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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 October 2014 (23.10.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/172227 A1

(51) **International Patent Classification**:
G06T 1/60 (2006.01)

(21) **International Application Number**:
PCT/US2014/033915

(22) **International Filing Date**:
14 April 2014 (14.04.2014)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
61/812,232 15 April 2013 (15.04.2013) US
13/918,892 14 June 2013 (14.06.2013) US

(71) **Applicant**: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) **Inventors**: **EGURO, Kenneth Hiroshi**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **BITTNER, Ray A., Jr.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **SMITH, George E.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **SWILLEY, Shawn Michael**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **AHMED, Rehan**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US).

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

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

[Continued on next page]

(54) **Title**: PARALLEL MEMORIES FOR MULTIDIMENSIONAL DATA ACCESS

(57) **Abstract**: The subject disclosure is directed towards loading parallel memories (e.g., in one or more FPGAs) with multidimensional data in an interleaved manner such that a multidimensional patch / window may be filled with corresponding data in a single parallel read of the memories. Depending on the position of the patch, the data may be rotated horizontally and/or vertically, for example, so that the data in each patch is consistently arranged in the patch regardless of from which memory each piece of data was read. Also described is leveraging dual ported memory for multiple line reads and/or loading one part of a buffer while reading from another.

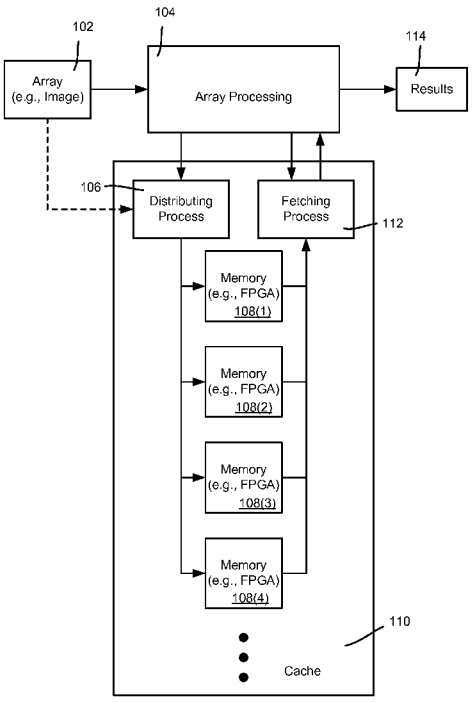

FIG. 1

WO 2014/172227 A1

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

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

PARALLEL MEMORIES FOR MULTIDIMENSIONAL DATA ACCESS

BACKGROUND

[0001] Standard image and signal processing algorithms generally process data within a sample window (e.g. a Gaussian blur). If this window "slides" neatly in raster order, producing this sample window is relatively straightforward and, to a large degree, does not become markedly more difficult as the sample window increases in size (e.g. larger radius Gaussian blurs).

[0002] While this works for brute-force algorithms, algorithms that work on large data sets often need perform more specific and targeted computations (for the sake of computational efficiency). In this case, the sample windows that are computed upon are generally arbitrarily-located windows within a larger search space. A similar situation occurs if the analysis is data-dependent. Producing sample windows for these more advanced algorithms is a much more difficult problem.

[0003] One solution stores the data representing the larger search space in a buffer and uses a series of random memory accesses into that buffer to gradually construct the required sample. However, this solution creates a bottleneck at the memory, limiting the speed of computation. This is because real memory has limited I/O capacity (i.e. simultaneous read ports), whereby the speed at which "full" samples can be accessed is inversely related to the size of the sample window; (e.g., algorithms with very large sample windows simply cannot run as quickly as those with smaller windows). Another issue is that unless the sample is very small, the entire sample cannot be accessed at once. This strongly limits the amount of parallel computation that can be performed. Notwithstanding, this is the solution that is implemented on a CPU.

[0004] An alternative solution avoids the memory bottleneck by creating multiple identical copies of the larger search space using multiple buffers. In this way, parallel data items are produced within the requested window, only limited by the number of parallel buffers. This solution is amenable to a direct hardware implementation. However, having multiple copies comes at a significant resource cost, as essentially the amount of memories needed is proportional to the window size if normalized to a constant performance requirement.

SUMMARY

[0005] This Summary is provided to introduce a selection of representative concepts in a simplified form that are further described below in the Detailed Description. This

Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used in any way that would limit the scope of the claimed subject matter.

[0006] Briefly, one or more of various aspects of the subject matter described herein are directed towards distributing multidimensional data among memories such that a patch / window of the multidimensional data is able to filled in parallel data read operations. The number of memories is determined based upon a product of a length of each dimension of the patch that is used in processing the multidimensional data. The memories are read to fill a positioned patch of data with the multidimensional data associated with a position of the patch via a parallel read of each of the memories.

[0007] In one or more aspects, a distribution process is configured to determine a number of memories based upon multidimensional patch dimensions. The distribution process loads the memories with data from a multidimensional array in an interleaved manner, in which the interleaving provides that any patch of data (that corresponds to the multidimensional patch dimensions) that is filled by reading the memories has each data access unit (one or more items read or written together as a unit) read from a different memory.

[0008] One or more aspects are directed towards loading multidimensional data into a plurality of memories, determining addresses in each of the memories based upon a position of a window, and filling a data window with a single parallel read of the plurality of memories. Data corresponding to the window data is output, and the process repeated for different window positions. The window data may be rotated to provide the data corresponding to the window data.

[0009] Other advantages may become apparent from the following detailed description when taken in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The present invention is illustrated by way of example and not limited in the accompanying figures in which like reference numerals indicate similar elements and in which:

[0011] FIGURE 1 is a block diagram representing example components that may be used to load multidimensional data for parallel reading, including into and from field programmable gate array (FPGA) memories, according to one or more example implementations.

[0012] FIGS. 2A and 2B are representations of how data may be loaded into memories in an interleaved manner for subsequent parallel reading, according to one or more example implementations.

[0013] FIG. 3 is a representation of a window of data being filled by parallel memory reads, according to one or more example implementations.

[0014] FIG. 4 is a representation of a window of data being filled by parallel memory reads, in which horizontal rotation of the data is performed based upon the window position to provide a consistent return pattern, according to one or more example implementations.

[0015] FIG. 5 is a representation of a window of data being filled by parallel memory reads, in which horizontal and vertical rotation of the data is performed based upon the window position to provide a consistent return pattern, according to one or more example implementations.

[0016] FIG. 6 is a representation of how a subset of multidimensional data may be buffered, according to one or more example implementations.

[0017] FIG. 7 is a representation of how memories may be arranged with section-based offsets, according to one or more example implementations.

[0018] FIG. 8 is a representation of how two lines of a memory may be read, according to one or more example implementations.

[0019] FIG. 9 is a flow diagram representing example steps taken to load memories in an interleaved manner for parallel reading, according to one or more example implementations.

[0020] FIG. 10 is a flow diagram representing example steps taken to read patch / window data in parallel and if necessary rotate the data, according to one or more example implementations.

[0021] FIG. 11 is a block diagram representing an exemplary non-limiting computing system or operating environment, in the form of a gaming system, into which one or more aspects of various embodiments described herein can be implemented.

DETAILED DESCRIPTION

[0022] Various aspects of the technology described herein are generally directed towards dividing data to be processed among separate memories (comprising a "patch cache"), each memory holding a different, but interleaved, portion of the data. The interleaving is based upon the data to be processed (such as image data or other real-world

sampled data) being physically or time adjacent, e.g., pixels in an image are adjacent other ones.

[0023] The division and interleaving (round-robin) are based upon the dimensions of the patch (e.g., window size in two-dimensional data processing). When a patch is needed for processing, the data are arranged such that each access into the "patch cache" needs to get one and only one value from each memory within the cache. This provides fast single-cycle access and a large degree of aggregate parallel bandwidth.

[0024] In general, the technology described herein provides a memory architecture that capitalizes on the natural physical spatial locality of the image or other real-world data to maintain high performance without duplication. This allows extremely high performance with little resource overhead.

[0025] It should be understood that any of the examples herein are non-limiting. For instance, benefits are readily apparent in hardware / FPGA / ASIC scenarios, however the technology may be used in other scenarios. Further, two-dimensional image data are used in some of the examples to help convey the concepts in a way that is relatively easy to understand, however image data is only one type of data, and other types of data, including in more than two dimensions, may benefit from the technology described herein. As such, the present invention is not limited to any particular embodiments, aspects, concepts, structures, functionalities or examples described herein. Rather, any of the embodiments, aspects, concepts, structures, functionalities or examples described herein are non-limiting, and the present invention may be used various ways that provide benefits and advantages in data processing and/or connected components in general.

[0026] FIG. 1 shows an example system in which a multidimensional array 102 of data to be processed, (e.g., two-dimensional image data) is processed by an array processing component 104. As described herein, for efficient access, the array processing component 104 is coupled to a data distributing process 106 that writes the array data into a plurality of independent (e.g., FPGA) memories 108(1) – 108(4), e.g., collectively arranged as a patch cache 110. The distributing process is shown as being coded into the cache 110, but may be a separate process, and may be incorporated into the array processing component. Note that only four independent memories 108(1) – 108(4) are shown in FIG. 1, but that any practical number may be used, and that the number depends on the dimensions of the patch.

[0027] To process the data, the array processing component 104 is coupled to a data fetching process 112 that reads a patch of data in parallel from the patch cache 110 /

independent memories 108(1) – 108(4). The array processing component 104 processes each patch, and uses the processing on one or more patches to ultimately provide results 114. Note that the array processing component also may be in hardware, e.g., the patch cache.

[0028] To produce arbitrarily located P-sized "patches" from the buffered data, the data is divided among separate memories, each holding a different, but interleaved, portion of the data. The total memory is thus divided among the separate memories; e.g., if a single, serially accessed memory held the data in D space, each of the divided, parallel memories hold D / P of the data.

[0029] The dimensions of the data may be enumerated as N, N', N'' and so forth, up to the number of data dimensions. Each of the dimensions of the patch may be enumerated as P, P', P'' and so forth. Internally, the patch cache 110 is organized as an array of independent memories. The number of independent memories (M) is the product of each of the lengths of dimensions in the patch.

[0030] Turning to an example, FIG. 2A shows an arrangement to produce a 2x2 patch into a two-dimensional 4x4 space. That is, the patch has dimensions P and P' (each of length two), while the data has dimensions N and N' (each of length four). The sixteen data items are alphabetically represented in FIG. 2A by letters A through P, and by the array indices (0, 0) to (3, 3).

[0031] In FIG. 2A and also shown in FIG. 2B, because the patch is 2x2, M = 2x2 = 4 memories are used, denoted as MemW, MemX, MemY and MemZ. If the overall buffer can hold B values over the entire dimensional space, each memory hold B / M values. In this example of FIGS. 2A and 2B, the buffer size B = 16, whereby each memory holds 4 values (FIG. 2B).

[0032] Data may be written into the cache in raster order, organized by some dimension from N to N' to N'' and so forth. These writes are generally low width (e.g., in FIG. 2A only one value wide), but wider data may be accepted. For example, data may come in as one-dimensional strips. A system may accept small length strips; this may be expanded to full-width N length strips or even higher order multi-dimensional strips.

[0033] The cache 110 accepts the data and writes it in a round-robin style by dimension. For example, if data arrives in raster-order, first along dimension N, then by dimension N', and so forth, the data for each dimension is written into each of the dimensions in the patch array in turn. This proceeds in a round-robin manner among the first dimension of the patch array across the entire length of first data dimension, wrapping on the first

dimension of the patch array. Subsequent data along progressively higher dimensions of the data are distributed round-robin across progressively higher dimensions of the patch array, again wrapping each dimension of the patch array. If the dimension order of the data is higher than the dimension order of the patch, the round-robin ordering restarts at the first dimension of the patch. This distribution (e.g., for three dimensions) may be represented as:

$$column = column\ address\ mod\ P$$
$$row = row\ address\ mod\ P'$$
$$depth = depth\ address\ mod\ P''$$

[0034] For example, in FIG. 2A, the MemW and MemX memories are in the first row of the dimension PxP' array (along the P dimension) and the MemY and MemZ memories are in the second row of the dimension PxP' array (again, along the P dimension but this time in the next P' dimension row).

[0035] Because in this example the length of the N dimension of the data is four, and the P dimension of the patch is two, the first two data points A and B enter the cache and are placed into the MemW and MemX memories, respectively. When the third data point C enters the cache, the length of the P dimension has been exhausted, but the N dimension has not. Thus, the P dimension will wrap around and the third and fourth data points C and D are placed in the MemW and MemX memories, respectively.

[0036] At this point, the N dimension has been exhausted, so the fifth, sixth, seventh, and eighth data points (E - H) enter the MemY, MemZ, MemY, MemZ memories in a similar fashion.

[0037] At this time, both the N dimension and P' dimension have been exhausted, but the N' dimension has not, whereby the ninth, tenth, eleventh and twelfth data points (I – L) wrap around along the P' dimension back to the MemW and MemX memories. This continues to the end of the NxN' data, e.g., M- P are written to the MemY and MemZ memories in the example of FIGS. 2A and 2B.

[0038] Note that to maintain a 2x2 patch but with a third data dimension (e.g. the input data were a 2D image over time), and the desire is to get any 2x2 patch of the image from any time slice across the N'' dimension (e.g., the dimension of the patch array is smaller than the dimension of the data), the second, third and so forth time slices across the N''

dimensions wrap back to the first P dimension in the same MemW, MemX, MemW, MemX, MemY, MemZ, MemY, MemZ, MemW, MemX … arrangement.

[0039] When the data is distributed and needs to be read back from the memory as a "PxP'xP''..." patch, the output from each memory can be re-arranged along each dimension to a consistent orientation. For example, as shown in FIGS. 3 - 5, the top left pixel in any 2x2 patch (represented by the dashed boxes) may come from any of the four memories in the array. Thus, if the data are returned in order from MemW, MemX, MemY, MemZ, the order does not correspond to the patch order. For example, in FIG. 4, B, C, F, G is desired as the patch data, however the order corresponding to MemW, MemX, MemY and MemZ is C, B, G and F.

[0040] For some usage scenarios, the order is irrelevant, e.g., if the array processing component 104 is simply summing the returned values. However, other applications expect the data to be returned in a consistent manner, e.g., top left, top right, and lower left, lower right.

[0041] Assuming that the outputs from the memories remain static, these values may need to be rotated along each axis (dimension P and then subsequently by dimension P') to ensure that the top left pixel of each resulting patch remains in a consistent location, and so on.

[0042] As can be seen, the data 330 from the patch 332 in FIG. 3 needs no reordering. The data 440 from the patch 442 in FIG. 4 needs a horizontal shift of each row to obtain the ordered data 444, namely B, C, F, G.

[0043] The data 550 in patch 552 in FIG. 5 needs both horizontal shifting (data 554) and vertical shifting (data 556) to get F, G, J, and K. Note further that the row data in FIG. 5 needs to be accessed via an offset to account for the next vertical row, e.g., memory MemW is arranged as A (0,0), C (2, 0), I (0, 2) and K (2,2). As used herein, with the patch window's row height being two in this example, MemW is considered as having two sections that correspond to this offset, namely section 0 containing A (0,0) and C (2, 0), and section 1 containing I (0, 2) and K (2,2). This section-based offset addressing allows the patch window to be filled with the correct data when positioned at any row, as in FIG. 5.

[0044] Rotation may be efficiently accomplished by a series of shift registers. The rotations (e.g., in two dimensions) for any patch (window) dimensions are determined according to:

$$X_{Rot} = X \,\%\, A_{wX}$$

$$Y_{Rot} = Y \,\%\, A_{wY}$$

where % indicates modulo and A_{wX} and A_{wY} define the access window, that is, the patch dimensions, and X and Y are the starting coordinates of the patch.

[0045] An entire array (e.g., a full set of image data) need not be put into the cache at the same time. For example, as generally shown in FIG. 6, a part (e.g., a band) of an image 660 may be written to the memories, and read back and processed. In FIG. 6, the band being read is between Y_{LOW} and Y_{HIGH}, indicated by the dashed horizontal lines. In general, the patch / access window 662 needs to be able to be positioned (aligned on pixels) anywhere in the band, and is defined by A_{wX} (patch width) and A_{wY} (patch height), which, for example, may be the 2x2 window in the above examples. As can be readily appreciated, in a sliding window scenario that never moves upwardly, once the window moves down a line, that line is freed and may be overwritten with more data to process, as long as the wrapping / circular buffer situation is tracked (as the next lower line in the image is now above the sliding window in the buffer).

[0046] In some implementations having dual ported memory, one part of the memories may be written while reading from another part. Thus, as a line is freed, it may be written while the next line is being processed. In a non-dual ported memory scenario, the reading needs to pause when new writes are needed. Also, as described below, with dual ported memory there may be times when both ports are being used for reads; if this is not the case, reads and writes can occur on the same cycle. However, this opportunity may not occur, or the writes may fall behind the reads, whereby some pausing of the reads needs to occur.

[0047] FIG. 7 shows each memory is divided into sections, e.g., sec 0 or sec 1, to provide the offset into the data to match the patch's vertical row position as described above. To this end, the section number is calculated from the *Y*-coordinate in order to give the starting address for the part of the memory that contains the line being accessed. In other words, the section number, and thus the address in the memory is based upon the *Y*-coordinate; to find an address in the memory for a section of a certain size (sec size), the following may be used:

$$sec\# = \frac{Y - Y_{LOW+} A_{wY} - 1}{A_{wY}} \,\%\, A_{wY}$$

$$ADDR = (sec\#) \cdot (\text{sec}\, size) + \frac{X}{A_{wX}}$$

[0048] FIG. 8 shows another alternative, in which instead of one item of data (e.g., pixel) being returned for a read, two pixels are returned as a unit. This may be because the pixels are eight bits wide and the memory is sixteen bits wide, for example, so two pixels are read at once. Note that as used herein, the term "access unit" refers to whatever reading and writing scheme is in place, e.g., one-byte reads, two-byte reads, four-byte reads and so on. Note that the scheme in use affects the number of memories needed; for example, if the patch is 4x4 and each memory provides two pixels, then two memories across and four memories down are needed.

[0049] However, as represented by the dashed lines in FIG. 8, using a 4x4 access window as an example, at times the window is positioned such that more than one line needs to be read from a memory at once to fill the window. This is not a problem with dual ported memory which allows fetching two lines out of one memory; the two lines are concatenated and any rotation handled as described above. However, if not dual ported, two clock cycles are needed. Further, as set forth above, if dual ported but one port is being used for simultaneous writing, then either the write needs to be paused to allow the dual read, or two clock cycles are needed for the read.

[0050] FIG. 9 is a flow diagram showing example steps for interleaving data among memories based upon a window (patch) size. FIG. 9 is generally described with respect to two dimensions, and assumes that the data will fit in the memories (whether as a whole or via a band at a time as described above). Step 902 obtains the patch dimensions, e.g., as part of a setup process performed by the array processing algorithm. Step 904 represents allocating memories according to the patch dimensions, e.g., a 2x2 patch has four memories, a 3x3 nine, a 4x4 sixteen, and so on.

[0051] Step 906 selects the first dimension of data, e.g., the *X*-dimension starting at coordinate zero. Step 908 selects the memories based upon the *X*- dimension, such as the first two of four memories for a 2x2 patch, the first three for a 3x3 patch, and so on.

[0052] Step 910 represents the interleaving of the data along the *X*-axis among the selected memories, e.g., alternating between them. Note that the data wraps in the selected memories as needed, as described above. This continues until the first dimension is exhausted, that is, the entire line is placed in the selected memories.

[0053] When the first dimension is exhausted, step 914 evaluates whether the second dimension is exhausted, e.g., the last row has been placed into the memories. If not, at step 916 the first dimension is "reset" (e.g., the *X*-coordinate returns to zero) and the next dimension incremented, e.g., the *Y*-coordinate is moved to the next line.

[0054] Step 908 selects the next memories, e.g., not the ones used previously. For example, with a 2x2 patch, every other row is placed into a different pair of the memories; for a 3x3 patch, every third row into a different set of three memories, and so on. In this way, every value in a window is in a different memory.

[0055] The process continues alternating among memories along the columns until the first dimension (row) is exhausted, and alternating among memories along the rows until all rows are exhausted. At this time, the memory is ready for reading. Note that as described above, if a sliding window scenario is in use, reading may begin as soon as enough lines to fill a patch with data have been written. If the window is allowed to be positioned anywhere in the buffer at any time, then the buffer needs to be filled.

[0056] FIG. 10 represents reading the data, beginning at step 1002 where the window data (e.g., the starting coordinate and size) are received. Note that in a sliding window scenario, the logic of FIG. 10 may simply receive a "next position" command and move the window horizontally until it needs to move down to the next line.

[0057] Step 1004 represents computing the address in each memory for the data points in the access window, e.g., using the address computations described above. Note that rather than the full computation, in a sliding window scenario the previous computation may be used to determine the next location in each memory because the window position and underlying memory changes regularly.

[0058] Step 1006 reads the memories at their respective addresses, in parallel, into a set of shift registers or the like. As described above, step 1008 performs any needed X rotation, and step 1010 any needed Y rotation. At this time, the window is output, filled with the correct data in the correct order.

EXAMPLE OPERATING ENVIRONMENT

[0059] FIG. 11 illustrates an example of a suitable computing and networking environment / system 1100 into which computer-related examples and implementations described herein may be implemented, for example. As one example, the computing and networking environment 1100 may program an FPGA with data and/or logic to perform multidimensional array processing as described herein, provide input data (e.g., capture images), receive output data, and so forth. Notwithstanding, the computing and networking environment 1100 also may implement the technology described in FIGS. 1 - 10 in software, at least in part.

[0060] It can be readily appreciated that the above-described implementation and its alternatives may be implemented on any suitable computing device, including a gaming

system, personal computer, tablet, DVR, set-top box, smartphone and/or the like. Combinations of such devices are also feasible when multiple such devices are linked together. For purposes of description, a gaming (including media) system is described as one exemplary operating environment hereinafter.

[0061] FIG. 11 is a functional block diagram of an example gaming and media system 1100 and shows functional components in more detail. Console 1101 has a central processing unit (CPU) 1102, and a memory controller 1103 that facilitates processor access to various types of memory, including a flash Read Only Memory (ROM) 1104, a Random Access Memory (RAM) 1106, a hard disk drive 1108, and portable media drive 1109. In one implementation, the CPU 1102 includes a level 1 cache 1110, and a level 2 cache 1112 to temporarily store data and hence reduce the number of memory access cycles made to the hard drive, thereby improving processing speed and throughput.

[0062] The CPU 1102, the memory controller 1103, and various memory devices are interconnected via one or more buses (not shown). The details of the bus that is used in this implementation are not particularly relevant to understanding the subject matter of interest being discussed herein. However, it will be understood that such a bus may include one or more of serial and parallel buses, a memory bus, a peripheral bus, and a processor or local bus, using any of a variety of bus architectures. By way of example, such architectures can include an Industry Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA) local bus, and a Peripheral Component Interconnects (PCI) bus also known as a Mezzanine bus.

[0063] In one implementation, the CPU 1102, the memory controller 1103, the ROM 1104, and the RAM 1106 are integrated onto a common module 1114. In this implementation, the ROM 1104 is configured as a flash ROM that is connected to the memory controller 1103 via a Peripheral Component Interconnect (PCI) bus or the like and a ROM bus or the like (neither of which are shown). The RAM 1106 may be configured as multiple Double Data Rate Synchronous Dynamic RAM (DDR SDRAM) modules that are independently controlled by the memory controller 1103 via separate buses (not shown). The hard disk drive 1108 and the portable media drive 1109 are shown connected to the memory controller 1103 via the PCI bus and an AT Attachment (ATA) bus 1116. However, in other implementations, dedicated data bus structures of different types can also be applied in the alternative.

[0064] A three-dimensional graphics processing unit 1120 and a video encoder 1122 form a video processing pipeline for high speed and high resolution (e.g., High Definition) graphics processing. Data are carried from the graphics processing unit 1120 to the video encoder 1122 via a digital video bus (not shown). An audio processing unit 1124 and an audio codec (coder/decoder) 1126 form a corresponding audio processing pipeline for multi-channel audio processing of various digital audio formats. Audio data are carried between the audio processing unit 1124 and the audio codec 1126 via a communication link (not shown). The video and audio processing pipelines output data to an A/V (audio/video) port 1128 for transmission to a television or other display / speakers. In the illustrated implementation, the video and audio processing components 1120, 1122, 1124, 1126 and 1128 are mounted on the module 1114.

[0065] FIG. 11 shows the module 1114 including a USB host controller 1130 and a network interface (NW I/F) 1132, which may include wired and/or wireless components. The USB host controller 1130 is shown in communication with the CPU 1102 and the memory controller 1103 via a bus (e.g., PCI bus) and serves as host for peripheral controllers 1134. The network interface 1132 provides access to a network (e.g., Internet, home network, etc.) and may be any of a wide variety of various wire or wireless interface components including an Ethernet card or interface module, a modem, a Bluetooth module, a cable modem, and the like.

[0066] In the example implementation depicted in FIG. 11, the console 1101 includes a controller support subassembly 1140, for supporting four game controllers 1141(1) - 1141(4). The controller support subassembly 1140 includes any hardware and software components needed to support wired and/or wireless operation with an external control device, such as for example, a media and game controller. A front panel I/O subassembly 1142 supports the multiple functionalities of a power button 1143, an eject button 1144, as well as any other buttons and any LEDs (light emitting diodes) or other indicators exposed on the outer surface of the console 1101. The subassemblies 1140 and 1142 are in communication with the module 1114 via one or more cable assemblies 1146 or the like. In other implementations, the console 1101 can include additional controller subassemblies. The illustrated implementation also shows an optical I/O interface 1148 that is configured to send and receive signals (e.g., from a remote control 1149) that can be communicated to the module 1114.

[0067] Memory units (MUs) 1150(1) and 1150(2) are illustrated as being connectable to MU ports "A" 1152(1) and "B" 1152(2), respectively. Each MU 1150 offers additional

storage on which games, game parameters, and other data may be stored. In some implementations, the other data can include one or more of a digital game component, an executable gaming application, an instruction set for expanding a gaming application, and a media file. When inserted into the console 1101, each MU 1150 can be accessed by the memory controller 1103.

[0068] A system power supply module 1154 provides power to the components of the gaming system 1100. A fan 1156 cools the circuitry within the console 1101.

[0069] An application 1160 comprising machine instructions is typically stored on the hard disk drive 1108. When the console 1101 is powered on, various portions of the application 1160 are loaded into the RAM 1106, and/or the caches 1110 and 1112, for execution on the CPU 1102. In general, the application 1160 can include one or more program modules for performing various display functions, such as controlling dialog screens for presentation on a display (e.g., high definition monitor), controlling transactions based on user inputs and controlling data transmission and reception between the console 1101 and externally connected devices.

[0070] The gaming system 1100 may be operated as a standalone system by connecting the system to high definition monitor, a television, a video projector, or other display device. In this standalone mode, the gaming system 1100 enables one or more players to play games, or enjoy digital media, e.g., by watching movies, or listening to music. However, with the integration of broadband connectivity made available through the network interface 1132, gaming system 1100 may further be operated as a participating component in a larger network gaming community or system.

CONCLUSION

[0071] While the invention is susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in the drawings and have been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific forms disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions, and equivalents falling within the spirit and scope of the invention.

CLAIMS

1. A method comprising, distributing multidimensional data among memories, in which a number of memories are determined based upon a product of a length of each dimension of a patch used in processing the multidimensional data, and reading the memories to fill a positioned patch of data with the multidimensional data that corresponds to a position of the patch via a parallel read of each of the memories.

2. The method of claim 1 wherein the distributing the multidimensional data comprises a) loading a band of the multidimensional data into a buffer comprising the memories, or b) alternating writing data corresponding to a first dimension of the data to a first set of memories corresponding to a first dimension of the patch, or c) alternating writing data corresponding to a first dimension of the data to a second set of second memories corresponding to a second dimension of the patch, or any combination of a), b) or c).

3. The method of claim 1 wherein the memories comprise dual ported memories, and a) wherein reading the memories comprises reading data from two addresses in a single cycle, or b) further comprising writing to a memory address of a memory while reading from a different or same memory address of the memory, or c) both a) and b).

4. The method of claim 1 further comprising, (a) rotating data in the patch zero or more times in a vertical rotation into a consistent order based upon a position of the patch, and returning the patch after any rotating of the data, or (b) rotating data in the patch zero or more times in a horizontal rotation and at least once in a vertical rotation into a consistent order based upon a position of the patch, and returning the patch after any rotating of the data, or both a) and b).

5. A system comprising, a distribution process configured to determine a number of memories based upon multidimensional patch dimensions, the distribution process further configured to load the memories with data from a multidimensional array in an interleaved manner, in which the interleaving provides that any patch of data corresponding to the multidimensional patch dimensions that is filled by reading the memories has each data access unit read from a different memory.

6. The system of claim 5 wherein the memories are contained in a single field programmable gate array, or wherein the memories are distributed among a plurality of field programmable gate arrays.

7. The system of claim 5 further comprising a fetching process configured to fill a patch at a given position relative to the multidimensional array with data read from the memories in parallel.

8. The system of claim 7 wherein the fetching process is configured to rotate the data read from the memories into a consistent order.

9. One or more computer-readable storage media or logic having executable instructions, which when executed perform steps, comprising:

(a) loading multidimensional data into a plurality of memories;

(b) determining addresses in each of the memories based upon a position of a window;

(c) filling a data window with a single parallel read of the plurality of memories;

(d) outputting data corresponding to the window data;

(e) returning to step (b) when the position of the window changes for at least a plurality of different window positions.

10. The one or more computer-readable storage media or logic of claim 9 having further executable instructions comprising rotating the data in the data window.

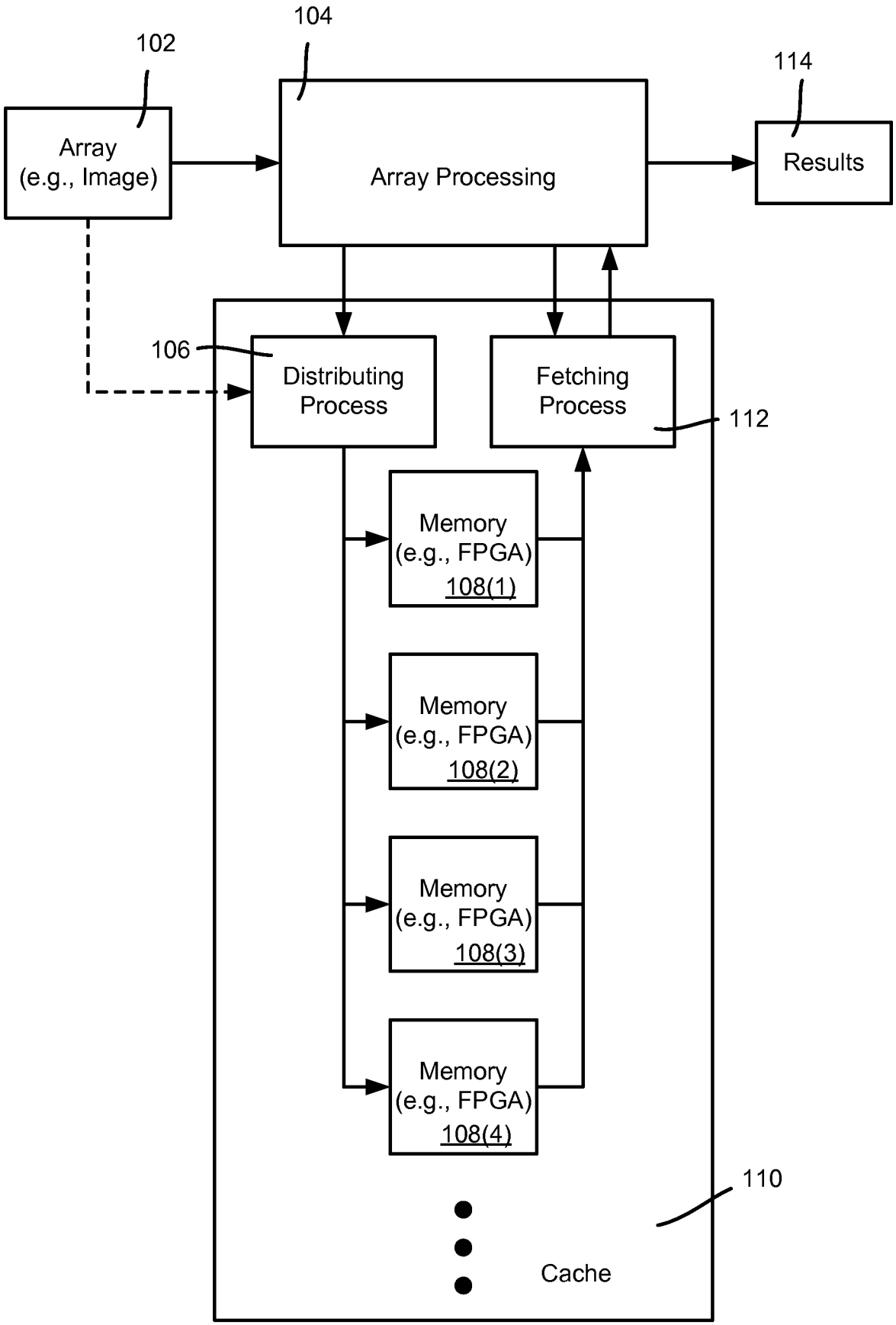
102
104
114
Array
(e.g., Image)
Array Processing
Results
106
Distributing
Process
Fetching
Process
112
Memory
(e.g., FPGA)
108(1)
Memory
(e.g., FPGA)
108(2)
Memory
(e.g., FPGA)
108(3)
Memory
(e.g., FPGA)
108(4)
110
Cache

FIG. 1

A 0, 0 MemW	B 1, 0 MemX	C 2, 0 MemW	D 3, 0 MemX
E 0, 1 MemY	F 1,1 MemZ	G 2, 1 MemY	H 3, 1 MemZ
I 0, 2 MemW	J 1, 2 MemX	K 2, 2 MemW	L 3, 2 MemX
M 0, 3 MemY	N 1, 3 MemZ	O 2, 3 MemY	P 3, 3 MemZ

FIG. 2A

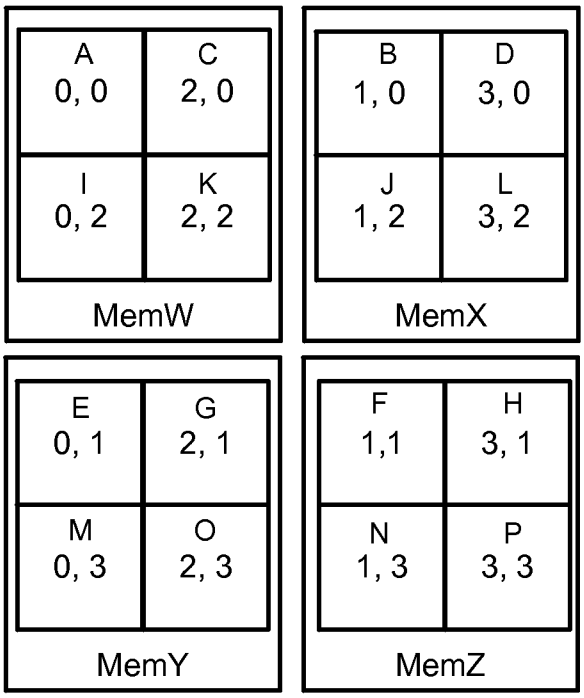

FIG. 2B

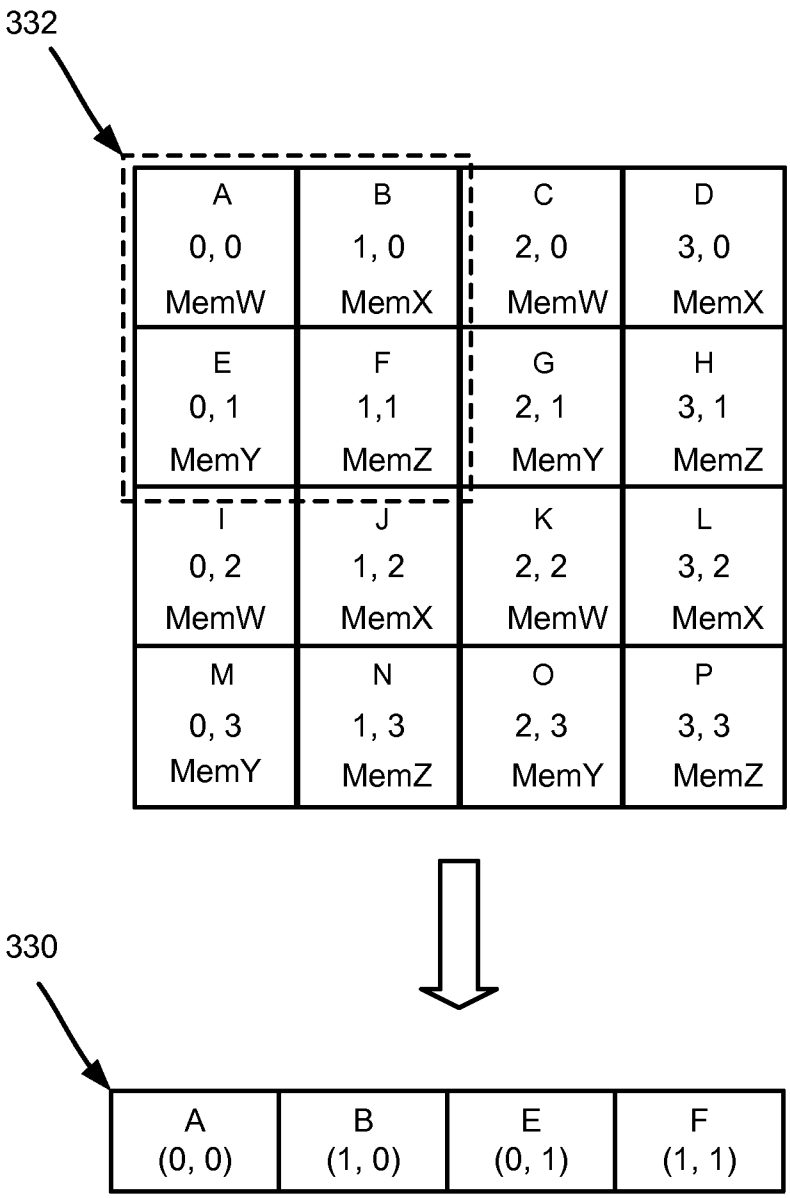
332
A
0, 0
MemW
B
1, 0
MemX
C
2, 0
MemW
D
3, 0
MemX
E
0, 1
MemY
F
1,1
MemZ
G
2, 1
MemY
H
3, 1
MemZ
I
0, 2
MemW
J
1, 2
MemX
K
2, 2
MemW
L
3, 2
MemX
M
0, 3
MemY
N
1, 3
MemZ
O
2, 3
MemY
P
3, 3
MemZ
330
A
(0, 0)
B
(1, 0)
E
(0, 1)
F
(1, 1)

FIG. 3

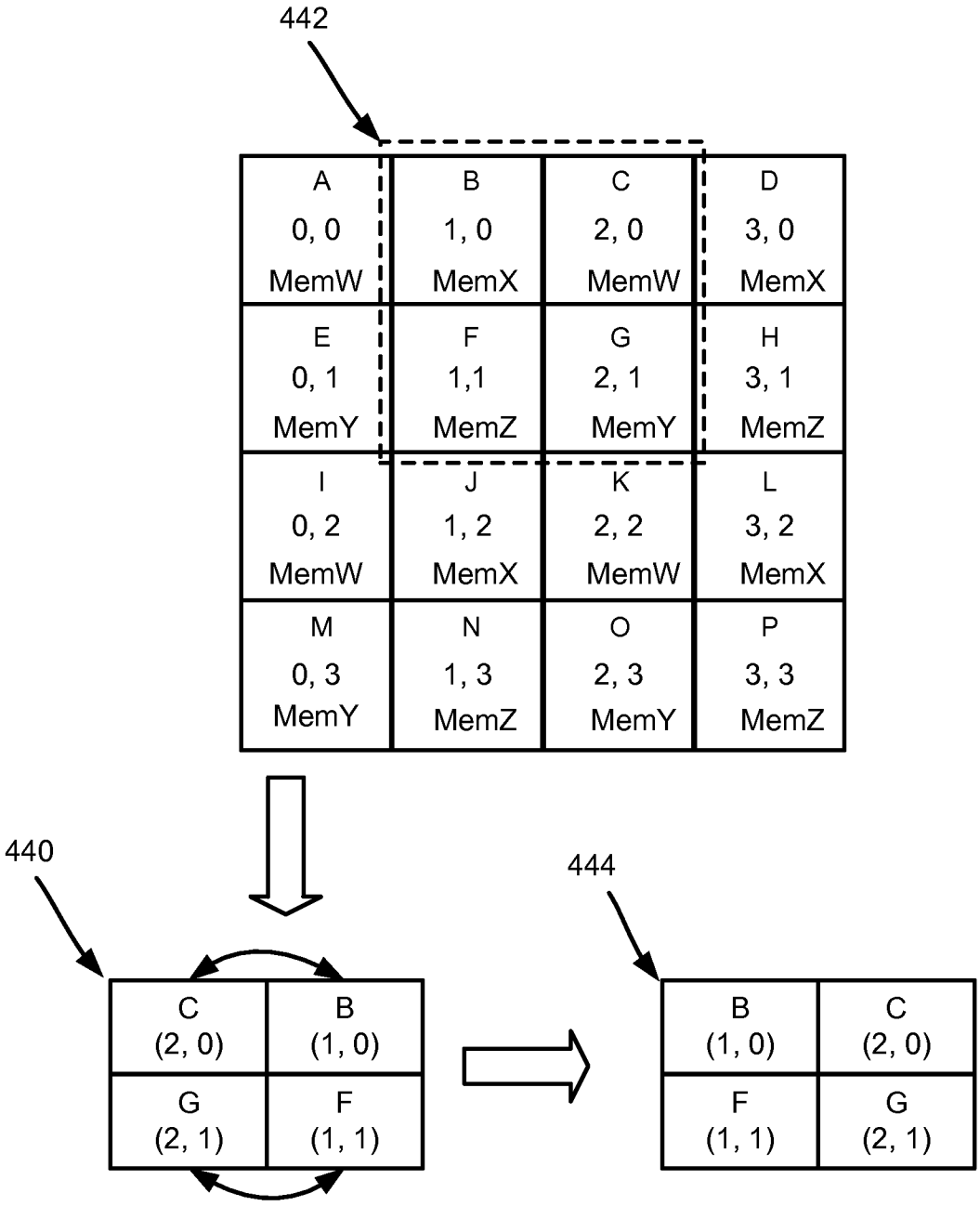
442
A
0, 0
MemW
B
1, 0
MemX
C
2, 0
MemW
D
3, 0
MemX
E
0, 1
MemY
F
1,1
MemZ
G
2, 1
MemY
H
3, 1
MemZ
I
0, 2
MemW
J
1, 2
MemX
K
2, 2
MemW
L
3, 2
MemX
M
0, 3
MemY
N
1, 3
MemZ
O
2, 3
MemY
P
3, 3
MemZ
440
C
(2, 0)
B
(1, 0)
G
(2, 1)
F
(1, 1)
444
B
(1, 0)
C
(2, 0)
F
(1, 1)
G
(2, 1)

FIG. 4

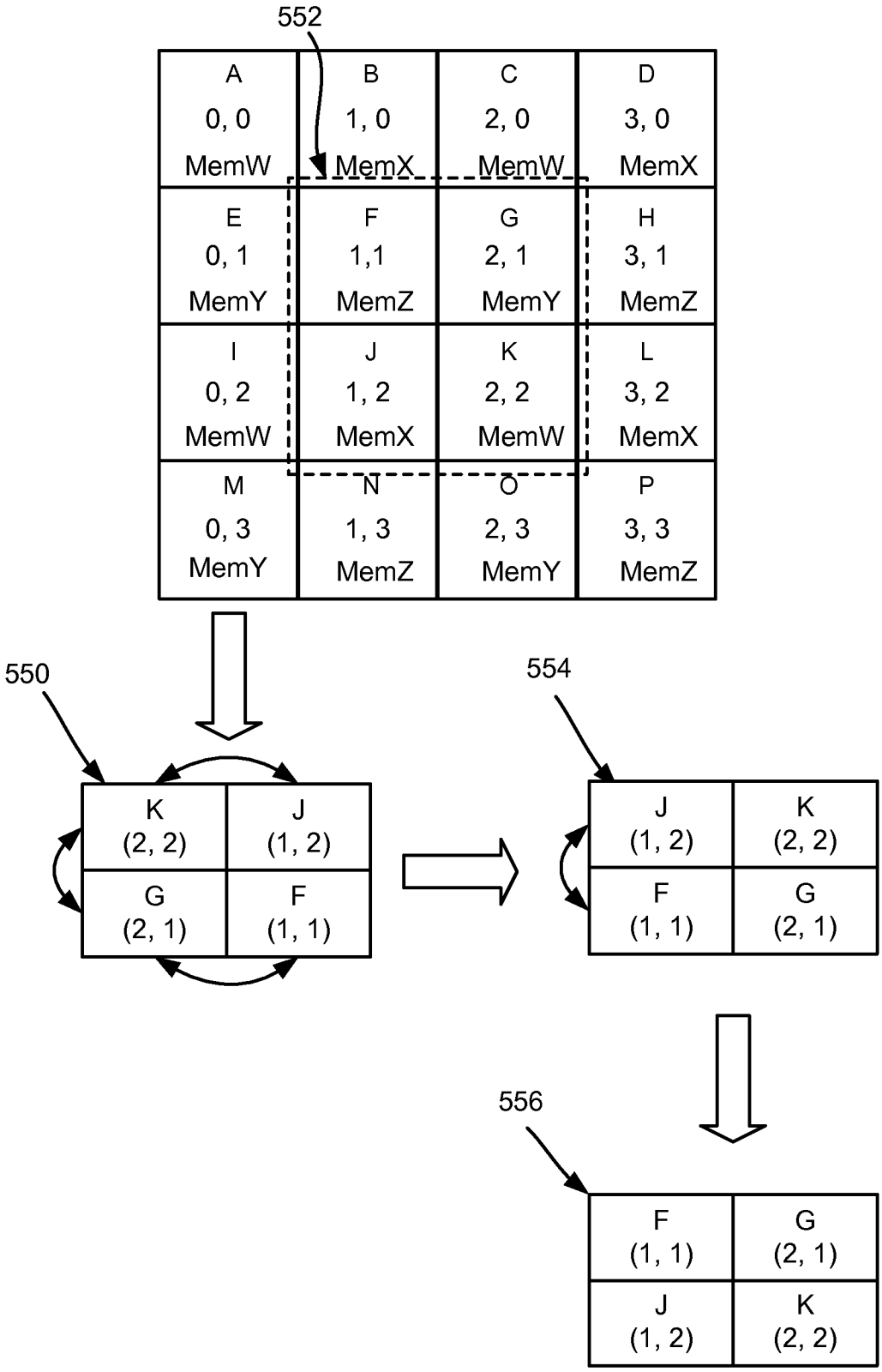
552
A
0, 0
MemW
B
1, 0
MemX
C
2, 0
MemW
D
3, 0
MemX
E
0, 1
MemY
F
1,1
MemZ
G
2, 1
MemY
H
3, 1
MemZ
I
0, 2
MemW
J
1, 2
MemX
K
2, 2
MemW
L
3, 2
MemX
M
0, 3
MemY
N
1, 3
MemZ
O
2, 3
MemY
P
3, 3
MemZ
550
K
(2, 2)
J
(1, 2)
G
(2, 1)
F
(1, 1)
554
J
(1, 2)
K
(2, 2)
F
(1, 1)
G
(2, 1)
556
F
(1, 1)
G
(2, 1)
J
(1, 2)
K
(2, 2)

FIG. 5

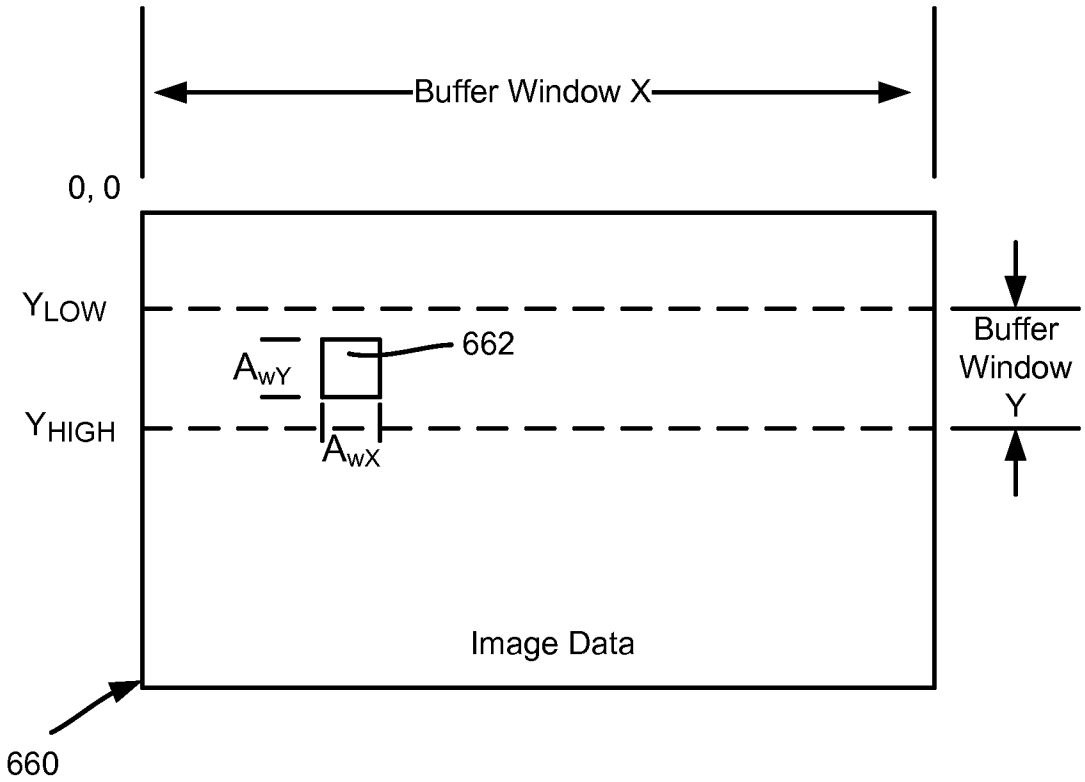
Buffer Window X
0, 0
YLOW
YHIGH
AwY
AwX
662
Buffer Window Y
Image Data
660

FIG. 6

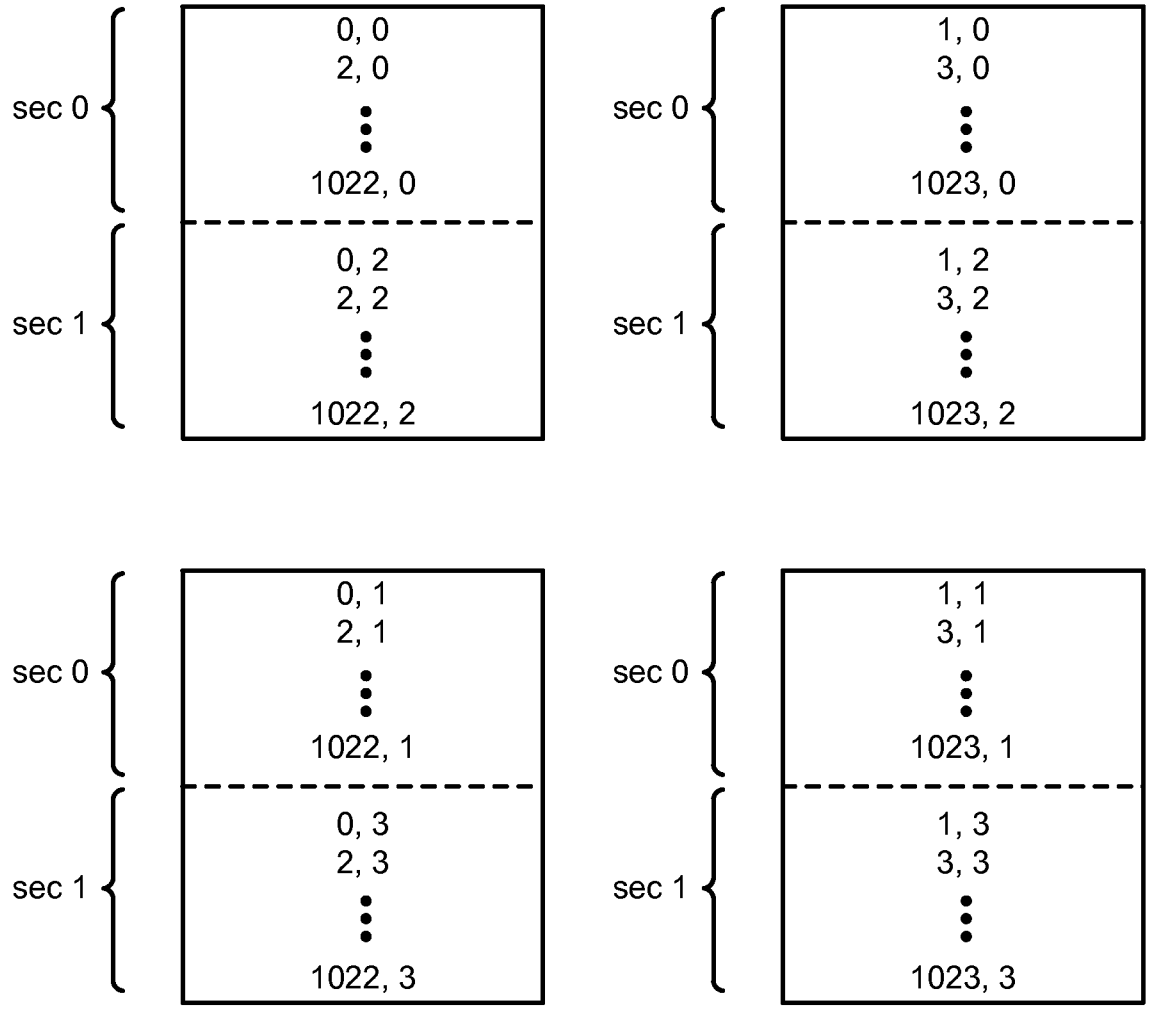
sec 0
0, 0
2, 0
1022, 0
sec 1
0, 2
2, 2
1022, 2
sec 0
1, 0
3, 0
1023, 0
sec 1
1, 2
3, 2
1023, 2
sec 0
0, 1
2, 1
1022, 1
sec 1
0, 3
2, 3
1022, 3
sec 0
1, 1
3, 1
1023, 1
sec 1
1, 3
3, 3
1023, 3

FIG. 7

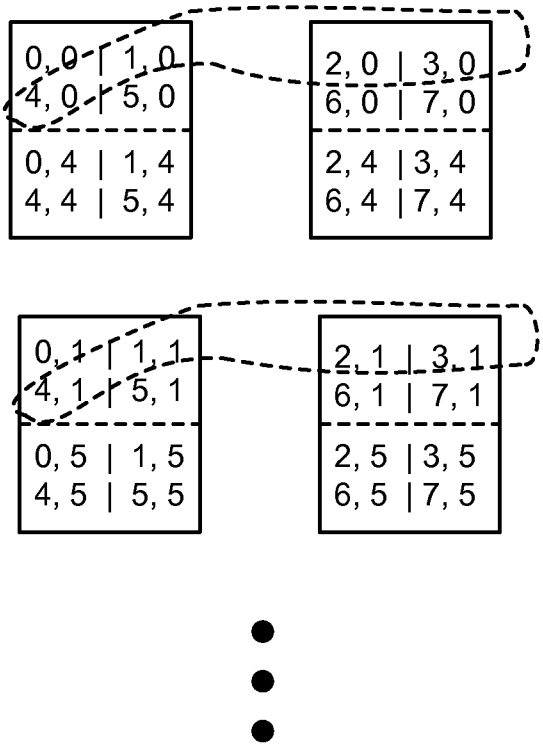
0, 0 | 1, 0
4, 0 | 5, 0
0, 4 | 1, 4
4, 4 | 5, 4
2, 0 | 3, 0
6, 0 | 7, 0
2, 4 | 3, 4
6, 4 | 7, 4
0, 1 | 1, 1
4, 1 | 5, 1
0, 5 | 1, 5
4, 5 | 5, 5
2, 1 | 3, 1
6, 1 | 7, 1
2, 5 | 3, 5
6, 5 | 7, 5

FIG. 8

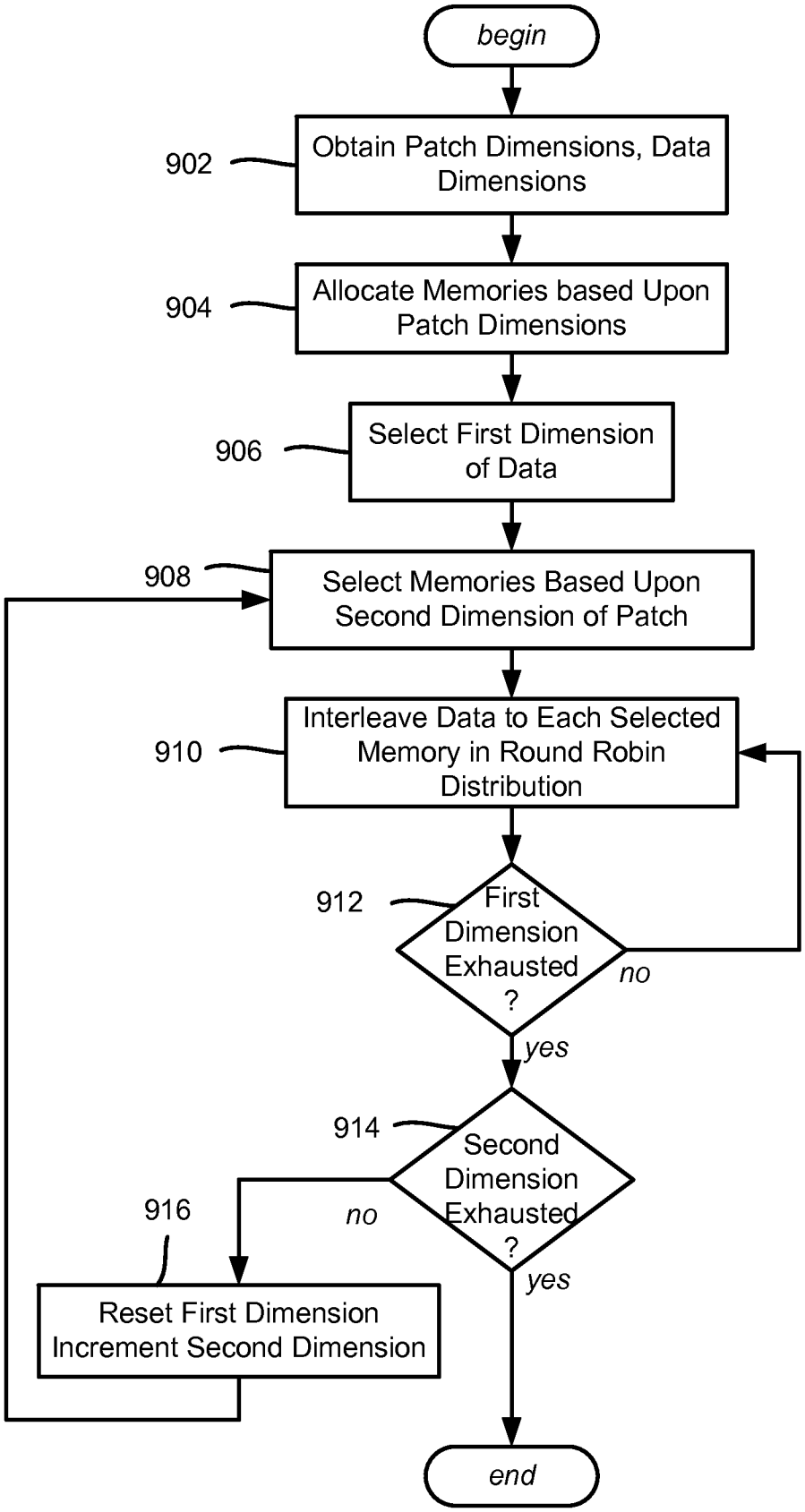
begin
902
Obtain Patch Dimensions, Data Dimensions
904
Allocate Memories based Upon Patch Dimensions
906
Select First Dimension of Data
908
Select Memories Based Upon Second Dimension of Patch
910
Interleave Data to Each Selected Memory in Round Robin Distribution
912
First Dimension Exhausted ?
no
yes
914
Second Dimension Exhausted ?
no
yes
916
Reset First Dimension Increment Second Dimension
end

FIG. 9

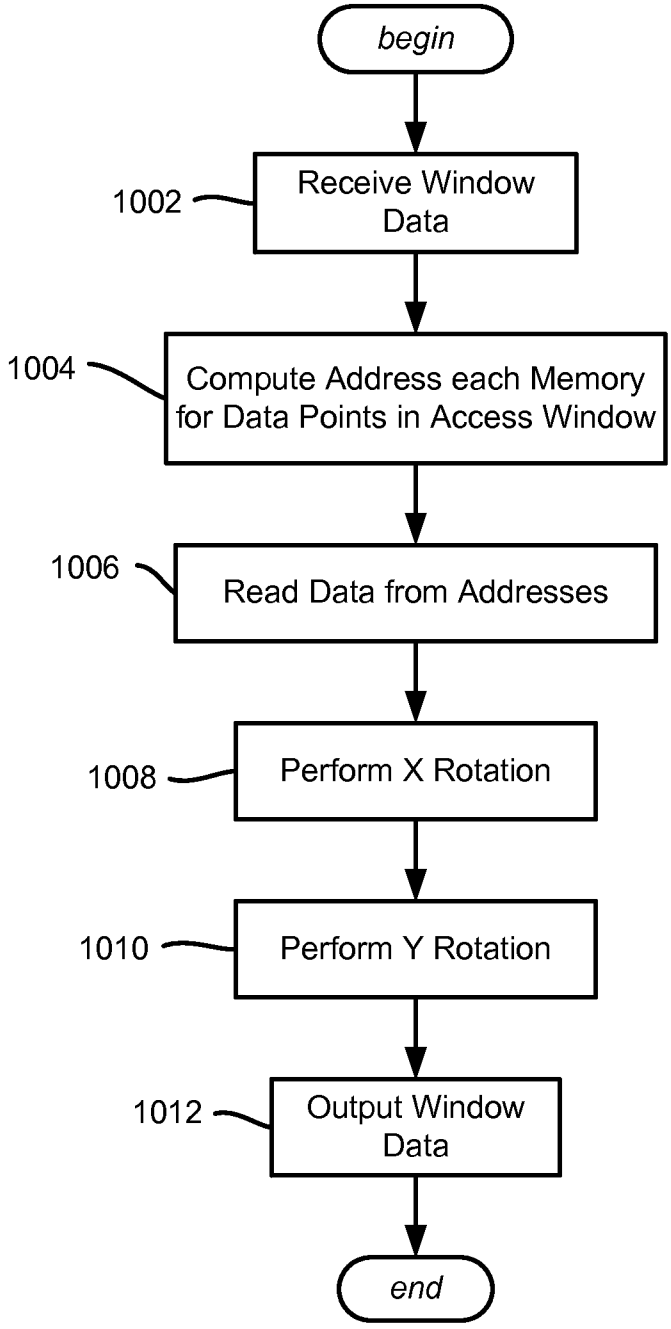
begin
1002
Receive Window Data
1004
Compute Address each Memory for Data Points in Access Window
1006
Read Data from Addresses
1008
Perform X Rotation
1010
Perform Y Rotation
1012
Output Window Data
end

FIG. 10

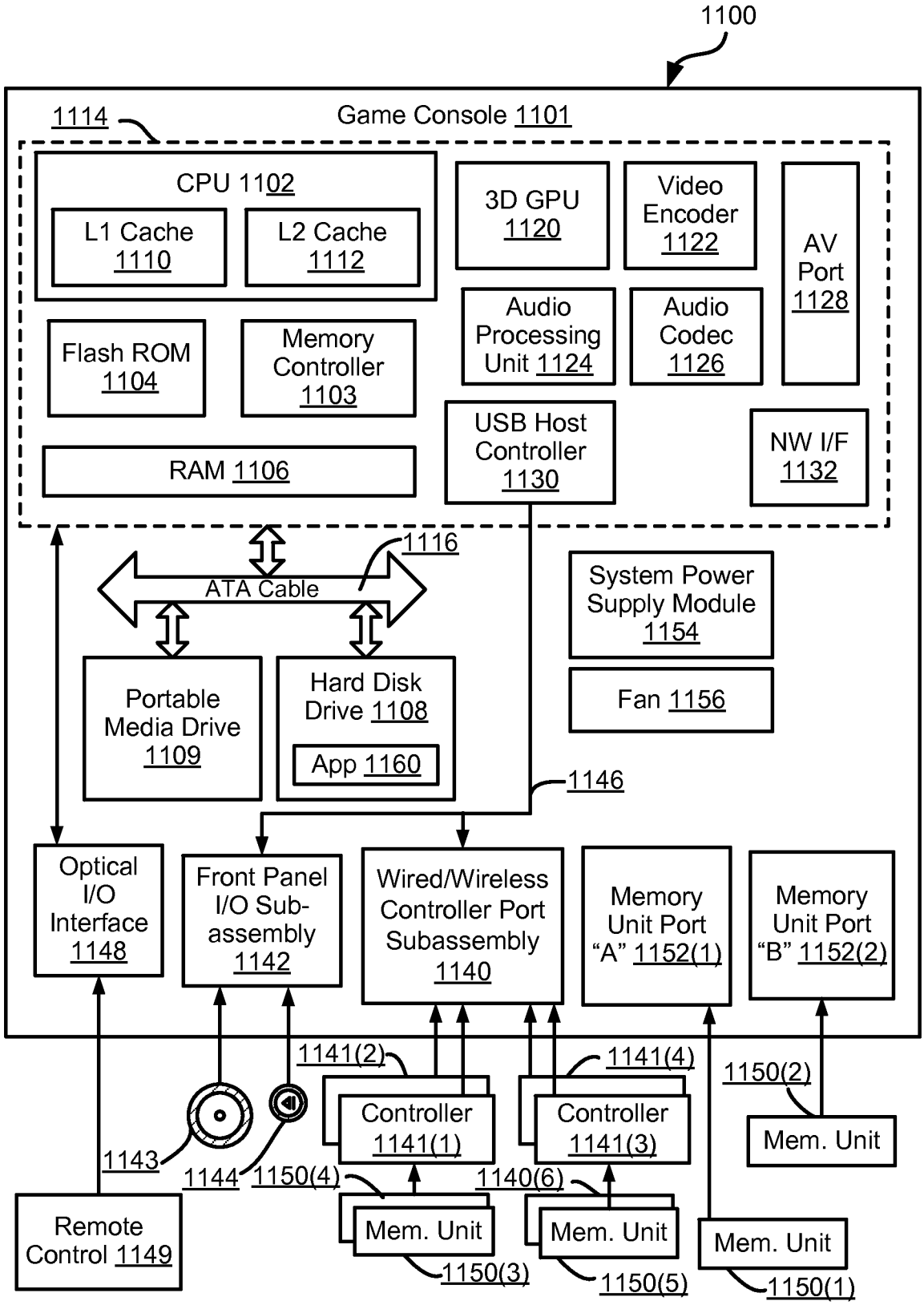
1100
1114
Game Console 1101
CPU 1102
L1 Cache 1110
L2 Cache 1112
3D GPU 1120
Video Encoder 1122
AV Port 1128
Audio Processing Unit 1124
Audio Codec 1126
Flash ROM 1104
Memory Controller 1103
USB Host Controller 1130
NW I/F 1132
RAM 1106
1116
ATA Cable
System Power Supply Module 1154
Fan 1156
Portable Media Drive 1109
Hard Disk Drive 1108
App 1160
1146
Optical I/O Interface 1148
Front Panel I/O Sub-assembly 1142
Wired/Wireless Controller Port Subassembly 1140
Memory Unit Port "A" 1152(1)
Memory Unit Port "B" 1152(2)
1141(2)
1141(4)
1150(2)
Controller 1141(1)
Controller 1141(3)
Mem. Unit
1143
1144
1150(4)
1140(6)
Remote Control 1149
Mem. Unit
Mem. Unit
Mem. Unit
1150(3)
1150(5)
1150(1)

FIG. 11

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/033915

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06T1/60
ADD.

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)
G06T

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	YAMAGUCHI K ET AL: "Interleaved pixel lookup for embedded computer vision", COMPUTER VISION AND PATTERN RECOGNITION, CVPR WORKSHOPS 2008. IEEE, PISCATAWAY, NJ, USA, 23 June 2008 (2008-06-23), pages 1-8, XP031285708, the whole document -----	1-10
X	WO 2007/132399 A1 (KONINKL PHILIPS ELECTRONICS NV [NL]; ALBA PINTO CARLOS A [NL]; SETHURA) 22 November 2007 (2007-11-22) page 1, line 9 - page 3, line 28 page 8, line 1 - page 10, line 2 -----	1,2,5,6, 9
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
3 July 2014	16/07/2014

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Pierfederici, A

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/033915

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 085 210 A1 (IBM [US]; IBM UK [GB]) 10 August 1983 (1983-08-10) page 2, paragraph 4 - page 3, paragraph 1 -----	1,2,5,6, 9
A	I. KUON ET AL.: "FPGA Architecture: Survey and Challenges", FOUNDATIONS AND TRENDS IN ELECTRONIC DESIGN AUTOMATION, vol. 2, no. 2, 2007, pages 135-253, XP002726220, page 167, paragraph 4 - page 168, paragraph 2 -----	1-10
A	CHIHOUB A ET AL: "A BAND PROCESSING IMAGING LIBRARY FOR A TRICORE-BASED DIGITAL STILL CAMERA", REAL-TIME IMAGING, ACADEMIC PRESS LIMITED, GB, vol. 7, no. 4, 1 August 2001 (2001-08-01), pages 327-337, XP001124987, page 330, right-hand column, paragraph 1 - paragraph 2 -----	1-10
A	WO 2006/016303 A2 (PHILIPS INTELLECTUAL PROPERTY [DE]; KONINKL PHILIPS ELECTRONICS NV [NL) 16 February 2006 (2006-02-16) page 2, line 18 - page 3, line 6 -----	1-10

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2014/033915

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2007132399 A1	22-11-2007	CN 101443809 A	27-05-2009
		EP 2024928 A1	18-02-2009
		JP 5248482 B2	31-07-2013
		JP 2009536392 A	08-10-2009
		KR 20090012320 A	03-02-2009
		US 2009135192 A1	28-05-2009
		WO 2007132399 A1	22-11-2007
EP 0085210 A1	10-08-1983	EP 0085210 A1	10-08-1983
		JP S642993 B2	19-01-1989
		JP S58132855 A	08-08-1983
WO 2006016303 A2	16-02-2006	CN 101002226 A	18-07-2007
		EP 1779321 A2	02-05-2007
		JP 2008510213 A	03-04-2008
		US 2008094406 A1	24-04-2008
		WO 2006016303 A2	16-02-2006