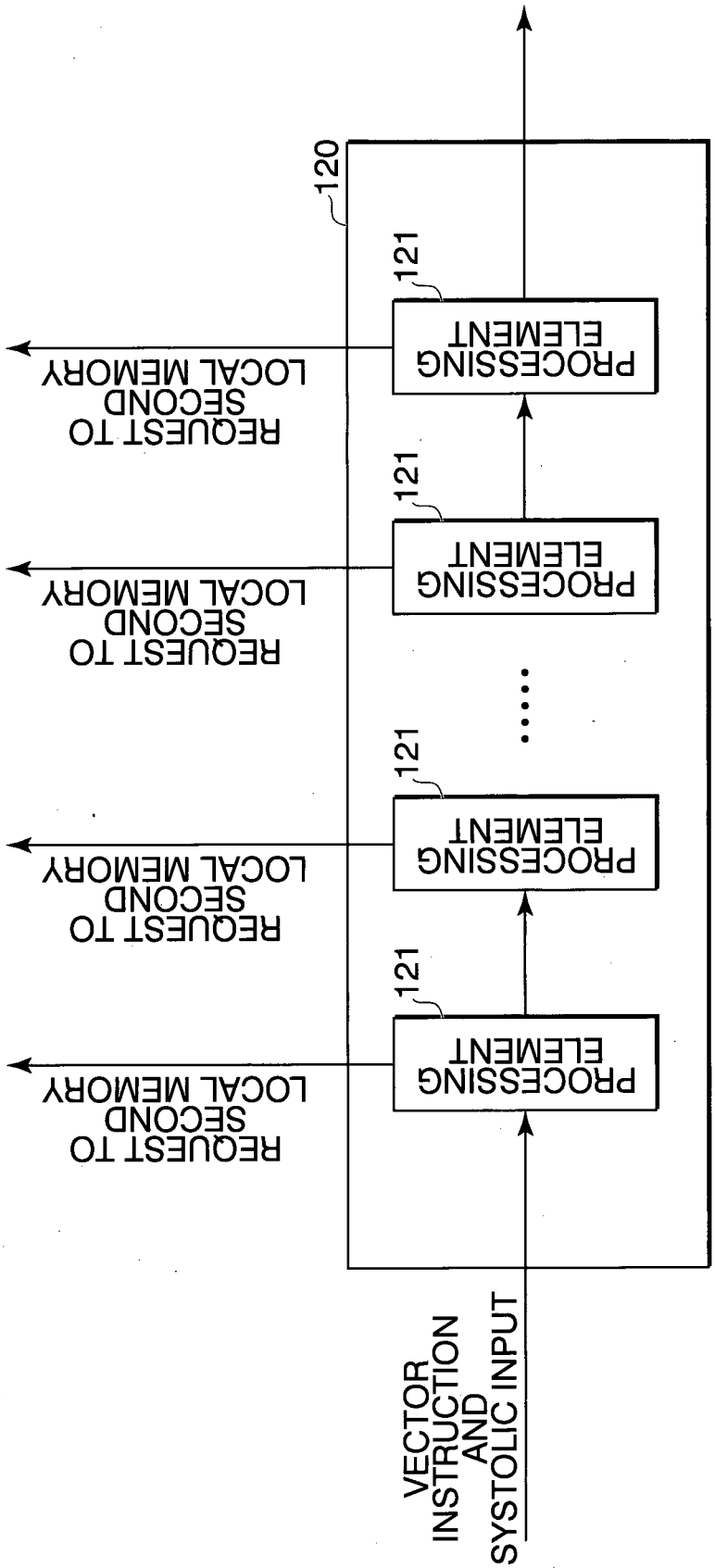

120
121
PROCESSING ELEMENT
121
PROCESSING ELEMENT
121
PROCESSING ELEMENT
121
PROCESSING ELEMENT
REQUEST TO SECOND LOCAL MEMORY
REQUEST TO SECOND LOCAL MEMORY
REQUEST TO SECOND LOCAL MEMORY
REQUEST TO SECOND LOCAL MEMORY
VECTOR INSTRUCTION AND SYSTOLIC INPUT

*FIG. 5*

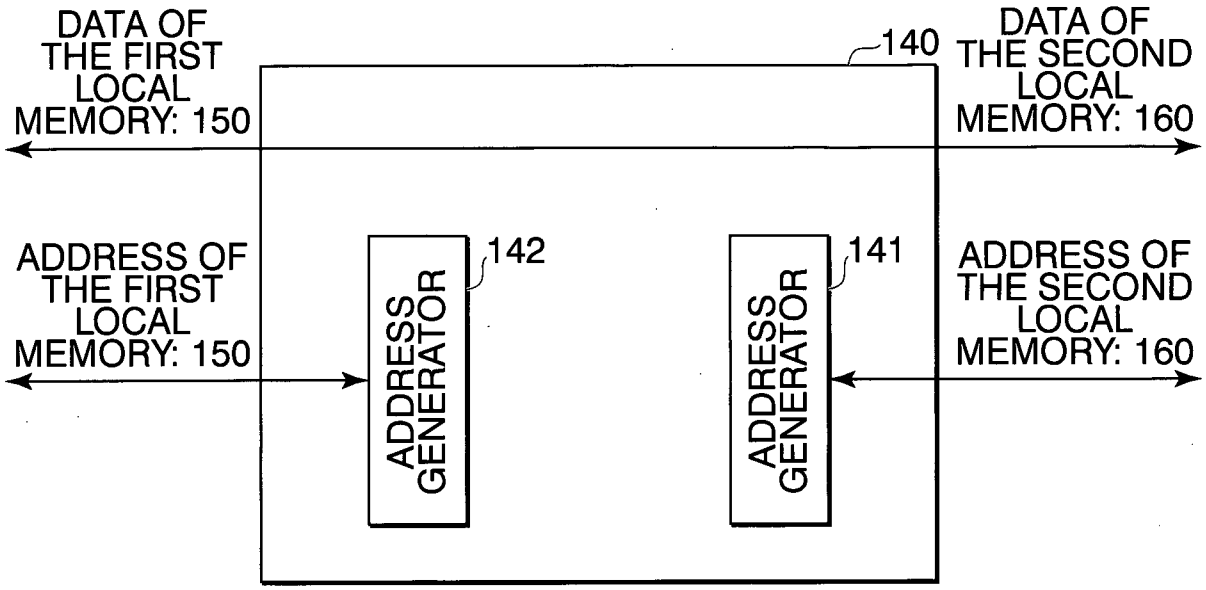

DATA OF THE FIRST LOCAL MEMORY: 150
140
DATA OF THE SECOND LOCAL MEMORY: 160
ADDRESS OF THE FIRST LOCAL MEMORY: 150
142
ADDRESS GENERATOR
141
ADDRESS GENERATOR
ADDRESS OF THE SECOND LOCAL MEMORY: 160

*FIG. 6*

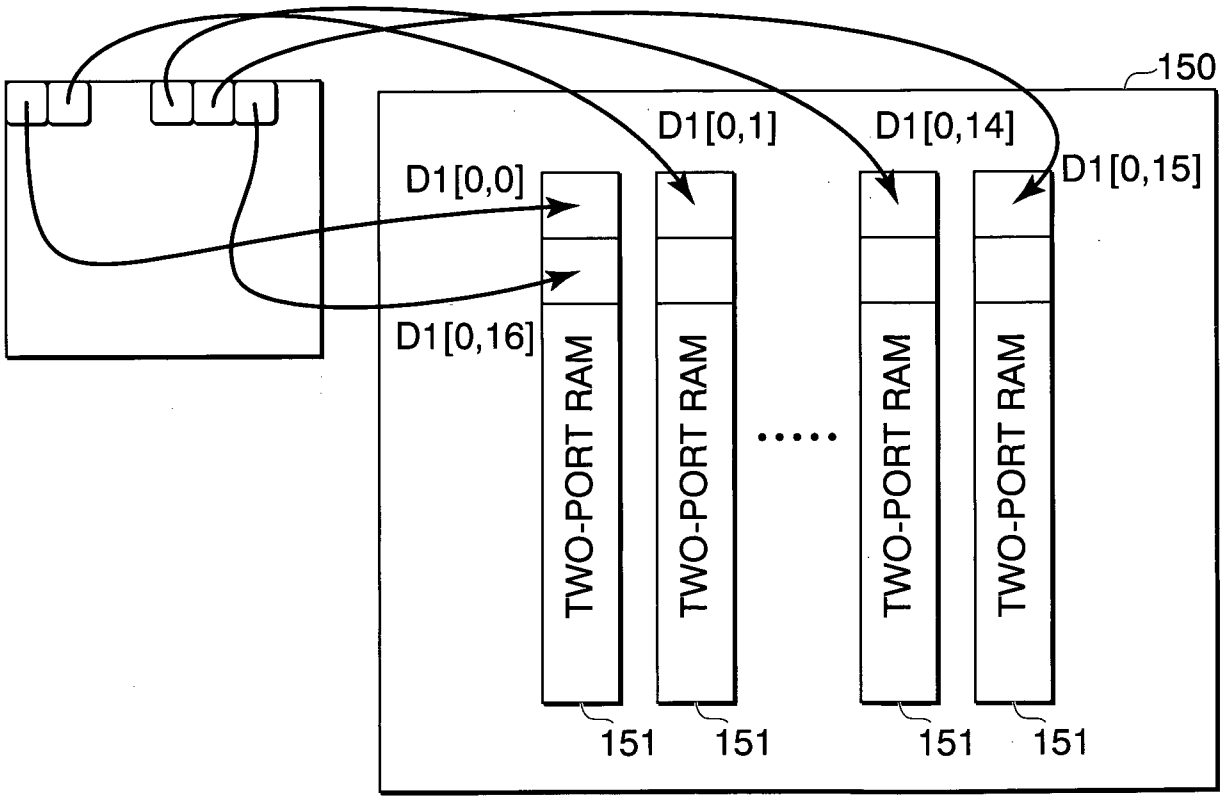

150
D1[0,0]
D1[0,1]
D1[0,14]
D1[0,15]
D1[0,16]
TWO-PORT RAM
TWO-PORT RAM
TWO-PORT RAM
TWO-PORT RAM
151
151
151
151

*FIG. 7*

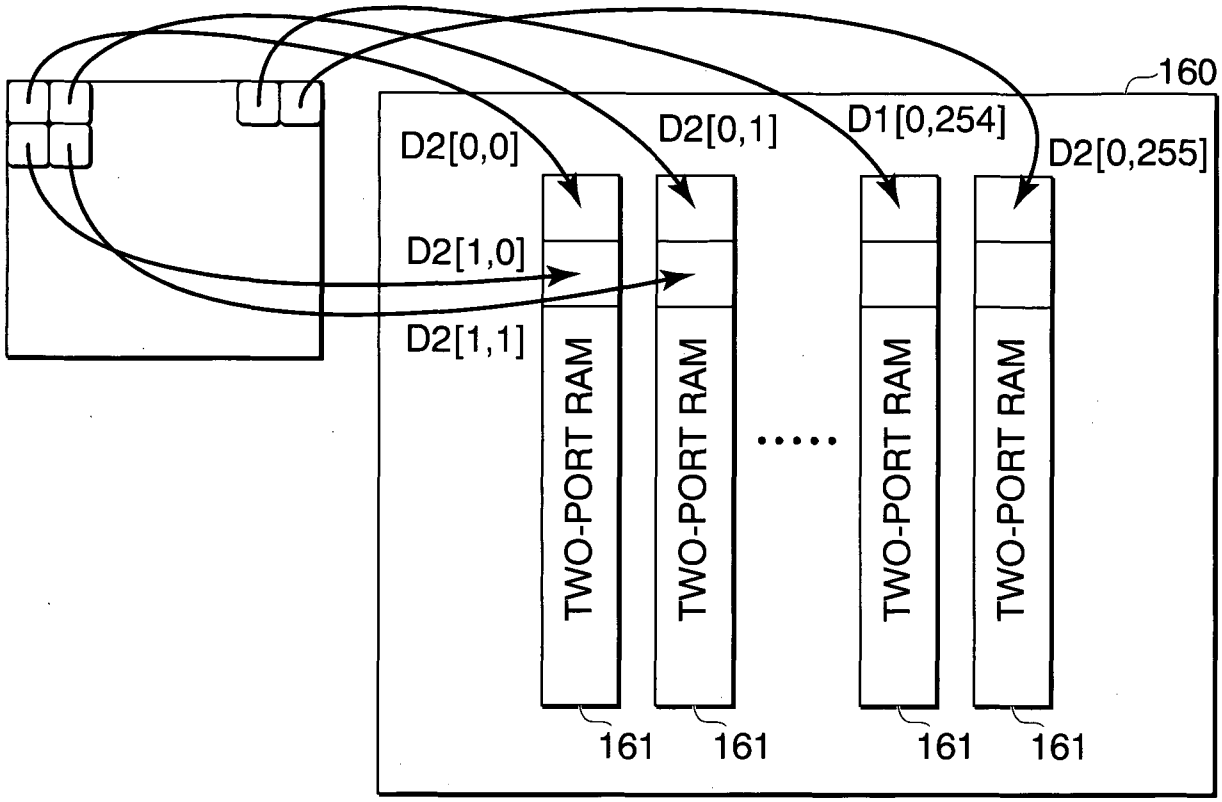

160
D2[0,0]
D2[0,1]
D1[0,254]
D2[0,255]
D2[1,0]
D2[1,1]
TWO-PORT RAM
TWO-PORT RAM
TWO-PORT RAM
TWO-PORT RAM
161
161
161
161

*FIG. 8*

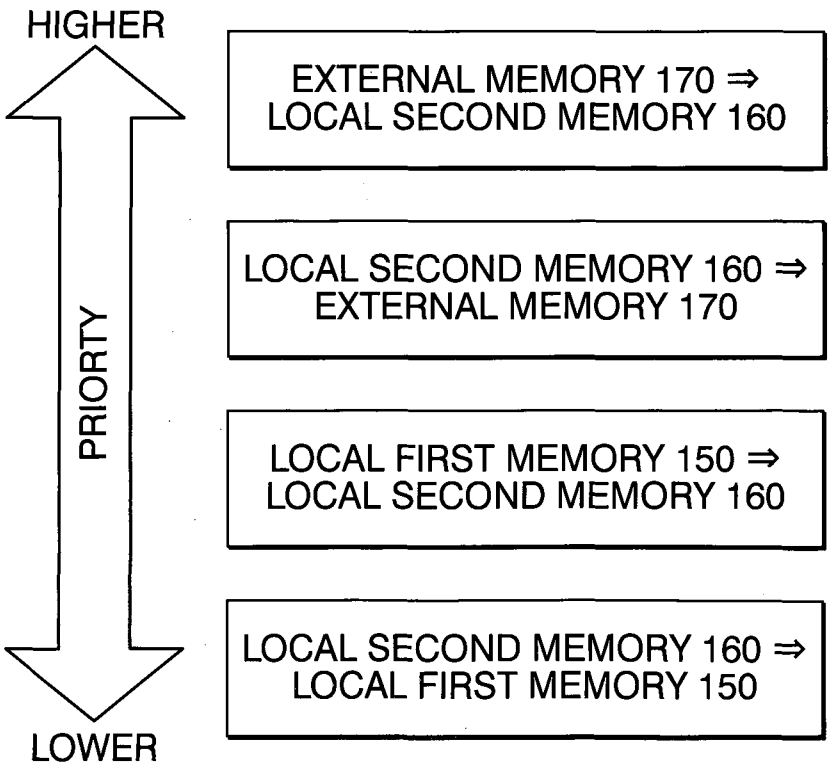

HIGHER
PRIORTY
LOWER
EXTERNAL MEMORY 170 ⇒ LOCAL SECOND MEMORY 160
LOCAL SECOND MEMORY 160 ⇒ EXTERNAL MEMORY 170
LOCAL FIRST MEMORY 150 ⇒ LOCAL SECOND MEMORY 160
LOCAL SECOND MEMORY 160 ⇒ LOCAL FIRST MEMORY 150

*FIG. 9*

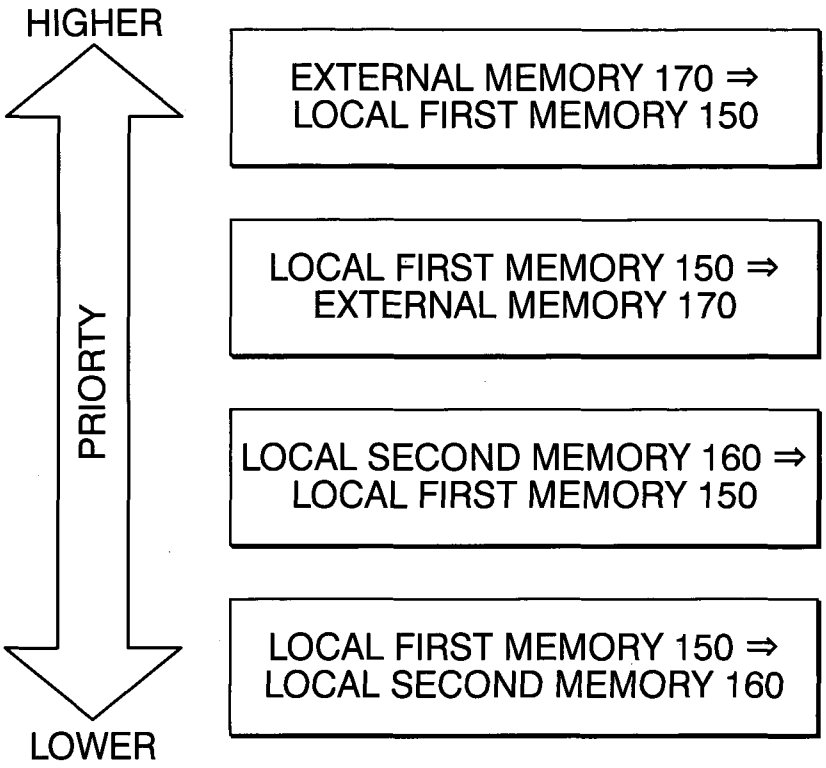

HIGHER
PRIORTY
LOWER
EXTERNAL MEMORY 170 ⇒ LOCAL FIRST MEMORY 150
LOCAL FIRST MEMORY 150 ⇒ EXTERNAL MEMORY 170
LOCAL SECOND MEMORY 160 ⇒ LOCAL FIRST MEMORY 150
LOCAL FIRST MEMORY 150 ⇒ LOCAL SECOND MEMORY 160

*FIG. 10*

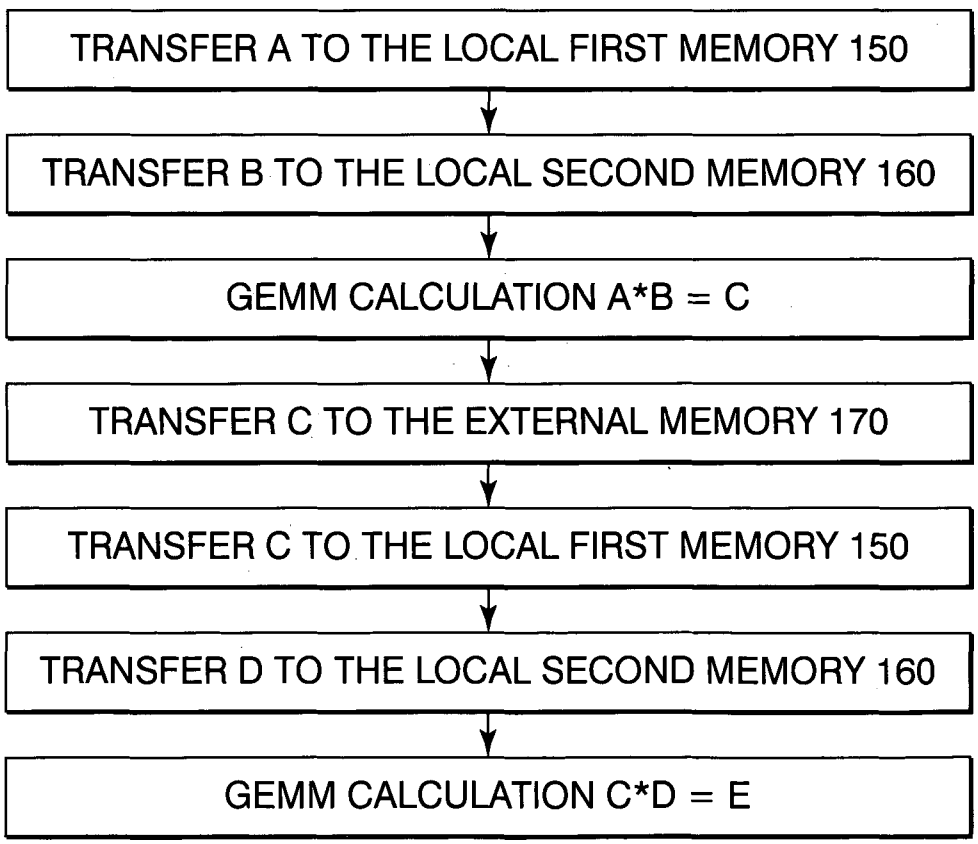

TRANSFER A TO THE LOCAL FIRST MEMORY 150
TRANSFER B TO THE LOCAL SECOND MEMORY 160
GEMM CALCULATION A*B = C
TRANSFER C TO THE EXTERNAL MEMORY 170
TRANSFER C TO THE LOCAL FIRST MEMORY 150
TRANSFER D TO THE LOCAL SECOND MEMORY 160
GEMM CALCULATION C*D = E

*FIG. 11*

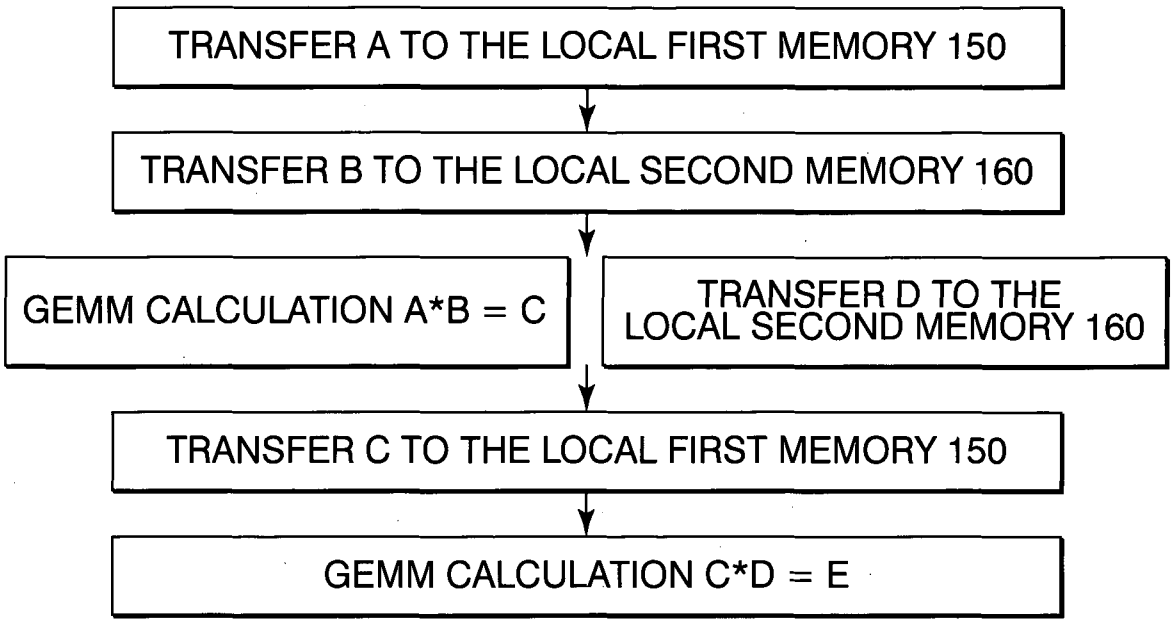

TRANSFER A TO THE LOCAL FIRST MEMORY 150
TRANSFER B TO THE LOCAL SECOND MEMORY 160
GEMM CALCULATION A*B = C
TRANSFER D TO THE LOCAL SECOND MEMORY 160
TRANSFER C TO THE LOCAL FIRST MEMORY 150
GEMM CALCULATION C*D = E

*FIG. 12*

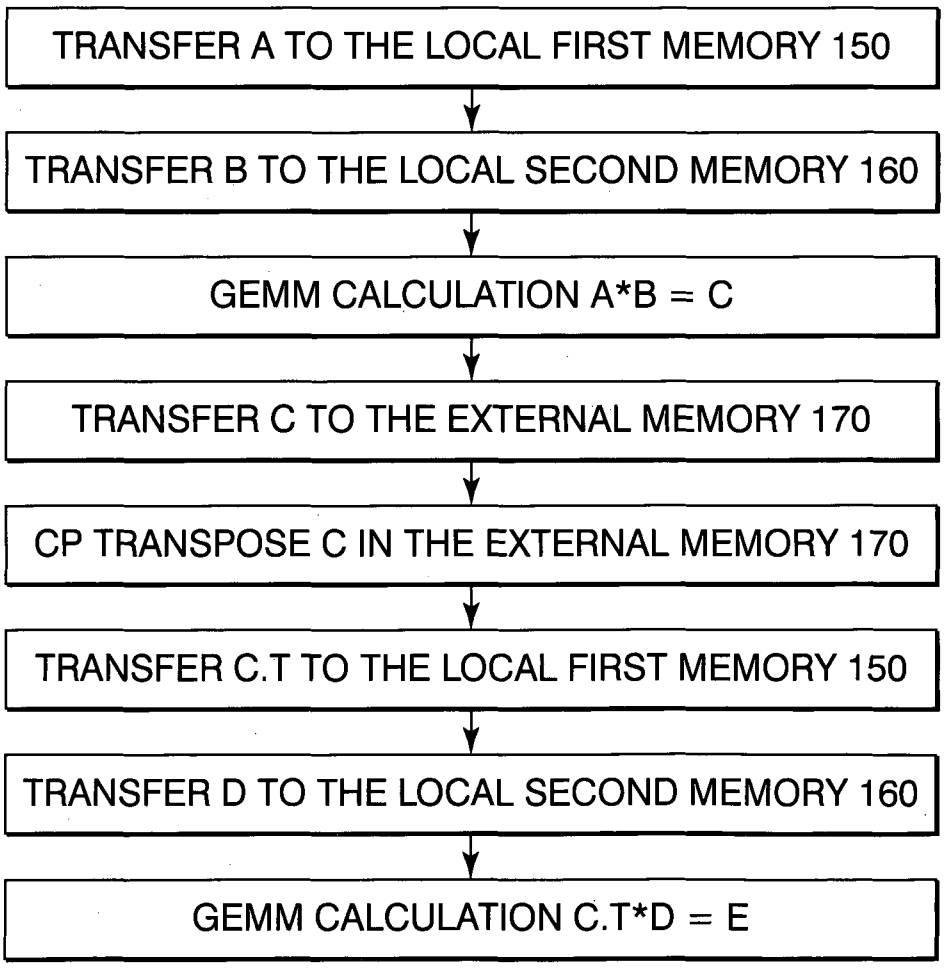

TRANSFER A TO THE LOCAL FIRST MEMORY 150
TRANSFER B TO THE LOCAL SECOND MEMORY 160
GEMM CALCULATION A*B = C
TRANSFER C TO THE EXTERNAL MEMORY 170
CP TRANSPOSE C IN THE EXTERNAL MEMORY 170
TRANSFER C.T TO THE LOCAL FIRST MEMORY 150
TRANSFER D TO THE LOCAL SECOND MEMORY 160
GEMM CALCULATION C.T*D = E

*FIG. 13*

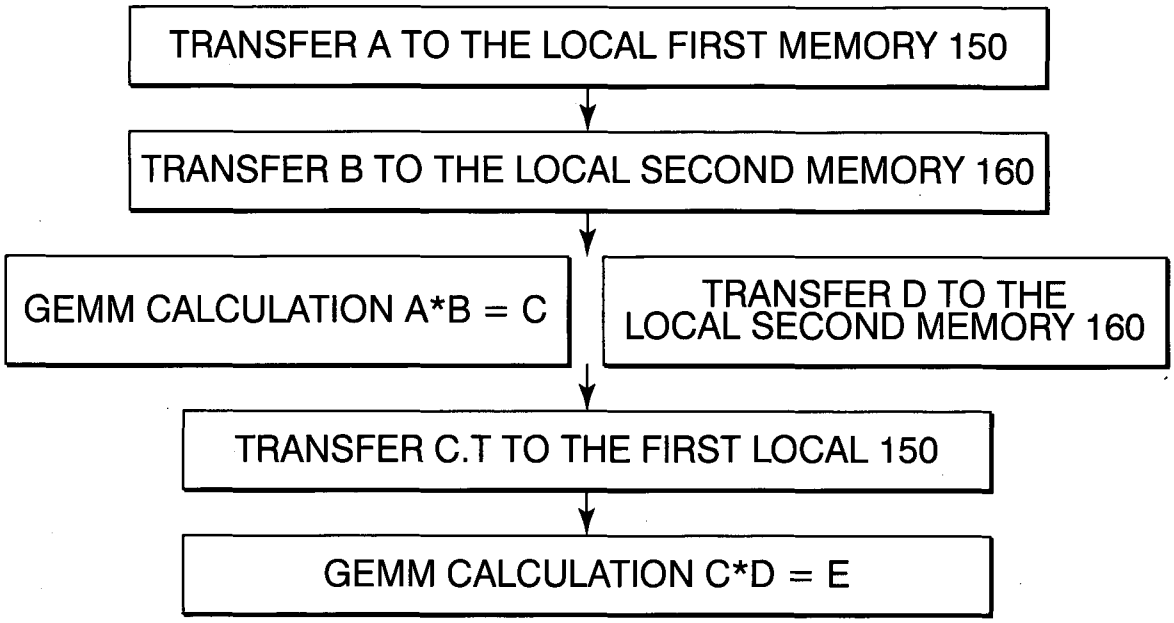

TRANSFER A TO THE LOCAL FIRST MEMORY 150
TRANSFER B TO THE LOCAL SECOND MEMORY 160
GEMM CALCULATION A*B = C
TRANSFER D TO THE LOCAL SECOND MEMORY 160
TRANSFER C.T TO THE FIRST LOCAL 150
GEMM CALCULATION C*D = E

*FIG. 14*

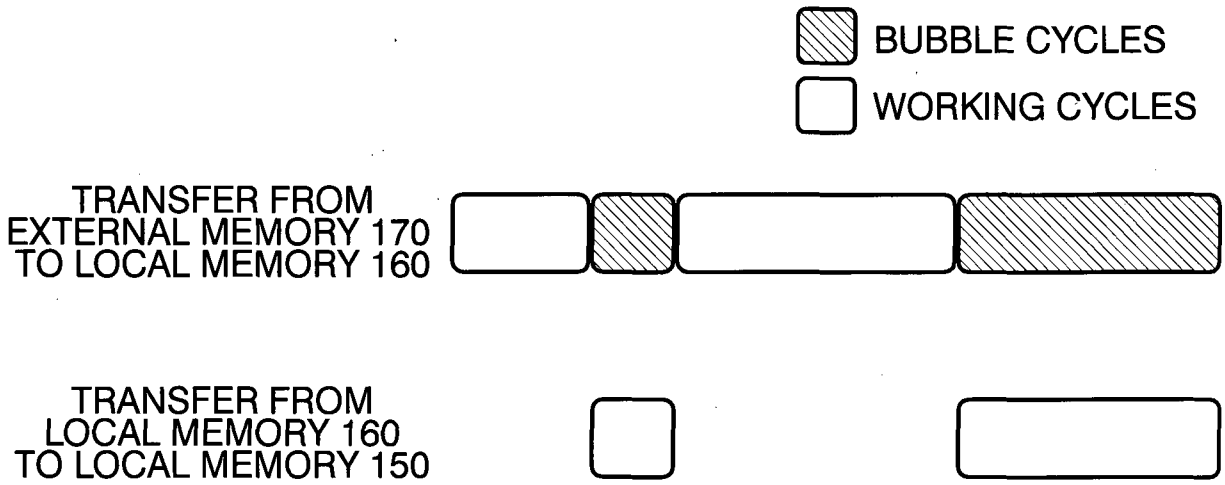
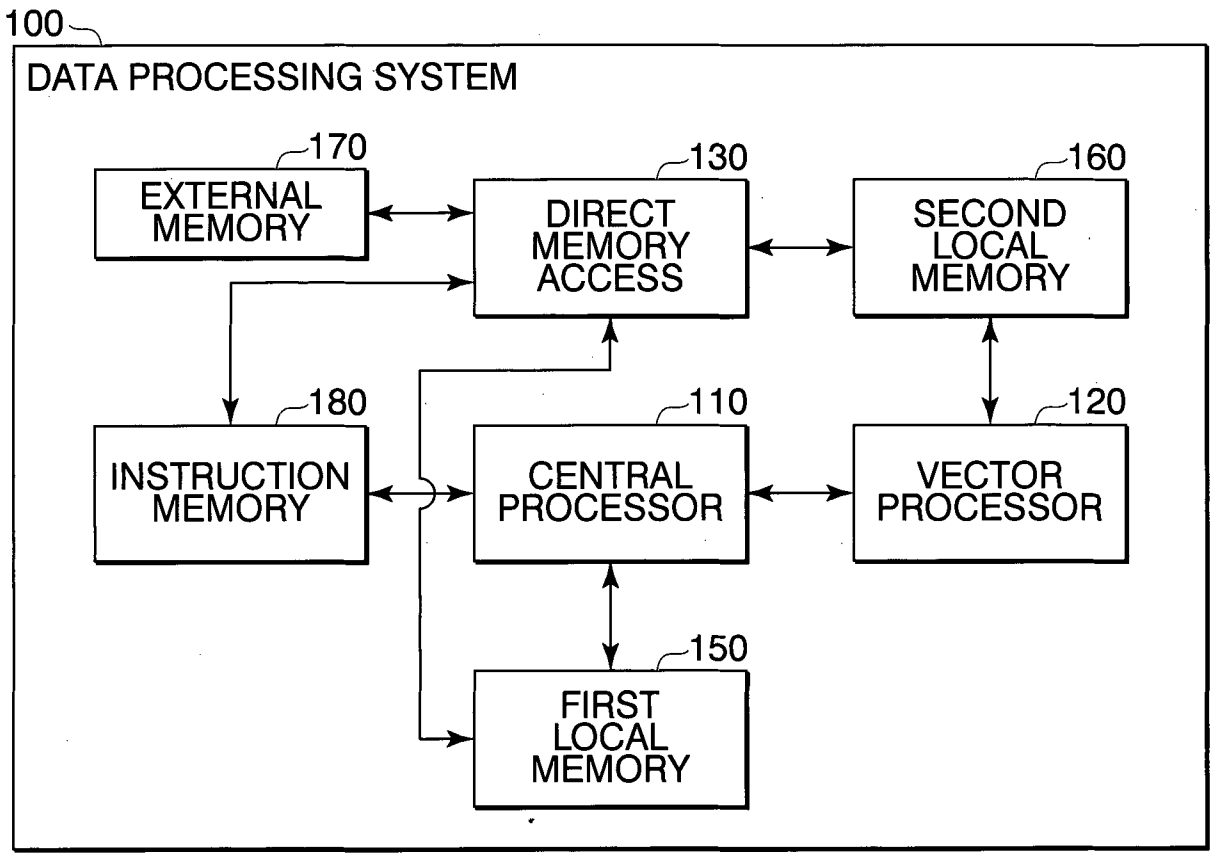

BUBBLE CYCLES
WORKING CYCLES
TRANSFER FROM EXTERNAL MEMORY 170 TO LOCAL MEMORY 160
TRANSFER FROM LOCAL MEMORY 160 TO LOCAL MEMORY 150

*FIG. 15*


100
DATA PROCESSING SYSTEM
170
EXTERNAL MEMORY
130
DIRECT MEMORY ACCESS
160
SECOND LOCAL MEMORY
180
INSTRUCTION MEMORY
110
CENTRAL PROCESSOR
120
VECTOR PROCESSOR
150
FIRST LOCAL MEMORY

# INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2018/035246

A. CLASSIFICATION OF SUBJECT MATTER

Int.Cl. G06F9/38(2006.01)i, G06F17/16(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)

Int.Cl. G06F9/38, G06F17/16

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

Published examined utility model applications of Japan 1922-1996
Published unexamined utility model applications of Japan 1971-2018
Registered utility model specifications of Japan 1996-2018
Published registered utility model applications of Japan 1994-2018

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y<br>A | JP 2008-158699 A (TOSHIBA CORPORATION) 2008.07.10, paragraphs [0009]-[0017],[0028], [0042],[0047], Fig.1 (Family: none) | 1,6-8<br>2-5 |
| Y<br>A | US 2002/0026543 A1 (TOJIMA, Masayoshi et al.) 2002.02.28, paragraphs [0084]-[0089],[0097], [0122],[0130], Fig.1 & JP 2002-41445 A & EP 1156422 A2 & KR 10-0429724 B1 | 1,6-8<br>2-5 |
| Y | JP 8-115594 A (OKI ELECTRIC INDUSTRY CO., LTD.) 1996.05.07, paragraph [0013], Fig.1 (Family: none) | 6 |

☑ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

| Date of the actual completion of the international search | Date of mailing of the international search report |
|---|---|
| 10.12.2018 | 18.12.2018 |

| Name and mailing address of the ISA/JP | Authorized officer |
|---|---|
| **Japan Patent Office**<br>3-4-3, Kasumigaseki, Chiyoda-ku, Tokyo 100-8915, Japan | OHMOMO, Yukio<br>Telephone No. +81-3-3581-1101 Ext. 3545 |

5B 6296

**INTERNATIONAL SEARCH REPORT**

International application No.
PCT/JP2018/035246

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| A | US 2003/0233511 A1 (SEIKO EPSON CORPORATION) 2003.12.18, Whole Document & JP 2003-280982 A & EP 1347387 A2 & CN 1445679 A | 2-5 |