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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 October 2008 (30.10.2008)

PCT

(10) International Publication Number
WO 2008/131308 A1

(51) International Patent Classification:
G06F 13/36 (2006.01) *G06F 13/362* (2006.01)

(21) International Application Number:
PCT/US2008/060942

(22) International Filing Date: 18 April 2008 (18.04.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/737,605 19 April 2007 (19.04.2007) US

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

(72) Inventors: XU, Ning-yi; One Microsoft Way, Redmond, Washington 98052-6399 (US). HSU, Feng-Hsiung; One Microsoft Way, Redmond, Washington 98052-6399 (US). CAI, Xiong-Fei; 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, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, 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*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

(54) Title: FIELD-PROGRAMMABLE GATE ARRAY BASED ACCELERATOR SYSTEM

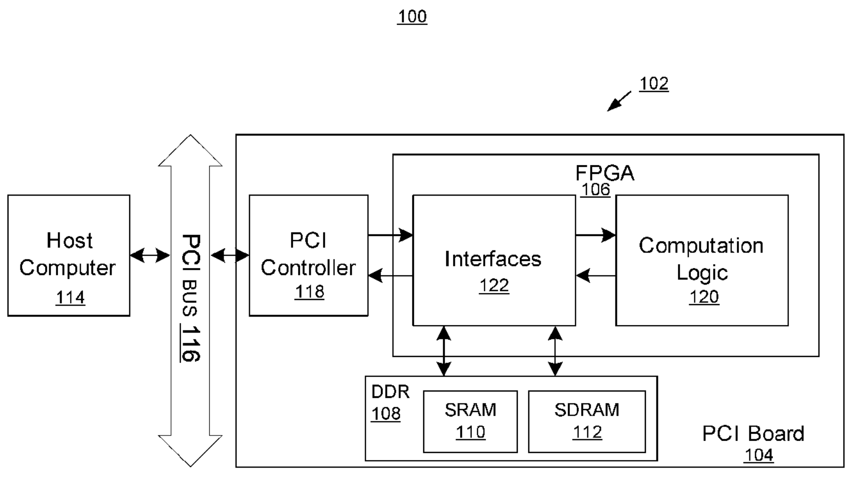

FIG. 1

(57) Abstract: Accelerator systems and methods are disclosed that utilize FPGA technology to achieve better parallelism and flexibility. The accelerator system may be used to implement a relevance-ranking algorithm, such as RankBoost, for a training process. The algorithm and related data structures may be organized to enable streaming data access and, thus, increase the training speed. The data may be compressed to enable the system and method to be operable with larger data sets. At least a portion of the approximated RankBoost algorithm may be implemented as a single instruction multiple data streams (SIMD) architecture with multiple processing engines (PEs) in the FPGA. Thus, large data sets can be loaded on memories associated with an FPGA to increase the speed of the relevance ranking algorithm.

WO 2008/131308 A1

FIELD-PROGRAMMABLE GATE ARRAY BASED ACCELERATOR SYSTEM

BACKGROUND

[0001] Web search based ad services and search engines have become important tools for providing information to users. One factor in attracting users and advertisers is providing relevant information and ads for a given search query. Search relevance may be determined by a ranking function that ranks resultant documents according to their similarities to the input query.

[0002] Information retrieval (IR) researchers have studied search relevance for various search engines and tools. Representative methods include Boolean, vector space, probabilistic, and language models. Earlier search engines and tools were mainly based on such IR algorithms. These search engines and tools incorporate in varying degrees the concept of the ranking function. Many factors may affect the ranking function for search relevance. These factors may include page content, title, anchor, URL, spam, and page freshness. It is extremely difficult to manually tune ranking function parameters to accommodate these factors for large-scale data sets, such as those that are common in many applications including World Wide Web ("Web") applications and speech and image processing. For these large data sets, machine based learning algorithms have been applied to learn complex ranking functions from large-scale data sets.

[0003] Early algorithms for ranking function learning include Polynomial-based regression, Genetic Programming, RankSVM and classification-based SVM. However, these algorithms were only evaluated on a small-scale dataset due to the high computational cost. In fact, these traditional machine-learning algorithms operate slowly when searching large-scale data sets. Users often wait many hours, days, or even weeks to get results from these data sets. This slow computation time may be due, in part, to a typical personal computer (PC) being unable to exploit full parallelism in machine-learning algorithms efficiently.

[0004] Instruction level parallelism techniques somewhat improve the processing time. More particularly, distributed implementations with process level parallelism are faster than many of the PC central processing units (CPUs), which execute instructions in sequential manner. However, distributed implementations occupy many machines. Additionally, for some algorithms, distributed computing yields poor speed improvement per processor added due to communication cost. A Graphics Processing Unit (GPU)-based accelerator could only accelerate a limited spectrum of machine learning algorithms due to its special hardware structure optimized for graphics applications. Thus, memory access bandwidth, communication cost, flexibility and granularity of parallelism remain bottlenecks for these solutions.

SUMMARY

[0005] An accelerator system and method is provided that, according to one exemplary implementation, utilizes FPGA technology to achieve better parallelism and flexibility. The FPGA-based accelerator uses a PCI controller to communicate with a host CPU. A memory hierarchy composed of embedded Random Access Memory (RAM) in the FPGA, Static Random Access Memory (SRAM) and Synchronous Dynamic Random Access Memory (SDRAM), allows the FPGA assisted accelerator to take advantage of memory locality in algorithms.

[0006] According to another exemplary implementation, an FPGA-based accelerator system is combined with a relevance-ranking algorithm, such as the algorithm known as RankBoost, to increase the speed of a training process. Using an approximated RankBoost algorithm reduces the computation and storage scale from $O(N^2)$ to $O(N)$. This algorithm could be mapped to the accelerator system to increase the speed of the pure software implementation by approximately 170 times. Several techniques assist in achieving the acceleration rate. The algorithm and related data structures associated with the FPGA-based accelerator may be organized to enable streaming data access and, thus, increase the training speed. The data may be compressed to enable the system and method to be operable with larger data sets. At least a portion of the approximated RankBoost algorithm may be implemented as a single instruction multiple data streams (SIMD) architecture with multiple processing engines (PEs) in the FPGA. Thus, large data sets, such as a training set can be loaded on memories associated with an FPGA to increase the speed of the relevance ranking algorithm.

[0007] By virtue of this system, a user can train a ranking model with much less time and cost, so they can attempt different learning parameters of the algorithm in the same time, or carry out a study that depends on numerous ranking models.

[0008] This Summary is provided to introduce a selection of 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 to limit the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 shows an exemplary architecture of an FPGA based accelerator system for machine learning.

[0010] FIG. 2 shows an exemplary deployment of the accelerator.

[0011] FIG. 3 shows an exemplary system architecture for an accelerator operable to perform relevance-ranking.

[0012] FIG. 4 shows an exemplary working flow of the accelerator system.

[0013] FIG. 5 shows an exemplary architecture of a processing engine (PE) for an accelerator operable to perform relevance-ranking.

[0014] FIG. 6 shows data sequences for the processing engine shown in FIG. 5.

[0015] FIG. 7 shows an exemplary data format and processing sequence for a First In First Out (FIFO) buffer.

DETAILED DESCRIPTION

[0016] Overview

An FPGA-based accelerator system for machine learning as described and claimed herein accelerates selected algorithms by providing better processing parallelism and memory access. The accelerator system may include an acceleration device, which may include a substrate, such as a Peripheral Component Interconnect (PCI) card, with a Field-Programmable Gate Array (FPGA) and memories acting as caches, e.g., SRAM, SDRAM, and so forth, connected to a computing device. One or more algorithms may be implemented on one or more of the FPGAs with direct parallel architecture and/or pipelined architecture to exploit both application parallelism and direct functional logic implementation. The PCI could also be replaced by other computer buses, including but not limited to PCI-X, PCI-Express, HyperTransport, Universal Serial Bus (USB) and Front-Side Bus (FSB).

[0017] A training data set or other data may be loaded onto one or more memories on the accelerator board, or onto embedded memories in the FPGA, to increase memory access bandwidth and data locality. The training data set may comprise information collected from Web searches to assess relevancy, and other characteristics. The system may include or be associated with one or more PCs or other computing devices, each computing device having one or more accelerator cards.

[0018] Exemplary System

[0019] *Accelerator System Architecture*

An exemplary system for use as an accelerator is shown in FIG. 1. The accelerator system 100 may include an acceleration device 102 comprising a Peripheral Component Interface (PCI) board 104 with a Field-Programmable Gate Array (FPGA) 106 and Double Data Rate (DDR) memory 108, e.g., SRAM 110, SDRAM 112, and so forth, connected to a computing device such as a host computer 114. The PCI board 104 may interface with a PCI bus 116 on or associated with the host computing device 114. The PCI board 104, and/or devices thereon, may communicate with the bus 116 thorough a PCI controller 118. The FPGA 106 may comprise computation logic 120 that communicates to the DDR memory devices 108 and/or the PCI controller 118 through one or more interfaces 122.

[0020] Training data or other data being accessed by the FPGA 106 may be loaded to DDR memory 108, including SRAM 110 or SDRAM 112, on the PCI board 104, or to embedded memories in the FPGA 106, in order to increase memory access bandwidth and data locality. Software loaded on the computer 114 may be capable of programming or re-programming the FPGA 106 at any time during processing.

[0021] As shown in FIG. 2, an acceleration system 200 may be composed of one or more computing devices 210, similar to computer 114, with each computing device 210 having one or more PCI cards 204, similar to PCI board 104. The computing devices 210 may be connected through a network 206. Thus, multiple cards 204 on multiple computing devices 210 may process data in parallel and thereby handle larger scale algorithms.

[0022] FIG. 3 shows a block diagram of a system 300 that is operable to implement relevance-ranking software 302 on an FPGA 304 residing on a substrate, such as a PCI card 305. The relevance ranking software 302 may have, or be associated with, a driver 306 having a register read/write (R/W) module 308 and/or a direct memory access read/write (DMA R/W) module 310 for operation of the software 302 with the CPU 312 and memory store 314 through a PCI 316 and/or Northbridge (NB) 318. The PCI card 305 may have a PCI 9054 Chip 320 or other 32-bit PCI bus mastering interface chip in order to facilitate communication between the FPGA 304 and the PCI 316.

[0023] The FPGA 304 may include a PCI local interface 322 for interfacing with the PCI 9054 chip 320. The PCI local interface 322 may also connect to the processing engine (PE) units, e.g., PE0, PE1, and PEn. The PE units implement the computation logic. The FPGA 304 may also have a DDR interface 324 for interfacing with DDR memory 326. The FPGA 304 may additionally have a control unit 328 for controlling the processing units PE0, PE1, PW, and PEn by sending a signal to the PE units. The FPGA 304 may also have a memory management unit (MMU) 330 for aligning or managing data for faster processing. The processing engines of the FPGA 304 may provide an output to the PCI local interface 320 for further implementation or use.

[0024] FIG. 4 illustrates an exemplary workflow 400 for processing and utilizing the training data in a system such as shown in FIG. 3. The first column 402 represents actions taken by the application software. The second column 404 represents driver-side activity. The third column 406 describes actions performed on, or by, hardware, such as accelerator device 301. In the following, steps (a)-(r) are steps for training. At (a), application software will process the training data for hardware. The possible processing may include organizing data in the sequence of how the FPGA logic will access and utilize it. At (b), application software will call the write routine in the driver (at (c)) to write the data to memories on the accelerator. The write routine may be implemented with a direct memory access (DMA) method to achieve high bandwidth access to the accelerator. At (d), upon receiving the training data, the PCI controller on the accelerator will write the data to the FPGA. Then the memory management unit (MMU) in the FPGA will write the data to DDR memory (or other cache memories). At (e), when all the data has been saved to the memory, the MMU may set a register, or issue an interrupt through PCI controller, indicating that the data transfer has been finished and the application software may proceed. At (g), the application software may check the status of the data transfer through register read routine in the driver (at (f)), or wait for the interrupt. At (h), application software configures the hardware to begin the training process by calling register write routine in the driver (at (i)). At (j), the control unit begins to control other hardware blocks to train using the training data. In the training process, at (k) application software may write (at (l)) some intermediate data to the accelerator hardware. At (m), the MMU in the accelerator sends this data to participate the training. At (p), this intermediate data may be generated from intermediate results of the training process (at (n)). At (q), the software may check the status of the training (at (r))) to determine if the training process needs to be continued for another round. The software continues to monitor the training process to decide when the training rounds should be stopped.

[0025] *Data organization*

The accelerator system supports hierarchical memory organization and access methods using SDRAM, SRAM and RAM/registers within the FPGA.

[0026] According to one example, training data that will be iteratively used may be loaded onto SDRAM onboard an accelerator device, such as accelerator device 301. The training data loaded in the SDRAM may be organized according to its access order in logic associated with the FPGA by a software tool so that the FPGA can fetch data in a so-called, and well-known, “burst” mode, thus enabling high bandwidth access to the data set.

[0027] Randomly used large-scale data structures could be loaded to SRAM onboard the accelerator device, such as accelerator device 301, and associated with an FPGA, such as FPGA 304. According to this implementation, the SRAM may be used as a large low latency cache.

[0028] Temporary data structures, such as intermediate variables, parameters, and so forth, and results, e.g., the learned model, could be stored in distributed memory or registers inside the FPGA, which would act as high bandwidth, low latency cache. The data could be utilized without needing to access memory off of the FPGA, which would enhance the access speed of the cache.

[0029] *Stream Data Processing Architecture*

A streaming data access architecture and method may be incorporated with the accelerator system and/or the data organization structure, such as described above, to enable fast access to data in the host memory hierarchy and to decrease the amount of hardware/software communication for selected algorithms. Software, which may be provided by or on a host CPU, may configure a PCI bridge chip on the accelerator board to fetch data from a host memory hierarchy. The host memory hierarchy may provide a variety of memories including hard disks. The data will contain necessary information (generated and inserted by software), with which FPGA logic can perform computing functions over the data stream without interaction with software or drivers. Furthermore, the data will be organized in the sequence of how FPGA logic is accessing and utilizing it, such that input data is consumed at the time FPGA logic receives it. The FPGA may buffer the result for a delayed read from the software, which reduces the time and processing cost of hardware/software communication.

[0030] *Data Compression/Decompression*

A bit-map based data compression/decompression method for the architecture may be implemented to increase memory capacity and bandwidth available in the accelerator system. Training data may be compressed by conventional compression software and stored in the memories associated with the acceleration device. The FPGA may then read and decompress the data before performing computations. Implementing compression and decompression techniques with the FPGA may increase the virtual bandwidth from a DDR to a PE by 2-4 times the virtual bandwidth for uncompressed data.

[0031] Relevance-Ranking Algorithm

A machine learning, relevance ranking, algorithm may be implemented using the accelerator system. Generally, when ranking objects, the goal is to find a ranking function to order the given set of objects. Such an object is denoted as an instance x in a domain (or instance space) X. As a form of feedback, information about which instance should be ranked above (or below) one another is provided for every pair of instances. This feedback is denoted as function Φ: $X \times X \rightarrow R$, where $\Phi\ (x_0, x_1)>0$ means x_1 should be ranked above x_0, and$\Phi\ (x_0, x_1)<0$ means x_0

should be ranked above x_1. A learner then attempts to find a ranking function H: $X \rightarrow R$, which is as consistent as possible to the given Φ, by asserting x_1 is preferred over x_0 if $H(x_1) > H(x_0)$.

[0032] A relevance-ranking algorithm may be used to learn the ranking function H by combining a given collection of ranking functions. The relevance-ranking algorithm may be pair-based or document-based. The psuedocode for one such relevance ranking algorithm, is shown below:

```
Initialize: Distribution $D$ over $X$ x $X$
Do for $t = 1,...,T$:
    (1) Train WeakLearn using distribution $D_t$.
    (2) WeakLearn returns a weak hypothesis $h_t$,
    (3) Choose $\alpha_t \in R$
    (4) Update weights: for each pair $(d_0, d_1)$:
        $D_{t+1}(d_0,d_1) = \frac{D_t(d_0,d_1)\exp(-\alpha_t(h_t(d_0)-h_t(d_1)))}{Z_t}$
        where $Z_t$ is the normalization factor:
        $Z_t = \sum_{x_0,x_1} D_t(d_0,d_1)\exp(-\alpha_t(h_t(d_0)-h_t(d_1)))$.
Output: the final hypothesis: $H(x) = \sum_{t=1}^{T} \alpha_t h_t$
```

[0033] The relevance-ranking algorithm is utilized in an iterative manner. In each round, a procedure named "WeakLearn" is called to select the best "weak ranker" from a large set of candidate weak rankers. The weak ranker has the form

h_t: $X \rightarrow R$ and $h_t(x_1) > h_t(x_0)$ means that instance x_1 is ranked higher than x_0 in round t. A distribution D_t over $X \times X$ is maintained in the training process. Weight $D_t(x_0, x_1)$ will be decreased if h_t ranks x_0 and x_1 correctly ($h_t(x_1) > h_t(x_0)$), and increased otherwise. Thus, D_t will tend to concentrate on the pairs that are hard to rank. The final strong ranker H is a weighted sum of the selected weak rankers in each round.

[0034] The WeakLearn algorithm may be implemented to find the weak ranker with a maximum $r(f, \theta)$, by generating a temporary variable $\pi(d)$ for each document. The WeakLearn algorithm may be defined as follows:

Given: Distribution $D(d_0, d_1)$ over all pairs

Initialize: (1) For each document $d(q)$:

$$\text{Compute } \pi(d(q)) = \sum\nolimits_{d'(q)} (D(d'(q), d(q)) - D(d(q), d'(q)))$$

(2) For every feature f_k and every threshold θ^k_s:

$$\text{Compute } r(f_k, \theta_s^k) = \sum_{d(q): f_k(d(q)) > \theta_s^k} \pi(d(q))$$

(3) Find the maximum $|r^*(f_{k^*}, \theta_{s^*}^{k^*})|$

(4) Compute: $\alpha = \frac{1}{2}\ln(\frac{1+r^*}{1-r^*})$

Output: weak ranking $(f_{k^*}, \theta_{s^*}^{k^*})$ and α.

[0035] To extend the relevance-ranking algorithm to Web relevance ranking, training pairs may be generated and weak rankers may be defined. To generate the training pairs, the instance space for a search engine may be partitioned according to queries issued by users. For each query q, the returned documents may be rated a relevance score, from 1 (means 'poor match') to 5 (means 'excellent match') using a manual or automated process. Unlabeled documents may be given a relevance score of 0. Based on the rating scores (ground truth), the training pairs for the relevance-ranking algorithm may be generated from the returned documents for each query.

[0036] So-called “weak rankers” may be defined as a transformation of a document feature, which is a one-dimensional real value number. Document features can be classified into query dependent features, such as query term frequencies in a document and term proximity, and query independent features, such as PageRank, and so forth. Thus, the same document may be represented by different feature vectors for different queries based upon its query-dependent features.

[0037] In keeping with the previous algorithm example, a document may be designated as $d(q)$, a pair as $\{d_1(q), d_2(q)\}$, and d^i_j means a document for query q_i. The k_{th} feature for document is denoted as $f_k(d^i_j)$. With these notations, an alternative relevance-ranking algorithm may be implemented as follows.

Initialize: initial distribution D over $X{\times}X$

Given: N_q queries $\{q_i \mid i=1..., N_q\}$.

N_i documents $\{ d^i_j \mid j=1,\dots, N_i \}$ for each query q_i, where $\sum_{i=1}^{N_q} N_i = N_{doc}$.

N_f features $\{f_k(d^i_j) \mid j=1,\dots, N_f\}$ for each document d^i_j

N^k_θ candidate thresholds $\{\theta^k_s \mid s=1,\dots, N^k_\theta\}$ for each f_k.

N_{pair} pairs (d^i_{j1}, d^i_{j2}) generated by ground truth rating $\{R(q_i, d^i_j)\}$ or $\{R^i_j\}$.

Initialize: initial distribution D(d^i_{j1}, d^i_{j2}) over $X \times X$

Do for $t = 1,...,T$:

(1) Train **WeakLearn** using distribution D_t.

(2) **WeakLearn** returns a weak hypothesis h_t, weight α_t

(3) Update weights: for each pair (d_0, d_1):

$$D_{t+1}(d_0, d_1) = \frac{D_t(d_0, d_1)\exp(-\alpha_t(h_t(d_0) - h_t(d_1)))}{Z_t}$$

where Z_t is the normalization factor:

$$Z_t = \sum_{x_0, x_1} D_t(d_0, d_1)\exp(-\alpha_t(h_t(d_0) - h_t(d_1))).$$

Output: the final hypothesis: $H(x) = \sum_{t=1}^{T} \alpha_t h_t$

[0038] For the relevance-ranking algorithms described by example above, WeakLearn may be defined as a routine that uses the N_f document features to form its weak rankers, attempting to find the one with the smallest pair-wise disagreement relative to distribution D over N_{pair} document pairs. The weak ranker may be defined by the following relationship:

$$h(d) = \begin{cases} 1 & if\ f_i(d) > \theta \\ 0 & if\ f_i(d) \le \theta\ or\ f_i(d)\ is\ undefined \end{cases}$$

[0039] To find the best $h(d)$, the weak learner checks all of the possible combinations of feature f_i and threshold θ. The WeakLearn algorithm may be implemented to ascertain a maximum $r(f, \theta)$ by generating a temporary variable $\pi(d)$ for each document. Intuitively, π contains information regarding labels and pair weights, and the weak ranker only needs to access π in a document-wise manner for each feature and each threshold, that is $O(N_{doc}N_f\, N_\theta)$, in a straightforward implementation. Based on this, an alternative weak learner may be utilized using an integral histogram to further reduce the computational complexity to $O(N_{doc}N_f)$. Because of this relatively low computational complexity, the algorithm may be implemented in both software and hardware, e.g., an accelerator system utilizing an FPGA, as described above.

[0040] According to the implementation, r may be calculated in $O(N_{doc}N_f)$ time in each round using an integral histogram in $O(N_{doc}N_f)$ time. First, feature values $\{f_k(d)\}$ in a dimension of the whole feature vector $(f_1,...,f_{N_f})$ may be classified into N_{bin} bins. The boundaries of these bins are:

$$\theta_s^k = \frac{f_{\max}^k - f_{\min}^k}{N_{bin}} \cdot s + f_{\min}^k, s = 0,1,..., N_{bin},$$

where $f_{\max}^k$ and $f_{\min}^k$ are maximum and minimum values of all f_k in the training data set. Then each document d can be mapped to one of the bins according to the value of $f_k(d)$:

$$Bin_k(d) = floor(\frac{f_k(d) - f_{\min}^k}{f_{\max}^k - f_{\min}^k} \cdot N_{bin} - 1)$$

The histogram of $\pi(d)$ over feature f_k is then built using:

$$Hist_k(i) = \sum_{d:Bin_k(d)=i} \pi(d), i = 0,...,(N_{bin} - 1)$$

Then, an integral histogram can be determined by adding elements in the histogram from the right ($i=N_{bin}-1$) to the left ($i=0$). That is,

$$Integral_k(i) = \sum_{a>i} Hist_k(a), i = 0,...,(N_{bin} - 1)$$

[0041] Exemplary Implementation of Relevance-Ranking Algorithm

A relevance-ranking algorithm, such as described above, may be implemented using an FPGA based accelerator system, also described above. The main computing data structure is a histogram, mapped to the architecture of single instruction multiple data streams (SIMD) with distributed memories. The SIMD architecture is capable of separately building several integral histograms with multiple PE units at the same time, as described above.

[0042] Software provided on or to the host computer will send the quantized feature values to a DDR memory through the PCI bus, PCI controller and FPGA. As described above, the data may be organized to enable streaming memory access, which can make full use of DDR memory bandwidth. In each training round, the software will call WeakLearn to compute $\pi(d)$ for every document, and send $\pi(d)$ to a First In First Out (FIFO) queue in the FPGA. The control unit (CU) in the FPGA will direct the PE arrays to build histograms and integral histograms, and will then send the results $r(f,\theta)$ as output to the FIFO queue. The CU is implemented as a finite state machine (FSM), which halts or resumes the pipeline in PE units according to the status of each FIFO. When the CU indicates that the calculation of r is finished, the software will read back these r

values and select the maximum value. Then the software will update the distribution $D(d_0, d_1)$ over all pairs and begin the next round.

[0043] It is noted that the micro-architecture of the PE supports fully-pipelined operation, which enhances the performance of hardware, particularly with regard to machine learning algorithms, such as a relevance-ranking algorithm. Fig. 5 illustrates an exemplary micro-architecture of a processing engine 500, such as PE0, PE1, or PEn, previously described. This micro-architecture may be used in building the integral histogram for a relevance ranking algorithm, such as RankBoost. The dual port RAM 502 is used to store the histograms and integral histograms in the building process. The base address of the RAM indexes the feature and the offset address indexes the bin of histogram or integral histogram as defined in the $Hist_k(i)$and $Integral_k(i)$ equations described above. The shift registers 504 are used to delay the input feature $f(d)$. First, the histogram is built. Feature values $f_k(d)$ are input as the offset address to read out the corresponding intermediate values $Hist_k(i)$ of the bin i. Then the other input $\pi(d)$ will be added to $Hist_k(i)$, and the result will be saved to the same bin where $Hist_k(i)$ is read out.

[0044] An example data input 600 into 8 PE arrays with 16 features per PE is illustrated in FIG. 6. First, software aligns a given amount data in its original format. A software tool may be used to rearrange this data in the memory to generate binary data for storage in a memory block. The data is ordered according to the order that the FPGA will access the data. Moreover the input data may be organized to be aligned with the PE, as shown in FIG. 6, thereby mapping the data structure to the memories. This data organization enables a streaming memory access. The integral histogram can be implemented with this data structure based on the histogram stored in the dual port RAM. The values are read out, added and then stored back to the memory according to the $Integral_k(i)$ equation described above.. At last the final result $r(f, \theta)$ will be read out.

[0045] A streaming memory access organization can also be used for the FIFO buffer that will provide data from the DDR memory to the group of PE units. The width of the FIFO associated with the PE array may be, for example, 128 bits, which is equivalent to 16 bytes. The data in the FIFO can be arranged as shown in Fig. 7 to map the data to the PE units and further enable streaming memory access to data transferred from the DDR to the FPGA. Thus, running the relevance-ranking algorithm utilizing an FPGA and incorporating streaming memory access provides fast processing of large data sets.

[0046] Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

CLAIMS

1. A system comprising:

a Field Programmable Gate Array (FPGA) provided on a substrate;

a memory connected to the substrate and the FPGA;

an interface for connecting the FPGA to a computing device; and

a relevance-ranking algorithm associated with the logic in FPGA.

2. A system as recited in claim 1, wherein the substrate comprises a Peripheral Component Interface (PCI) board, PCI-X board, PCI-Express board, HyperTransport board, Universal Serial Bus (USB) board or Front-Side Bus (FSB) board.

3. A system as recited in claim 1, wherein the FPGA has at least one processing engine, and the processing engine is controlled by a control unit.

4. A system as recited in claim 3, wherein the memory comprises Double Data Rate (DDR) memory.

5. A system as recited in claim 1, wherein the relevance ranking algorithm incorporates a RankBoost algorithm.

6. A system as recited in claim 4, wherein the FPGA comprises a number of processing engine (PE) unit, and wherein data is arranged in a First In First Out (FIFO) buffer to map the data onto the PE unit.

7. A system as recited in claim 1, wherein the FPGA associated with a computing device is a first FPGA associated with a first computing device, the system further comprising a network connecting the first FPGA associated with the first computing device to a second FPGA associated with the second computing device.

8. A system as recited in claim 7, wherein the first and second computing devices are each associated with multiple FPGA devices

9. A method comprising:

mapping data in one or more data structures to one or more memories associated with a Field Programmable Gate Array (FPGA); and

enabling the performance of a relevance ranking-algorithm with respect to the data structures.

10. A method as recited in claim 9, wherein the relevance-ranking algorithm comprises at least a portion of a RankBoost algorithm.

11. A method as recited in claim 9, further comprising loading a training dataset into a memory associated with the FPGA.

12. A method as recited in claim 9, further comprising utilizing memory within the FPGA as a cache memory.

13. The method of claim 9, further comprising compressing data in the one or more data structures prior to mapping the one or more data structures in the one or more memories.

14. The method of claim 9, wherein the relevance-ranking algorithm is document based.

15. The method of claim 9, further comprising organizing the data according to input data structure of processing units in the FPGA and according to the order by which the FPGA will access the data.

16. A system comprising:

an FPGA logic device operable to perform a machine learning algorithm;

a PCI controller to communicate with a Central Processing Unit (CPU) of a host computing device, and a memory hierarchy composed of Static Random Access Memory (SRAM) and Synchronous Dynamic Random Access Memory (SDRAM) associated with the FPGA and embedded Random Access Memory (RAM) within the FPGA.

17. A system according to claim 16, wherein the machine learning algorithm comprises a document-based relevance-ranking algorithm.

18. A system according to claim 16, wherein:

training data that will be iteratively used is loaded onto the SDRAM and organized according to its usage pattern in logic associated with the FPGA;

randomly used large-scale data structures are loaded onto the SRAM to be used as a large low latency cache; and

temporary data structures are stored in the embedded RAM to act as high bandwidth, low latency cache.

19. A system according to claim 16, wherein multiple FPGA devices are provided for connection to the computer.

20. A system according to claim 19, wherein multiple computers are connected together through a network.

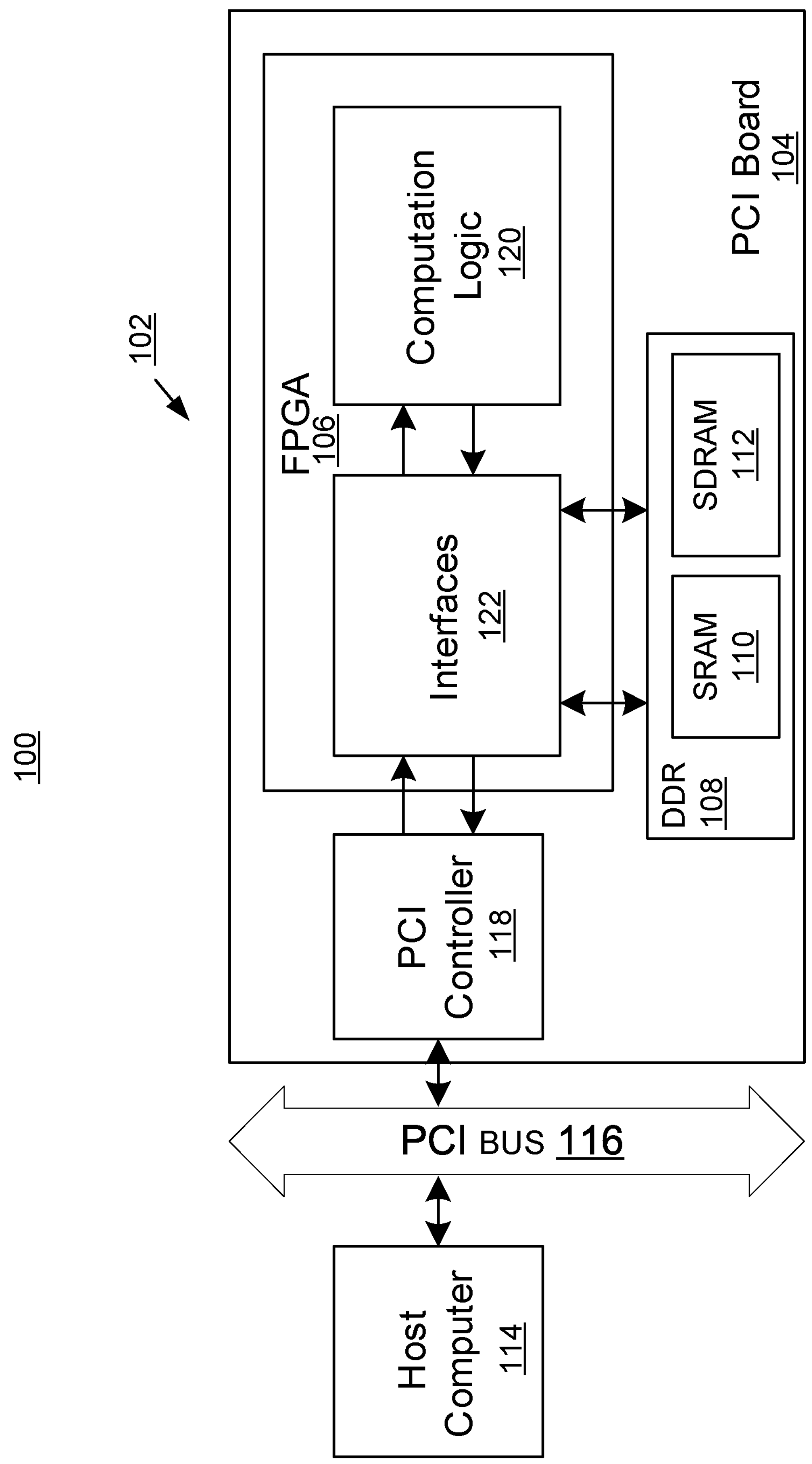
100
102
PCI Board 104
FPGA 106
Computation Logic 120
Interfaces 122
DDR 108
SDRAM 112
SRAM 110
PCI Controller 118
PCI BUS 116
Host Computer 114

FIG. 1

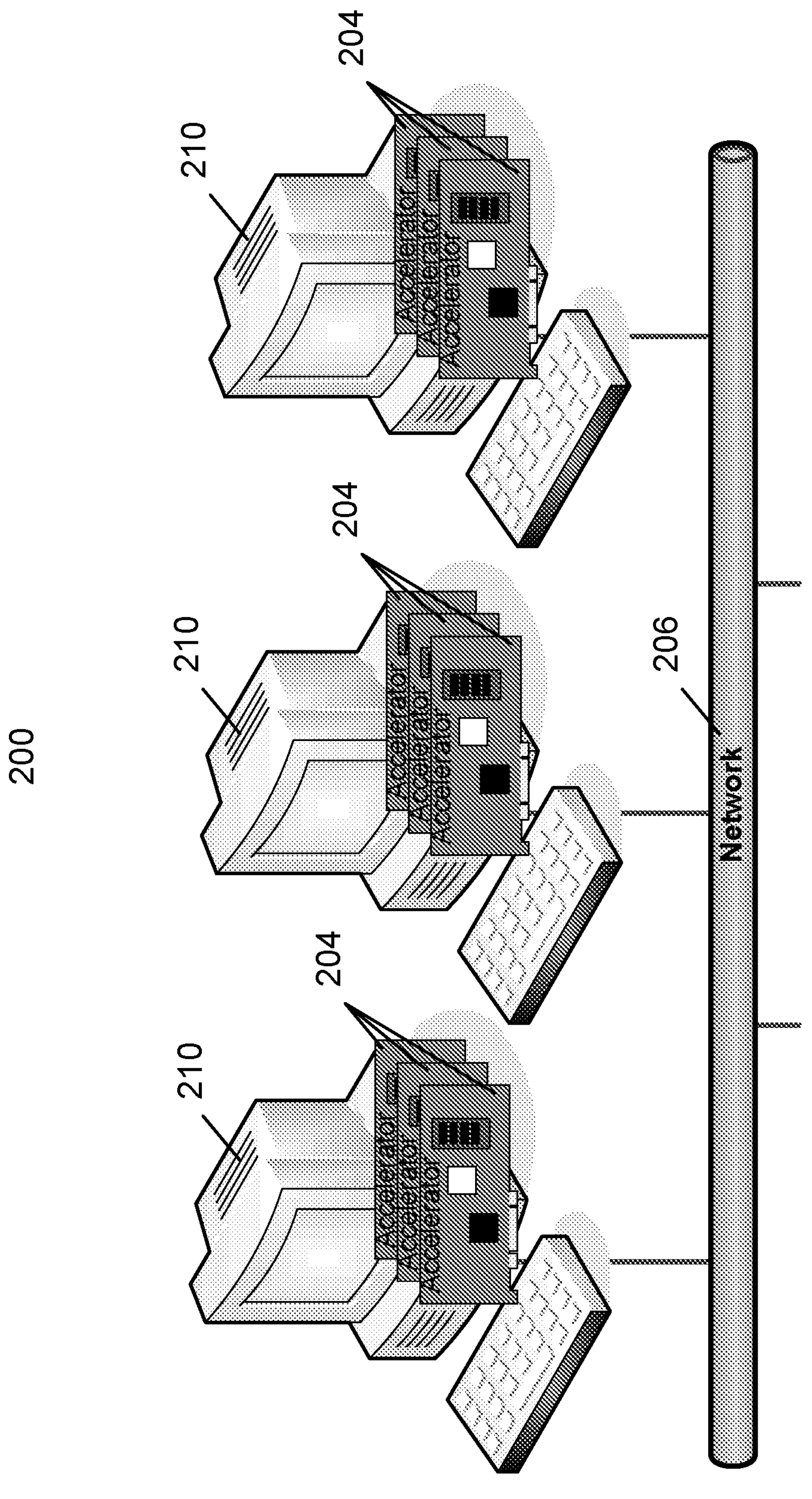
200
210
204
Accelerator
Accelerator
Accelerator
210
204
Accelerator
Accelerator
Accelerator
206
Network
210
204
Accelerator
Accelerator
Accelerator

FIG. 2

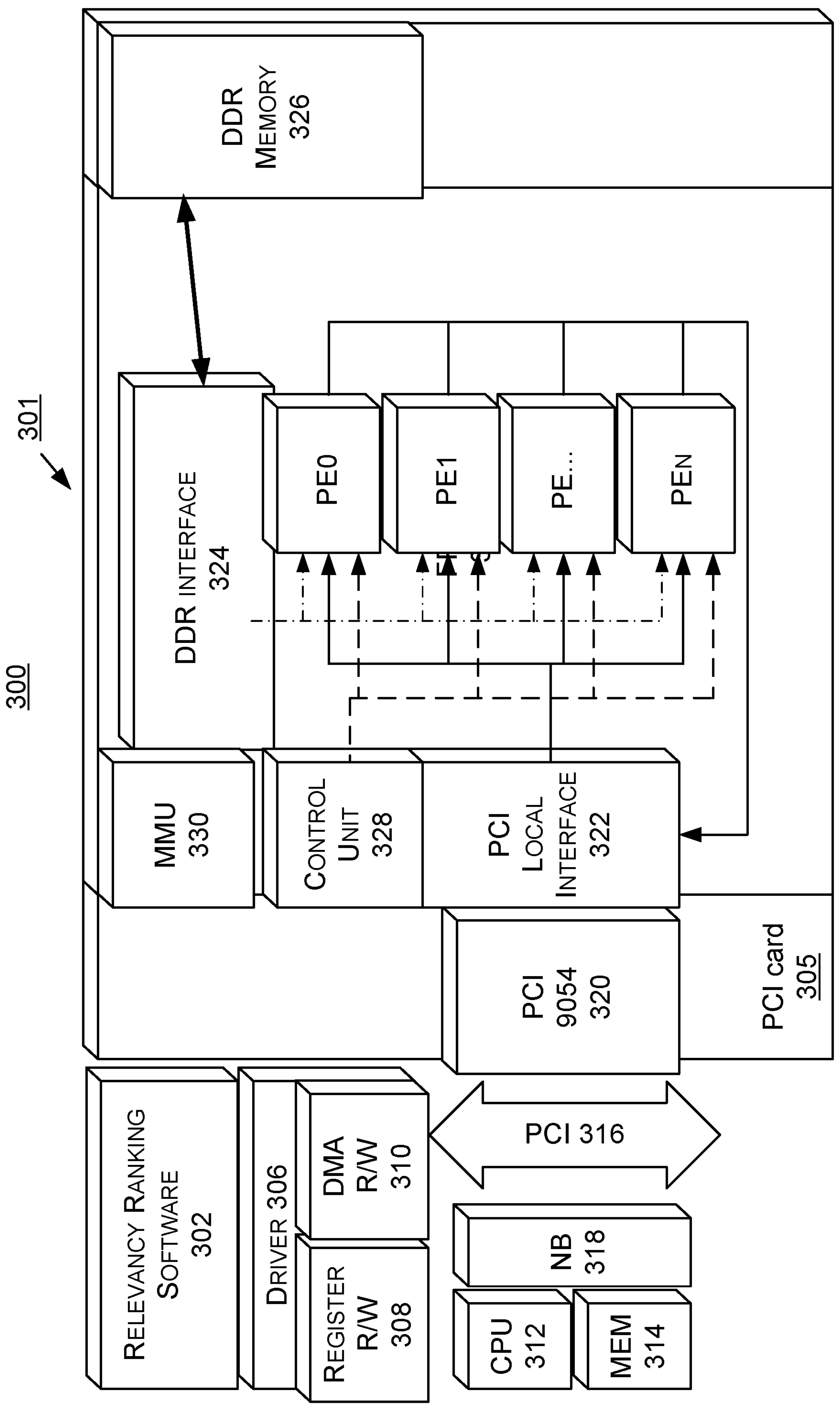
300
301
DDR MEMORY 326
DDR INTERFACE 324
PE0
PE1
PE...
PEN
MMU 330
CONTROL UNIT 328
PCI LOCAL INTERFACE 322
PCI 9054 320
PCI card 305
RELEVANCY RANKING SOFTWARE 302
DRIVER 306
DMA R/W 310
REGISTER R/W 308
PCI 316
NB 318
CPU 312
MEM 314

FIG. 3

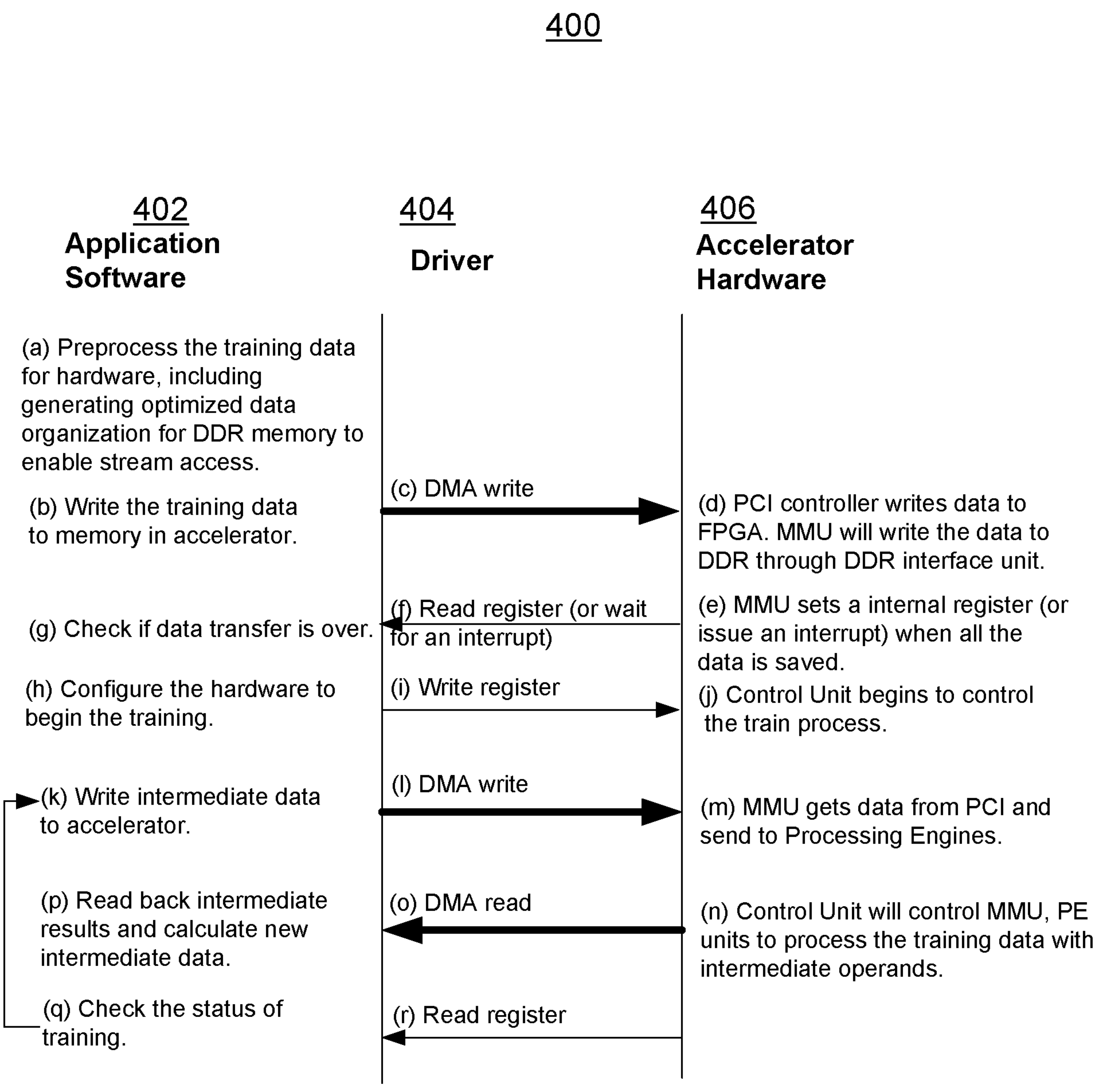
400
402
Application Software
404
Driver
406
Accelerator Hardware
(a) Preprocess the training data for hardware, including generating optimized data organization for DDR memory to enable stream access.
(b) Write the training data to memory in accelerator.
(c) DMA write
(d) PCI controller writes data to FPGA. MMU will write the data to DDR through DDR interface unit.
(e) MMU sets a internal register (or issue an interrupt) when all the data is saved.
(f) Read register (or wait for an interrupt)
(g) Check if data transfer is over.
(h) Configure the hardware to begin the training.
(i) Write register
(j) Control Unit begins to control the train process.
(k) Write intermediate data to accelerator.
(l) DMA write
(m) MMU gets data from PCI and send to Processing Engines.
(n) Control Unit will control MMU, PE units to process the training data with intermediate operands.
(o) DMA read
(p) Read back intermediate results and calculate new intermediate data.
(q) Check the status of training.
(r) Read register

FIG. 4

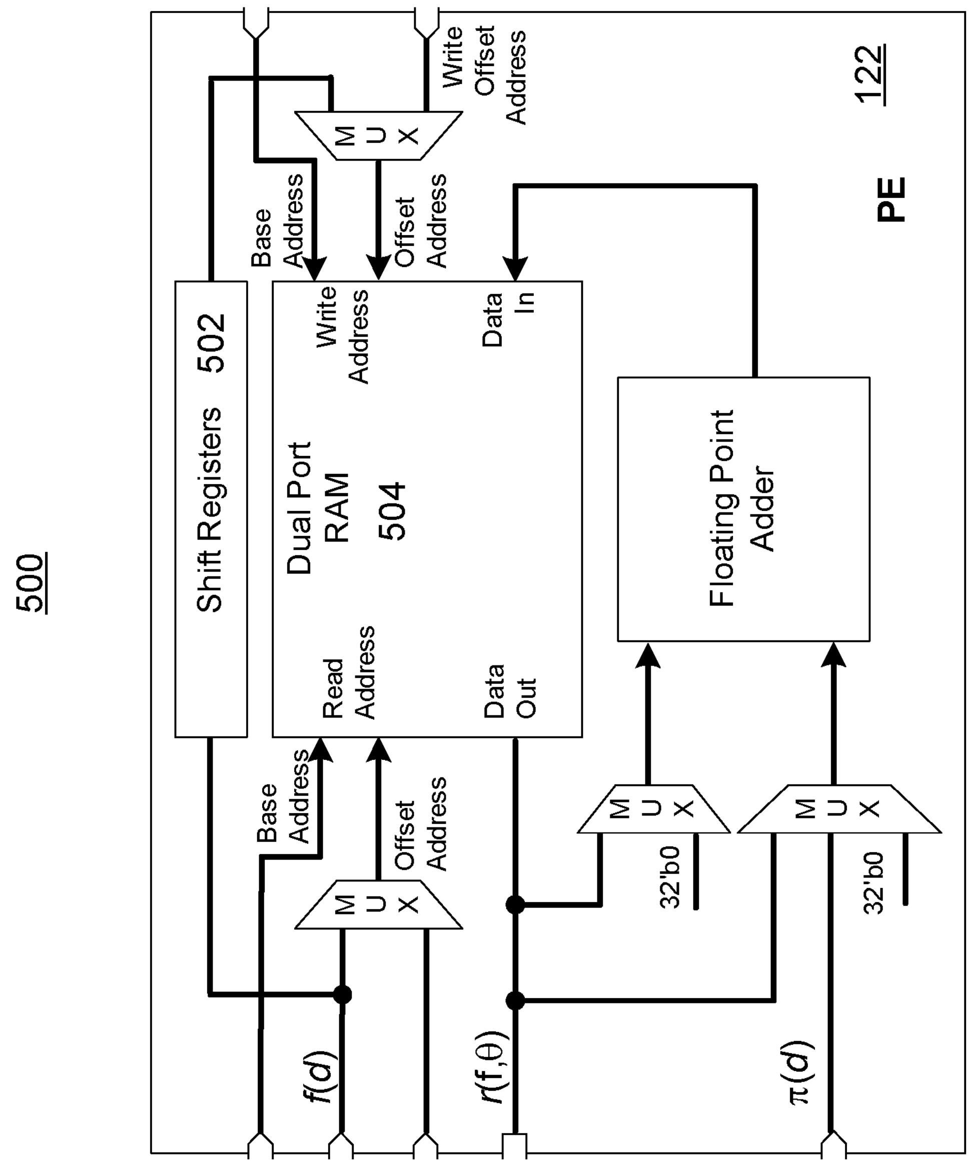
500
Shift Registers 502
Dual Port RAM 504
Write Address
Read Address
Data In
Data Out
Base Address
Offset Address
Write Offset Address
MUX
Floating Point Adder
32'b0
f(d)
r(f,θ)
π(d)
PE
122

FIG. 5

600

PE0	F(0,0)	F(8,0)	F(16,0)	...	F(120,0)	F(0,1)	F(8,1)	F(16,1)	...	F(120,1)	...	F(0,N-1)	F(8,N-1)	F(16,N-1)	...	F(120,N-1)
PE1	F(1,0)	F(9,0)	F(17,0)	...	F(121,0)	F(1,1)	F(9,1)	F(17,1)	...	F(121,1)	...	F(1,N-1)	F(9,N-1)	F(17,N-1)	...	F(121,N-1)
PE2~6	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
PE7	F(7,0)	F(15,0)	F(23,0)	...	F(127,0)	F(7,1)	F(15,1)	F(23,1)	...	F(127,1)	...	F(7,N-1)	F(15,N-1)	F(23,N-1)	...	F(127,N-1)

FIG. 6

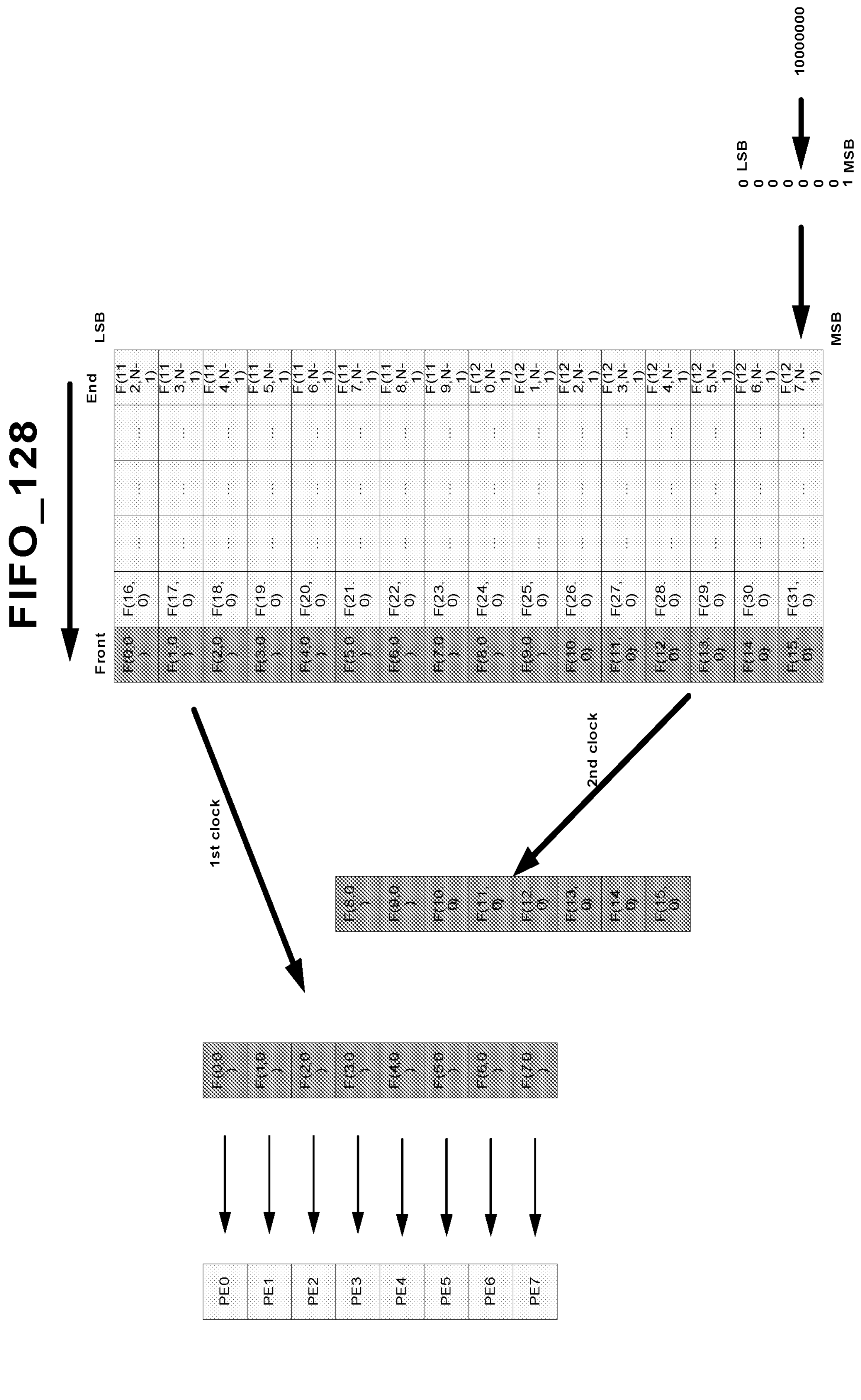
FIFO_128
Front
End
LSB
MSB
0 LSB
0
0
0
0
0
0
1 MSB
10000000
F(0,0)
F(1,0)
F(2,0)
F(3,0)
F(4,0)
F(5,0)
F(6,0)
F(7,0)
F(8,0)
F(9,0)
F(10,0)
F(11,0)
F(12,0)
F(13,0)
F(14,0)
F(15,0)
F(16,0)
F(17,0)
F(18,0)
F(19,0)
F(20,0)
F(21,0)
F(22,0)
F(23,0)
F(24,0)
F(25,0)
F(26,0)
F(27,0)
F(28,0)
F(29,0)
F(30,0)
F(31,0)
F(112,N-1)
F(113,N-1)
F(114,N-1)
F(115,N-1)
F(116,N-1)
F(117,N-1)
F(118,N-1)
F(119,N-1)
F(120,N-1)
F(121,N-1)
F(122,N-1)
F(123,N-1)
F(124,N-1)
F(125,N-1)
F(126,N-1)
F(127,N-1)
1st clock
2nd clock
PE0
PE1
PE2
PE3
PE4
PE5
PE6
PE7

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2008/060942

A. CLASSIFICATION OF SUBJECT MATTER

G06F 13/36(2006.01)i, G06F 13/362(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 8 : G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean Utility models and applications for Utility models since 1975
Japanese Utility models and applications for Utility models since 1975

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKIPASS(KIPO internal) "relevance", "ranking", "function", "search", "accelerator"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 7197497 B2(DAVID COSSOCK) 27 March 2007 See abstract; figures 1-6.	1-20
A	US 2005-0246328 A1(BENYU ZHANG et al.) 3 November 2005 See abstract; figures 1-5.	1-20
A	US 2004-0111388 A1(FREDRIC BOISCUVIER et al.) 10 June 2004 See abstract; figure 2.	1-20
A	US 2005-0234953 A1(BENYU ZHANG et al.) 10 June 2004 See abstract.	1-20

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

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

Date of the actual completion of the international search 26 AUGUST 2008 (26.08.2008)	Date of mailing of the international search report **26 AUGUST 2008 (26.08.2008)**
Name and mailing address of the ISA/KR Korean Intellectual Property Office Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea Facsimile No. 82-42-472-7140	Authorized officer HAN, Seon Kyoung Telephone No. 82-42-481-8523

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2008/060942

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 7197497 B2	27.03.2007	CN 1826597 A	30.08.2006
		EP 1623298 A2	08.02.2006
		JP 2006-524869 T2	02.11.2006
		KR 10-2006-0006945 A	20.01.2006
		US 2004-0215606 A1	28.10.2004
		WO 2004-097568 A2	11.11.2004
		WO 2004-097568 A3	05.01.2006
US 2005-0246328 A1	03.11.2005	AU 2005-201824 A1	17.11.2005
		CA 2505904 AA	30.10.2005
		CN 1758244 A	12.04.2006
		EP 1591923 A1	02.11.2005
		JP 2005-322244 A2	17.11.2005
		KR 10-2006-0047664 A	18.05.2006
		PA 05004681 A	08.03.2006
		RU 2005113189 A	10.11.2006
US 2004-0111388 A1	10.06.2004	AU 2003-283793 A1	30.06.2004
		EP 1570381 A1	07.09.2005
		US 2006-206466 AA	14.09.2006
		WO 2004-053734 B1	29.07.2004
US 2005-0234953 A1	10.06.2004	AU 2005-201684 A1	27.10.2005
		BR 200503051 A	06.12.2005
		CA 2504181 A1	15.10.2005
		CN 1691019 A	02.11.2005
		EP 1587010 A2	19.10.2005
		EP 1587010 A3	02.11.2006
		JP 2005-302041 A2	27.10.2005
		KR 10-2006-0045786 A	17.05.2006
		PA 05004098 A	19.10.2005
		RU 2005111001 A	20.10.2006
		US 7260568 B2	21.08.2007