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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
03 May 2018 (03.05.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/080624 A1

(51) International Patent Classification:
G06N 3/10 (2006.01)

(21) International Application Number:
PCT/US2017/047992

(22) International Filing Date:
22 August 2017 (22.08.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15/336,066 27 October 2016 (27.10.2016) US
15/465,774 22 March 2017 (22.03.2017) US

(71) Applicant: **GOOGLE LLC** [US/US]; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).

(72) Inventors: **WOO, Dong Hyuk**; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **NARAYANASWAMI, Ravi**; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).

(74) Agent: **HENRY, Joel** et al.; Fish & Richardson P.C., P.O. Box 1022, Minneapolis, Minnesota 55440-1022 (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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH,

(54) Title: EXPLOITING INPUT DATA SPARSITY IN NEURAL NETWORK COMPUTE UNITS

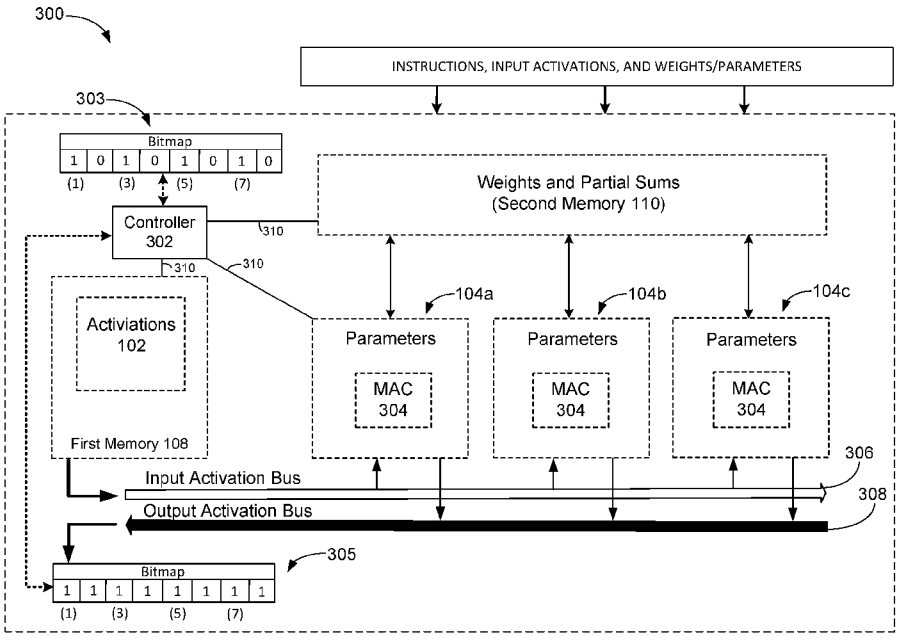

FIG. 3

(57) Abstract: A computer-implemented method includes receiving, by a computing device, input activations and determining, by a controller of the computing device, whether each of the input activations has either a zero value or a non-zero value. The method further includes storing, in a memory bank of the computing device, at least one of the input activations. Storing the at least one input activation includes generating an index comprising one or more memory address locations that have input activation values that are non-zero values. The method still further includes providing, by the controller and from the memory bank, at least one input activation onto a data bus that is accessible by one or more units of a computational array. The activations are provided, at least in part, from a memory address location associated with the index.

WO 2018/080624 A1

[Continued on next page]

GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, 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))*

EXPLOITING INPUT DATA SPARSITY IN NEURAL NETWORK COMPUTE UNITS

BACKGROUND

[0001] This specification relates to performing machine learning computations using a special purpose computational unit.

[0002] Neural networks are machine learning models that employ one or more layers of models to generate an output, e.g., a classification, for a received input. Some neural networks include one or more hidden layers in addition to an output layer. The output of each hidden layer is used as input to the next layer in the network, i.e., the next hidden layer or the output layer of the network. Each layer of the network generates an output from a received input in accordance with current values of a respective set of parameters.

[0003] Some neural networks include one or more convolutional neural network layers. Each convolutional neural network layer has an associated set of kernels. Each kernel includes values established by a neural network model created by a user. In some implementations, kernels identify particular image contours, shapes, or colors. Kernels can be represented as a matrix structure of weight inputs. Each convolutional layer can also process a set of activation inputs. The set of activation inputs can also be represented as a matrix structure.

SUMMARY

[0004] One way of computing convolution calculations requires numerous matrix multiplications in a large dimensional space. A processor or controller device of a compute unit can compute matrix multiplications through a brute force method. For example, although compute-intensive and time-intensive, the processor can repeatedly calculate individual sums and products for convolution calculations. The degree to which the processor parallelizes calculations is limited due to its architecture.

[0005] An innovative aspect of the subject matter described in this specification can be embodied in a computer-implemented method. The method includes receiving, by a computing device, a plurality of input activations, the input activations being provided, at least in part, from a source external to the computing device and determining, by a controller of the computing device, whether each of the plurality of input activations is one of a zero value or a non-zero value. The method further includes storing, in a memory bank of the computing device, at least one input activation, wherein storing the at least one of the input activations includes generating, by the controller, an index comprising one or more memory

address locations having input activation values that are non-zero values. The method still further includes providing, by the controller and from the memory bank, at least one input activation onto a data bus that is accessible by one or more units of a computational array, wherein the activations are provided, at least in part, from a memory address location associated with the index.

[0006] In some implementations, the index is created based on a bitmap comprising a plurality of bits and, wherein each bit of the bitmap indicates at least one of a non-zero input activation value or a zero input activation value. In some implementations, the method further includes, providing a first input activation that has a non-zero value to perform, by at least one unit, a computation using the non-zero value, and subsequently providing a second input activation that has a zero value, and preventing, in at least one unit, computation that would otherwise be performed using the zero value.

[0007] In some implementations, preventing occurs in response to the controller determining that the input activation is provided from a memory address location that is not associated with the index. In some implementations, the method further includes, detecting, by the controller, that the input activation is provided from a memory address location that is not associated with the index, and, in response to detecting, providing a control signal to at least one unit of the computational array to prevent a multiply operation associated with the zero input activation value.

[0008] In some implementations, the method further comprises, mapping, by the controller and to a first unit, a first portion of a tensor computation that uses a first input activation and mapping, to a second unit that differs from the first unit, a second portion of the tensor computation that also uses the first input activation. In some implementations, the method further comprises, sequentially providing a single input activation onto the data bus, the single input activation being accessed and selected from memory address locations that are associated with the index. In some implementations, providing further comprises, not providing input activations that have a zero value.

[0009] Another innovative aspect of the subject matter described in this specification can be embodied in one or more machine-readable storage devices storing instructions that are executable by one or more processing devices to perform operations comprising, receiving, by a computing device, a plurality of input activations, the input activations being provided, at least in part, from a source external to the computing device and determining, by a controller of the computing device, whether each of the plurality of input activations is one of a zero value or a non-zero value. The operations further comprise storing, in a memory bank

of the computing device, at least one of the input activations, wherein storing the at least one input activation includes generating, by the controller, an index comprising one or more memory address locations having input activation values that are non-zero values. The operations still further comprise providing, by the controller and from the memory bank, at least one input activation onto a data bus that is accessible by one or more units of a computational array, wherein the activations are provided, at least in part, from a memory address location associated with the index.

[0010] Another innovative aspect of the subject matter described in this specification can be embodied in an electronic system comprising a controller disposed in a computing device, the controller including one or more processing devices; and one or more machine-readable storage devices for storing instructions that are executable by the one or more processing devices to perform operations comprising: receiving, by the computing device, a plurality of input activations, the input activations being provided, at least in part, from a source external to the computing device; and determining, by the controller, whether each of the plurality of input activations is one of a zero value or a non-zero value. The operations further comprise, storing, in a memory bank of the computing device, at least one of the input activations, wherein storing the at least one input activation includes generating an index comprising one or more memory address locations having input activation values that are non-zero values. The operations still further comprise, providing, by the controller and from the memory bank, at least one input activation onto a data bus that is accessible by one or more units of a computational array, wherein the activations are provided, at least in part, from a memory address location associated with the index.

[0011] The subject matter described in this specification can be implemented in particular embodiments so as to realize one or more of the following advantages. Activations accessible from a first memory and a weights accessible from a second memory, in a single compute system, can be traversed based on memory address values retrieved from registers. A controller of the compute system can compress activation data by storing only non-zero values in first memory, thereby saving memory storage space and corresponding bandwidth. Matrix multiplications occur in the compute system based, in part, on primarily providing non-zero input activations. Moreover, when the compute system uses a communication scheme that includes primarily non-zero activation values, computational efficiency can be enhanced or accelerated by eliminating multiplication by zeros.

[0012] Other implementations of this and other aspects include corresponding systems, apparatus, and computer programs, configured to perform the actions of the methods,

encoded on computer storage devices. A system of one or more computers can be so configured by virtue of software, firmware, hardware, or a combination of them installed on the system that in operation cause the system to perform the actions. One or more computer programs can be so configured by virtue of having instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

[0013] The subject-matter described in this specification also relate to an image recognition and/or classification method/system. The system(s) can be implemented using the disclosed techniques for exploiting input data sparsity when computing units of a hardware computing system process inputs for a neural network layer to perform inference computations.

[0014] The details of one or more implementations of the subject matter described in this specification are set forth in the accompanying drawings and the description below. Other potential features, aspects, and advantages of the subject matter will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIG. 1 illustrates an example computation structure that includes activations and parameters.

[0016] FIG. 2 illustrates an example computation structure that includes activations and multiple parameter structures for an output depth greater than one.

[0017] FIG. 3 illustrates an example computation system for feeding input activations to one or more parameters.

[0018] FIG. 4 illustrates an example architecture that includes a memory unit providing input activations to one or more multiply accumulate (MAC) operators.

[0019] FIG. 5 is an example flow chart of a process for reducing parameter computations and exploiting input data sparsity.

[0020] Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0021] The subject matter described in this specification relates to reducing computations that occur within a compute unit or tile of an example neural network hardware computing system. In general, as part of computing a neural network inference, an input activation is

multiplied with a parameter or weight value to produce an output activation. Herein, inputs and input activation can refer to a data element included in a multi-dimensional data structure such as a tensor, matrix and/or data array commonly used in neural networks. Due to the algorithmic properties of computing inferences for deep neural networks, a large fraction of input activations are zero. In other words, current compute units perform a large number of unnecessary computations that include multiplying one number (e.g., a weight) against zero (input activation value).

[0022] This specification describes, in part, a more efficient activation storage and communication scheme as well as a custom architecture design for deep neural network processing, especially for processing convolutional layers of a neural network. Unlike conventional hardware accelerators that perform dense matrix multiplication over time, this specification describes an architecture that can 1) skip or bypass a computation upon seeing zero input values; and 2) reduce memory usage in a compute unit by storing compressed input activations that include only non-zero values. Overall, through the teachings of this specification, compute unit performance for neural network inference computations is improved and energy savings are realized by skipping unnecessary computations.

[0023] FIG. 1 illustrates an example computation structure 100 that includes an activation structure 102 and a parameter structure 104. Activation structure 102 can include a first data structure 102a including multiple data elements corresponding to a first input depth (denoted by subscript 0). Likewise, activation structure 102 can also include a second data structure 102b including multiple data elements corresponding to a second input depth (denoted by subscript 1). The multiple data elements shown in data structure 102a and 102b are indicated as a_0, b_0, c_0, d_0 and a_1, b_1, c_1, d_1, respectively. Each data element (a_0, a_1, b_0, b_1, c_0, d_0 and etc.) of the data structure 102a/b is an input activation value and each input depth corresponds to a depth of an input to a neural network layer. In some implementations, a neural network layer can have an input depth of one while in other implementations a neural network layer can have an input depth of more than one.

[0024] Parameter structure 104 can be described in a similar manner as activation structure 102. Parameter structure 104 includes a first data structure 104a and a second data structure 104b. Each data structure 104a/b can include multiple data elements that each contain kernel values. As shown in FIG. 1, the multiple data elements corresponding to data structure 104a and 104b are indicated as x_0, y_0, z_0 and x_1, y_1, z_1, respectively.

[0025] As discussed above, each layer of the neural network generates an output from a received input in accordance with values of a respective set of operands. Like other neural

network layers, each convolutional layer can process a set of activation inputs that can be represented as a matrix structure. A convolutional neural network layer will also have an associated set of kernels that includes values and the kernels can also be represented as a matrix structure of weights. In FIG. 1, activation structure 102 can correspond to a matrix structure having one or more activation inputs and parameter structure 104 can correspond to a matrix structure having one or more kernels or weight parameters.

[0026] As described in more detail below, various layers of a neural network process machine learning inferences by performing large quantities of computations that include matrix multiplications. Computation processes performed within a neural network layer (e.g., a convolutional layer) can include multiplying an input activation (i.e., a first operand) with a weight (i.e., a second operand) on one or more cycles and performing an accumulation of products over many cycles. An output activation is generated based on multiply and accumulation operations performed on the two operands.

[0027] As shown, equation 106 provides an example series-sequence based mathematical operation that can be performed when an input activation associated with a certain data element of activation structure 102 is multiplied with a kernel value or weight/parameter associated with a certain data element of parameter structure 104. For example, in equation 106, when index "i" equals 0, the input activation associated with data element a_0 of activation structure 102 is multiplied with the weight/parameter associated with data element x_0 of parameter structure 104. Moreover, because equation 106 is, in part, a series based equation, additional multiply operations will occur between sets of operands that correspond to other data elements of activation structure 102 and parameter structure 104. In some implementations, multiplication of a set of operands can be produce a partial sum 106a/b for a particular output feature or activation. Hence, as shown in equation 106, partial sums can be added to produce an output feature.

[0028] Neural networks can be embodied in one or more hardware computing systems that include multiple computing units configured to accelerate machine learning inference workloads of a network layer. Each computing unit can process a sub-set of computations for a given layer. In some implementations, structure 100 can be embodied in one or more computing units that each include at least two memory banks and one or more multiply accumulate (MAC) cells that can collectively form a MAC array (described below).

[0029] In one example, a first memory bank 108 of an example computing unit stores data associated with activation structure 102 and can be configured to receive and write input activation values to memory address locations within memory bank 108. Likewise, a second

memory bank 110 of the example computing unit stores data associated with parameter structure 104 and can be configured to receive and write weight values to memory address locations within memory bank 110. In this example, each element (e.g., a_0, b_0, c_0, d_0) of data elements 102a can be stored at a respective memory address of first memory bank 108. Similarly, each element (e.g., x_0, y_0, z_0) of data elements 104a can be stored at a respective memory address of second memory 110.

[0030] In some implementations, first memory bank 108 and second memory bank 110 are each a volatile memory unit or units. In some other implementations, memory bank 108 and memory bank 110 are each a non-volatile memory unit or units. Memory banks 108, 110 can also be another form of a computer-readable storage medium, such as a floppy disk device, a hard disk device, an optical disk device, or a tape device, a flash memory or other similar solid state memory device, or an array of devices, including devices in a storage area network or other configurations.

[0031] In general, a computing unit of a hardware computing system can include one or more registers to keep track of memory address values. The data elements of the matrix structure corresponding to activation structure 102 can be accessed from first memory bank 108 while data elements of the matrix structure corresponding to parameter structure 104 can be accessed from second memory bank 110. An example control device of the computing tile/compute unit can access and/or traverse data elements of the matrix structures based on address values that are accessible from the one or more registers. An example compute unit/tile including an example control device, activation structure 102, parameter structure 104, first memory bank 108, and second memory bank 110 are described more detail below with reference to FIG. 3.

[0032] Moreover, additional details and descriptions relating to hardware computing systems for accelerating neural network tensor computations and matrix-multiplications for neural network inference workloads are described in U.S. Patent Application No. 15/335,769, entitled "Neural Network Compute Tile," filed on October 27, 2016. The entire disclosure of U.S. Patent Application No. 15/335,769 is expressly incorporated by reference herein in its entirety.

[0033] FIG. 2 illustrates an example computation structure 200 that includes an activation structure and multiple parameter structures for an output feature depth greater than one. In some implementations, a neural network can have multiple layers that generate outputs that have multiple output feature depths. In some implementations, each parameter structure can be responsible for a respective one of the output depths. Hence, computation structure 200

depicts a scalable computing structure in which additional parameter structures 104 a/b/c are added to facilitate computations associated with N number of output depths. N is a variable and can have an integer value that ranges from, for example, 1 to 5, or alternatively, 1 to N depending the preferences or needs of a computing system designer.

[0034] As shown by data path 105, individual input activation values for elements associated with data structure 102a can be fed to each parameter structure 104a/b/c for use in computations performed by multiply operators associated with respective parameter structures 104. Each parameter structure 104 can then pass an activation value received from its left neighbor to its right neighbor in a pipelined manner. Alternatively, activations can be provided and consumed by each parameter structure 104 at the same time.

[0035] Matrix 202 can represent an example matrix structure that corresponds to activations 102. More specifically, element row 202a can correspond to data structure 102a and element row 202b can correspond to data structure 102b. In general, and by way of example, a first parameter structure 104 (1) is accessed to perform computations associated with space 206 and a second parameter structure 104 (2) is accessed to perform computations associated with space 208. Although not shown, additional computations can also be performed corresponding to the z – dimension. As an example, element row 202a can be in an R plane of an RGB image and element row 202b can be in a G plane of the same RGB image. An example convolutional layer of a neural network typically produces multiple output features. Example output features can include an output feature for classifying an apple, and another output feature for classifying a banana. Regarding data structures 204, space(s) 206 and 208 can represent different planes for different classifications.

[0036] FIG. 3 illustrates an example compute system 300 for feeding input activations to one or more parameter structures. Compute system 300 generally includes a controller 302 that provides one or more control signals 310 to cause input activations for activation structure 102 to be either stored to or retrieved from a memory address of memory bank 108. Likewise, controller 302 also provides one or more control signals 310 to cause weights for parameter structure 104a/b/c to be either stored to or retrieved from a memory address of memory bank 110. Compute system 300 further includes one or more multiply accumulate (MAC) cell/unit(s) 304, an input activation bus 306 and an output activation bus 308. Control signals 310 can, for example, cause memory bank 108 to provide one or more input activations unto input activation bus 306, cause memory bank 110 to provide one or more weights to parameter structure 104 a/b/c, and/or cause MAC unit 304 to perform computations that produce output activations that are provided to output activation bus 308.

[0037] Controller 302 can include one or more processing units and memory. In some embodiments, processing units of controller 302 can include one or more processors (e.g., microprocessors or central processing units (CPUs)), graphics processing units (GPUs), application specific integrated circuits (ASICs), or a combination of different processors. In alternative embodiments, controller 302 can include other storage or computing resources/devices (e.g., buffers, registers, control circuitry, etc.) that provide additional processing options for performing one or more of the determinations and calculations described in this specification.

[0038] In some implementations, processing unit(s) of controller 302 executes programmed instructions stored in memory to cause controller 302 and compute system 300 to perform one or more functions described in this specification. The memory of controller 302 can include one or more non-transitory machine-readable storage mediums. The non-transitory machine-readable storage medium can include solid-state memory, magnetic disk, and optical disk, a portable computer diskette, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (e.g., EPROM, EEPROM, or Flash memory), or any other tangible medium capable of storing information.

[0039] In general, compute system 300 is an example compute unit or tile and can include additional hardware structures to perform computations associated with multi-dimensional data structures such as tensors, matrices and/or data arrays. In some implementations, input activation values can be pre-loaded to memory bank 108 for activation structure 102 and weight values can be pre-loaded to second memory bank 110 using data values received by compute system 300 that arrive at a compute system 300 from an external or higher level control device associated with a neural network hardware computing system.

[0040] Instructions, inputs or input activations, and weights can be provided to system 300 from an external source, such as an external input/output (I/O) device or a high level control device associated with a neural network hardware computing system. In some implementations, one or more data buses provide data communications between the external source (e.g., a control device) and system(s) 300. The data buses are used to provide instructions, inputs or activations, and weights from an example I/O device to each of multiple systems 300 or between multiple compute tiles (e.g., multiple systems 300) included in a hardware computing system for a neural network.

[0041] System 300 can receive instructions that define a particular compute operation to be performed by system 300. Moreover, controller 302 can execute programed instructions to, for example, analyze a data stream associated with the received input activations. Analyzing

the input activation data stream can enable controller 302 to detect or determine whether a value associated with each of the input activations is a zero value or a non-zero value. In some implementations, controller 302 analyzes an example input activation data stream and maps each detected zero activation value and each detected non-zero activation value to bitvector or bitmap 303.

[0042] As shown in FIG. 3, bitmap 303 can use binary values to map detected zero value input activations and detected non-zero value input activations. For example, a binary value of "0" can correspond to a detected zero input activation value and a binary value of "1" can correspond to a detected non-zero input activation value. For example, bitmap 303 can be an 8-bit bitmap in which odd numbered bit positions that include a binary "1" correspond to non-zero activation values and even numbered bit positions that include a binary "0" correspond to zero activation values.

[0043] Controller 302 can cause input activations to be stored in memory bank 108. In general, data values stored in memory bank 108 are typically each written to a respective memory address location. The address location in memory bank 108 can then be accessed by an example control device (e.g., controller 302) when a data value such as an input activation is needed to perform a particular compute operation. Controller 302 can use bitmap 303 to create an index of memory address locations that include non-zero input activation values.

[0044] In some implementations, controller 302 uses bitmap 303 to determine which input activations to write to memory bank 108. For example, analysis of bitmap 303 can indicate that only activation values corresponding to bitmap positions 1, 3, 5, 7 (non-zero values) should be written to address locations in memory bank 108. Moreover, data values associated with bitmap positions 2, 4, 6, 8 (zero values) can either be discarded or written to memory address locations which may or may not be accessed by controller 302 when activation values are provided to input bus 306. Thus, bitmap 303 can be used as a basis to compress zero activation values in which compression occurs when zero value input activations are not written to memory address locations, thereby reducing the overall memory usage and freeing address locations for storing other data values.

[0045] Controller 302 can provide one or more control signals 310 to memory bank 108 to load input activations, from memory bank 108, onto input activation bus 306 and provide the values to an array of computational units that include MAC 304. In some implementations, bitmap 303, or the non-zero memory address index that corresponds to bitmap 303, can be referenced by controller 302 so as to determine which memory address values should be

accessed to provide non-zero activation values. Activation values are provided by controller 302 from memory bank 108 and onto data bus 306.

[0046] In some implementations, the input activations are provided, at least in part, from a memory address location associated with the index or bitmap 303. In other implementations, controller 302 can detect or determine, based on one of bitmap 303 or the index, whether an input activation that is provided has a zero value. In response to making this determination, controller 302 can then provide a control signal to unit or MAC 304 in the computational array to prevent, stall, or otherwise inhibit the occurrence of an unnecessary multiply operation (e.g., a multiply by zero). Within compute system 300, energy savings can be realized from providing a zero activation value and subsequently or simultaneously disabling a compute operation associated with that activation.

[0047] As discussed above, the index includes all memory address locations having input activations with non-zero values. Data bus 306 is accessible by one or more units of a computational array. The units of the computational array can receive, from data bus 306, one or more non-zero activation values to perform computations relating to matrix multiplication based on the received activation values. In some implementations, compute system 300 will only provide input activations from memory address locations that correspond to the indexed addresses. Thus, no zero activations will be provided to input bus 306 and so, no compute operations will be disabled or otherwise prevented from occurring. When compute system 300 uses this communication scheme, computational efficiency can be enhanced or accelerated by eliminating multiplication by zeros.

[0048] For a given compute cycle, compute system 300 can require access to an element of activation structure 102 and parameter structure 104 to execute multiplication operations associated with inference computations for a neural network layer. As noted above, certain memory address values for memory bank 108 and memory bank 110 can correspond to elements of activation structure 102 and parameter structure 104 respectively.

[0049] For a cycle in which computations are performed, controller 302 will provide one input activation value at a time and the array of computational units including MAC cell 304 will multiply an activation with a weight to produce different output activations for a given input activation. Each element (described above as a parameter structure) or MAC cell 304 of the array of computational units can be responsible for different output depths of a neural network layer. In general, whenever controller 302 detects a zero activation value, controller 302 can either: 1) not store that activation value in memory bank 108; 2) not provide the activation value, or 3) provide the value and provide a control signal to a particular

computational unit to cause the unit to not perform a multiply operation corresponding to that zero activation value.

[0050] The array of computational units is fully controlled by controller 302, which can determine, based on detection of a zero activation value, when there is a need to skip or prevent a particular computation. Thus, there is no need for additional complex hardware structures within the array of computational units to skip a particular computation. Furthermore, input activation values can be analyzed upon arriving at compute system 300 for storage in memory bank 108. In response to analyzing the input activations, controller 302 can execute an instruction to efficiently compress activation data by storing only non-zero values in memory 108, thereby saving memory storage space and corresponding bandwidth.

[0051] When compute system 300 receives input activations and weights, controller 302 can, for example, execute one or more direct memory access operations. Execution of these memory access operations includes storing, in address locations of memory bank 108, input activations corresponding to dimensional elements of activation structure 102. Likewise, controller 302 can also store, in address locations of memory bank 110, parameters corresponding to dimensional elements of parameter structure 104. In addition to bitmap 303, controller 302 can further include one or more address registers that maintain the memory addresses from which a particular input activation (e.g., having a zero value or non-zero value) will be fetched. Moreover, the one or more registers will also store the memory addresses from which a corresponding weight is fetched to be multiplied with the particular input activation.

[0052] As discussed above, controller 302 identifies memory addresses for non-zero activation values based, in part, on bitmap 303. In some implementations, controller 302 reads bitmap 303 and determines, for example, at least two memory addresses that have non-zero activation values. If controller 302 is configured to provide, and subsequently skip or disable computes for, zero activation values, then controller 302 may also determine at least one memory address that has a zero activation value. In this implementation, controller 302 can reference the above mentioned registers to determine a corresponding weight (and memory address) for the first input activation and to determine a corresponding weight (and memory address) for the second input activation.

[0053] As noted above, controller 302 maintains one or more address registers in memory. So, to mitigate or prevent any potential misalignment of operands (input activation and weight), upon detection of the zero value input activation, controller 302 can disable the

corresponding compute unit, skip loading a particular weight, and retrieve the appropriate corresponding weight (and memory address) for the next non-zero input activation to resume computing output activations for a given neural network layer.

[0054] In some implementations, the output activations computed at a first neural network layer are used as input activations to a next second layer in the network, e.g., a next hidden layer or the output layer of the network. In general, each layer of the neural network generates an output from a received input in accordance with current values of a respective set of parameters. In some instances, controller 302 can execute programed instructions (i.e., output logic) to analyze data streams associated with output activations provided to output activation bus 308. Analyzing the output activation data stream can enable controller 302 to detect or determine whether a value associated with each of the output activations is a zero value or a non-zero value. Controller 302 can analyze an example output activation data stream and map each detected non-zero activation value to bitmap 305. Mapped non-zero activation values in bitmap 305 can be used to supply only non-zero values as input activations to a subsequent compute system 300 that is responsible for computations associated with the next second layer in the network.

[0055] In alternative implementations, there can be some compute operations in which a single non-zero input activation is used as an operand for several multiply operations covering a variety of weights for a given dimensional element of parameter structure 104 (i.e., iterate a "x" or "y" dimension). For example, when controller 302 causes memory bank 108 to provide a first input activation (e.g., non-zero value), parameter structure 104a receives the activation and a corresponding weight at a given address is also loaded to parameter structure 104a. Parameter structure 104a will proceed to update a particular number of partial sums (e.g., denoted by variable "K") that the first input activation affects over K compute cycles. As a result, for these K cycles, parameter structure 104a will receive no additional input activations. Controller 302 can then provide a control signal to memory bank 108 to cause the next input activation to be provided to input activation bus 306.

[0056] FIG. 4 illustrates an example architecture that includes a memory bank 108 that provides activations 404 via input bus 306 to one or more multiply accumulate (MAC) operators. A shift register 404 can provide shift functionality whereby activations 404 are sent out one at a time onto input bus 306 for receipt by one or more MAC operators in a MAC cell 304. As shown, in one implementation, activation 406 may have an activation value of zero and, therefore, may not be consumed by a MAC cell 304.

[0057] In general, MAC cells 304 comprising MAC operators are defined as compute units that calculate a partial sum and, in some implementations, are configured to write a partial sum datum to output bus 308. As shown, cells 304 may consist of one or more MAC operators. In one implementation, the number of MAC operators in MAC cell 304 is referred to as the issue width of the cell. As an example, a dual issue cell refers to a cell with two MAC operators that can compute the multiplication of two activations values (from memory bank 108) with two parameters (from memory 110) and perform an addition between the results of the two multipliers and the current partial sum.

[0058] As described above, input bus 306 is a communication bus that provides input activations to MAC operators of the linear unit (i.e., MAC array 304). In some implementations, the same input is shared between all MAC operators. The width of input bus 306 must be wide enough to supply the input activations to the corresponding number of cells for a given MAC array 304. Consider the following example to illustrate the structure of input bus 306. When the number of cells in the linear unit equals four and the activation width equals eight bits, input bus 306 can be configured to provide up to four input activations every cycle. In this example, every cell in MAC array 304 will only access one out of the four activations that are provided.

[0059] In some examples, instruction data 312 can indicate that cells of MAC array 304 will need to perform computations using the same input activation. This may be referred to as Zout partitioning within a cell of MAC array 304. Likewise, Zin partitioning within a cell occurs when cells of MAC array 304 need different activations to perform computations. In the former case, the single input activation is replicated four times and four activations read from memory bank 108 are provided over four cycles. In the latter case, a read of memory bank 108 is required every cycle.

[0060] FIG. 5 is an example flow chart of process for reducing parameter computations and exploiting input data sparsity. At block 502, compute system 300 receives input activations that have either a zero activation value or a non-zero activation value. As discussed above, in some implementations, compute system 300 can receive input activations from a host interface device or higher level controller of an example neural network hardware system.

[0061] At block 504, controller 302 determines whether each of the input activations is a zero value or a non-zero value. In some implementations, controller 302 analyzes an input activation data stream and maps each detected zero value and non-zero value to bitmap 303

that includes binary values that correspond to zero input activation values ("0") and non-zero input activation values ("1").

[0062] At block 506, controller 302 stores, in memory bank 108, received input activations. Storing the input activation can include controller 302 generating an index of one or more memory address locations having input activations that include non-zero values. In some implementations, the index is created based on bitmap 303. For example, because each bit of bitmap 303 indicates either a non-zero activation value or a zero activation value, bitmap 303 can be referenced by controller 302 to create an index of memory address locations having non-zero values when writing input activations to memory bank 108.

[0063] At block 508, controller 302 provides, from memory bank 108, at least one input activation onto data bus 306. In some implementations, the input activations are provided, at least in part, from a memory address location identified in the index. As discussed above, the index identifies all memory address locations storing input activations with non-zero values. Data bus 306 is accessible by one or more units of a computational array. The units of the computational array receive, from data bus 306, one or more non-zero activation values to perform computations relating to matrix multiplication. In some implementations, compute system 300 will only provide input activations from memory addresses that correspond to the indexed addresses. When compute system 300 uses this communication scheme, computational efficiency can be enhanced by eliminating multiplication by zeros.

[0064] At block 510, in implementations where all activation values are provided rather than only input activations from indexed addresses, controller 302 detects that an input activation is provided from a memory address that is not associated with any indexed addresses that include non-zero activation values. In response to this detecting step, controller 302 can then provide a control signal to at least one unit of the computational array to prevent a multiply operation associated with the zero input. When compute system 300 uses this communication scheme, energy savings can be realized by preventing unnecessary or wasteful computations that yield no useful results (e.g., useful results includes compute of a partial sum or output activation).

[0065] Embodiments of the subject matter and the functional operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, i.e., one or more modules of computer program instructions

encoded on a tangible non transitory program carrier for execution by, or to control the operation of, data processing apparatus. Alternatively or in addition, the program instructions can be encoded on an artificially generated propagated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, which is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus. The computer storage medium can be a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them.

[0066] The processes and logic flows described in this specification can be performed by one or more programmable computers executing one or more computer programs to perform functions by operating on input data and generating output(s). The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array), an ASIC (application specific integrated circuit), a GPGPU (General purpose graphics processing unit), or some other processing unit.

[0067] Computers suitable for the execution of a computer program include, by way of example, can be based on general or special purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a central processing unit for performing or executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices.

[0068] Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[0069] While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment.

Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[0070] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

[0071] Further implementations are summarized in the following examples:

[0072] Example 1: A computer-implemented method, comprising: receiving, by a computing device, a plurality of input activations , the input activations being provided, at least in part, from a source external to the computing device ; determining, by a controller of the computing device, whether each of the plurality of input activations has one of a zero value or a non-zero value; storing, in a memory bank of the computing device, at least one of the input activations; generating, by the controller, an index comprising one or more memory address locations having input activation values that are non-zero values; and providing, by the controller and from the memory bank, at least one input activation onto a data bus that is accessible by one or more units of a computational array, wherein the activations are provided, at least in part, from a memory address location associated with the index.

[0073] Example 2: The method of example 1, wherein the index is created based on a bitmap comprising a plurality of bits and, wherein each bit of the bitmap indicates at least one of a non-zero input activation value or a zero input activation value.

[0074] Example 3: The method of example 1 or 2, further including, providing a first input activation that has a non-zero value to perform, by at least one unit, a computation using the non-zero value, and subsequently providing a second input activation that has a zero value, and preventing, in at least one unit, computation that would otherwise be performed using the zero value.

[0075] Example 4: The method of example 3, wherein preventing occurs in response to the controller determining that the input activation is provided from a memory address location that is not associated with the index.

[0076] Example 5: The method of example 4, further including, detecting, by the controller, that the input activation is provided from a memory address location that is not associated with the index, and, in response to detecting, providing a control signal to at least one unit of the computational array to prevent a multiply operation associated with the zero input activation value.

[0077] Example 6: The method of one of examples 1 to 5, wherein the method further comprises, mapping, by the controller and to a first unit, a first portion of a tensor computation that uses a first input activation and mapping, to a second unit that differs from the first unit, a second portion of the tensor computation that also uses the first input activation.

[0078] Example 7: The method of one of examples 1 to 6, further comprising, sequentially providing a single input activation onto the data bus, the single input activation being accessed and selected from memory address locations that are associated with the index.

[0079] Example 8: The method of one of examples 1 to 7, wherein providing further comprises, not providing input activations that have a zero value.

[0080] Example 9: One or more machine-readable storage devices storing instructions that are executable by one or more processing devices to perform operations comprising: receiving, by a computing device, a plurality of input activations, the input activations being provided, at least in part, from a source external to the computing device; determining, by a controller of the computing device, whether each of the plurality of input activations has one of a zero value or a non-zero value; storing, in a memory bank of the computing device, at least one of the input activations; generating, by the controller, an index comprising one or more memory address locations having input activation values that are non-zero values; and providing, by the controller and from the memory bank, at least one input activation onto a data bus that is accessible by one or more units of a computational array, wherein the activations are provided, at least in part, from a memory address location associated with the index.

[0081] Example 10: The machine-readable storage devices of example 9, wherein the index is created based on a bitmap comprising a plurality of bits and, wherein each bit of the bitmap indicates at least one of a non-zero input activation value or a zero input activation value.

[0082] Example 11: The machine-readable storage devices of example 9 or 10, further including, providing a first input activation that has a non-zero value to perform, by at least one unit, a computation using the non-zero value, and subsequently providing a second input activation that has a zero value, and preventing, in at least one unit, computation that would otherwise be performed using the zero value.

[0083] Example 12: The machine-readable storage devices of example 11, wherein preventing occurs in response to the controller determining that the input activation is provided from a memory address location that is not associated with the index.

[0084] Example 13: The machine-readable storage devices of example 12, further including, detecting, by the controller, that the input activation is provided from a memory address location that is not associated with the index, and, in response to detecting, providing a control signal to at least one unit of the computational array to prevent a multiply operation associated with the zero input activation value.

[0085] Example 14: The machine-readable storage devices of one of examples 9 to 13, wherein the operations further comprise, mapping, by the controller and to a first unit, a first portion of a tensor computation that uses a first input activation and mapping, to a second unit that differs from the first unit, a second portion of the tensor computation that also uses the first input activation.

[0086] Example 15: An electronic system comprising: a controller disposed in a computing device, the controller including one or more processing devices; and one or more machine-readable storage devices for storing instructions that are executable by the one or more processing devices to perform operations comprising: receiving, by the computing device, a plurality of input activations, the input activations being provided, at least in part, from a source external to the computing device; determining, by the controller, whether each of the plurality of input activations has one of a zero value or a non-zero value; storing, in a memory bank of the computing device, at least one of the input activations; generating, by the controller, an index comprising one or more memory address locations having input activation values that are non-zero values; and providing, by the controller and from the memory bank, at least one input activation onto a data bus that is accessible by one or more units of a computational array, wherein the activations are provided, at least in part, from a memory address location associated with the index.

[0087] Example 16: The electronic system of example 15, wherein the index is created based on a bitmap comprising a plurality of bits and, wherein each bit of the bitmap indicates at least one of a non-zero input activation value or a zero input activation value.

[0088] Example 17: The electronic system of example 15 or 16, further including, providing a first input activation that has a non-zero value to perform, by at least one unit, a computation using the non-zero value, and subsequently providing a second input activation that has a zero value, and preventing, in at least one unit, computation that would otherwise be performed using the zero value.

[0089] Example 18: The electronic system of example 17, wherein preventing occurs in response to the controller determining that the input activation is provided from a memory address location that is not associated with the index.

[0090] Example 19: The electronic system of example 17 or 18, further including, detecting, by the controller, that the input activation is provided from a memory address location that is not associated with the index, and, in response to detecting, providing a control signal to at least one unit of the computational array to prevent a multiply operation associated with the zero input activation value.

[0091] Example 20: The electronic system of one of examples 15 to 19, wherein the operations further comprise, mapping, by the controller and to a first unit, a first portion of a tensor computation that uses a first input activation and mapping, to a second unit that differs from the first unit, a second portion of the tensor computation that also uses the first input activation.

[0092] Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In certain implementations, multitasking and parallel processing may be advantageous.

What is claimed is:

1. A computer-implemented method, comprising:

receiving, by a computing device, a plurality of input activations, the input activations being provided, at least in part, from a source external to the computing device;

determining, by a controller of the computing device, whether each of the plurality of input activations has one of a zero value or a non-zero value;

storing, in a memory bank of the computing device, at least one of the input activations;

generating, by the controller, an index comprising one or more memory address locations having input activation values that are non-zero values; and

providing, by the controller and from the memory bank, at least one input activation onto a data bus that is accessible by one or more units of a computational array, wherein the activations are provided, at least in part, from a memory address location associated with the index.

2. The method of claim 1, wherein the index is created based on a bitmap comprising a plurality of bits and, wherein each bit of the bitmap indicates at least one of a non-zero input activation value or a zero input activation value.

3. The method of claim 1, further including, providing a first input activation that has a non-zero value to perform, by at least one unit, a computation using the non-zero value, and subsequently providing a second input activation that has a zero value, and preventing, in at least one unit, computation that would otherwise be performed using the zero value.

4. The method of claim 3, wherein preventing occurs in response to the controller determining that the input activation is provided from a memory address location that is not associated with the index.

5. The method of claim 4, further including, detecting, by the controller, that the input activation is provided from a memory address location that is not associated with the index, and, in response to detecting, providing a control signal to at least one unit of the

computational array to prevent a multiply operation associated with the zero input activation value.

6. The method of claim 1, wherein the method further comprises, mapping, by the controller and to a first unit, a first portion of a tensor computation that uses a first input activation and mapping, to a second unit that differs from the first unit, a second portion of the tensor computation that also uses the first input activation.

7. The method of claim 1, further comprising, sequentially providing a single input activation onto the data bus, the single input activation being accessed and selected from memory address locations that are associated with the index.

8. The method of claim 1, wherein providing further comprises, not providing input activations that have a zero value.

9. One or more machine-readable storage devices storing instructions that are executable by one or more processing devices to perform operations comprising:

receiving, by a computing device, a plurality of input activations, the input activations being provided, at least in part, from a source external to the computing device;

determining, by a controller of the computing device, whether each of the plurality of input activations has one of a zero value or a non-zero value;

storing, in a memory bank of the computing device, at least one of the input activations;

generating, by the controller, an index comprising one or more memory address locations having input activation values that are non-zero values; and

providing, by the controller and from the memory bank, at least one input activation onto a data bus that is accessible by one or more units of a computational array, wherein the activations are provided, at least in part, from a memory address location associated with the index.

10. The machine-readable storage devices of claim 9, wherein the index is created based on a bitmap comprising a plurality of bits and, wherein each bit of the bitmap indicates at least one of a non-zero input activation value or a zero input activation value.

11. The machine-readable storage devices of claim 9, further including, providing a first input activation that has a non-zero value to perform, by at least one unit, a computation using the non-zero value, and subsequently providing a second input activation that has a zero value, and preventing, in at least one unit, computation that would otherwise be performed using the zero value.

12. The machine-readable storage devices of claim 11, wherein preventing occurs in response to the controller determining that the input activation is provided from a memory address location that is not associated with the index.

13. The machine-readable storage devices of claim 12, further including, detecting, by the controller, that the input activation is provided from a memory address location that is not associated with the index, and, in response to detecting, providing a control signal to at least one unit of the computational array to prevent a multiply operation associated with the zero input activation value.

14. The machine-readable storage devices of claim 9, wherein the operations further comprise, mapping, by the controller and to a first unit, a first portion of a tensor computation that uses a first input activation and mapping, to a second unit that differs from the first unit, a second portion of the tensor computation that also uses the first input activation.

15. An electronic system comprising:

a controller disposed in a computing device, the controller including one or more processing devices; and

one or more machine-readable storage devices for storing instructions that are executable by the one or more processing devices to perform operations comprising:

receiving, by the computing device, a plurality of input activations, the input activations being provided, at least in part, from a source external to the computing device;

determining, by the controller, whether each of the plurality of input activations has one of a zero value or a non-zero value;

storing, in a memory bank of the computing device, at least one of the input activations;

generating, by the controller, an index comprising one or more memory address locations having input activation values that are non-zero values; and

providing, by the controller and from the memory bank, at least one input activation onto a data bus that is accessible by one or more units of a computational array, wherein the activations are provided, at least in part, from a memory address location associated with the index.

16. The electronic system of claim 15, wherein the index is created based on a bitmap comprising a plurality of bits and, wherein each bit of the bitmap indicates at least one of a non-zero input activation value or a zero input activation value.

17. The electronic system of claim 15, further including, providing a first input activation that has a non-zero value to perform, by at least one unit, a computation using the non-zero value, and subsequently providing a second input activation that has a zero value, and preventing, in at least one unit, computation that would otherwise be performed using the zero value.

18. The electronic system of claim 17, wherein preventing occurs in response to the controller determining that the input activation is provided from a memory address location that is not associated with the index.

19. The electronic system of claim 17, further including, detecting, by the controller, that the input activation is provided from a memory address location that is not associated with the index, and, in response to detecting, providing a control signal to at least one unit of the computational array to prevent a multiply operation associated with the zero input activation value.

20. The electronic system of claim 15, wherein the operations further comprise, mapping, by the controller and to a first unit, a first portion of a tensor computation that uses a first input activation and mapping, to a second unit that differs from the first unit, a second portion of the tensor computation that also uses the first input activation.

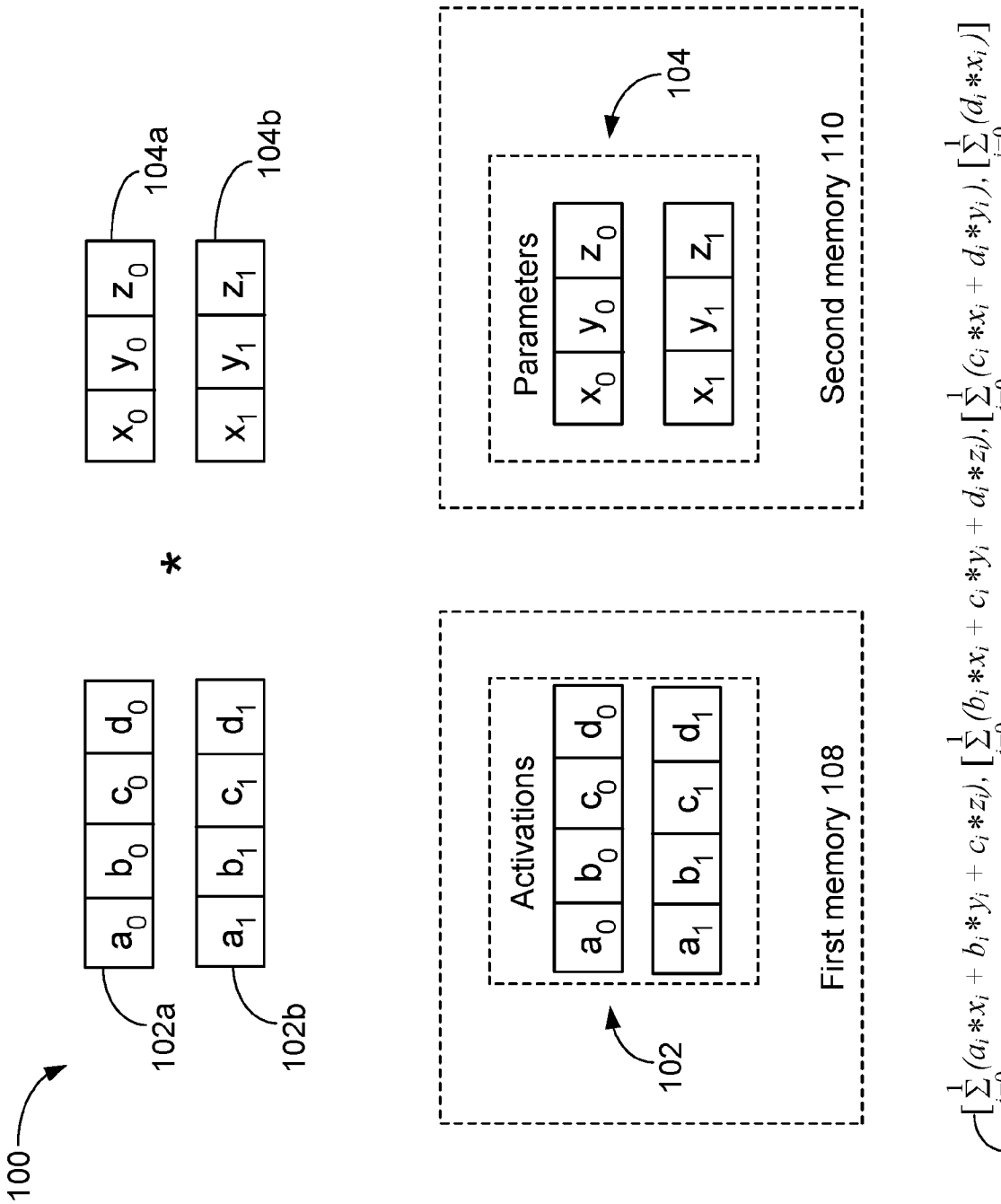
100
102a
102b
104a
104b
106
106a
106b
102
104
Activations
Parameters
First memory 108
Second memory 110
$[\sum_{i=0}^{1}(a_i * x_i + b_i * y_i + c_i * z_i), [\sum_{i=0}^{1}(b_i * x_i + c_i * y_i + d_i * z_i), [\sum_{i=0}^{1}(c_i * x_i + d_i * y_i), [\sum_{i=0}^{1}(d_i * x_i)]$

FIG. 1

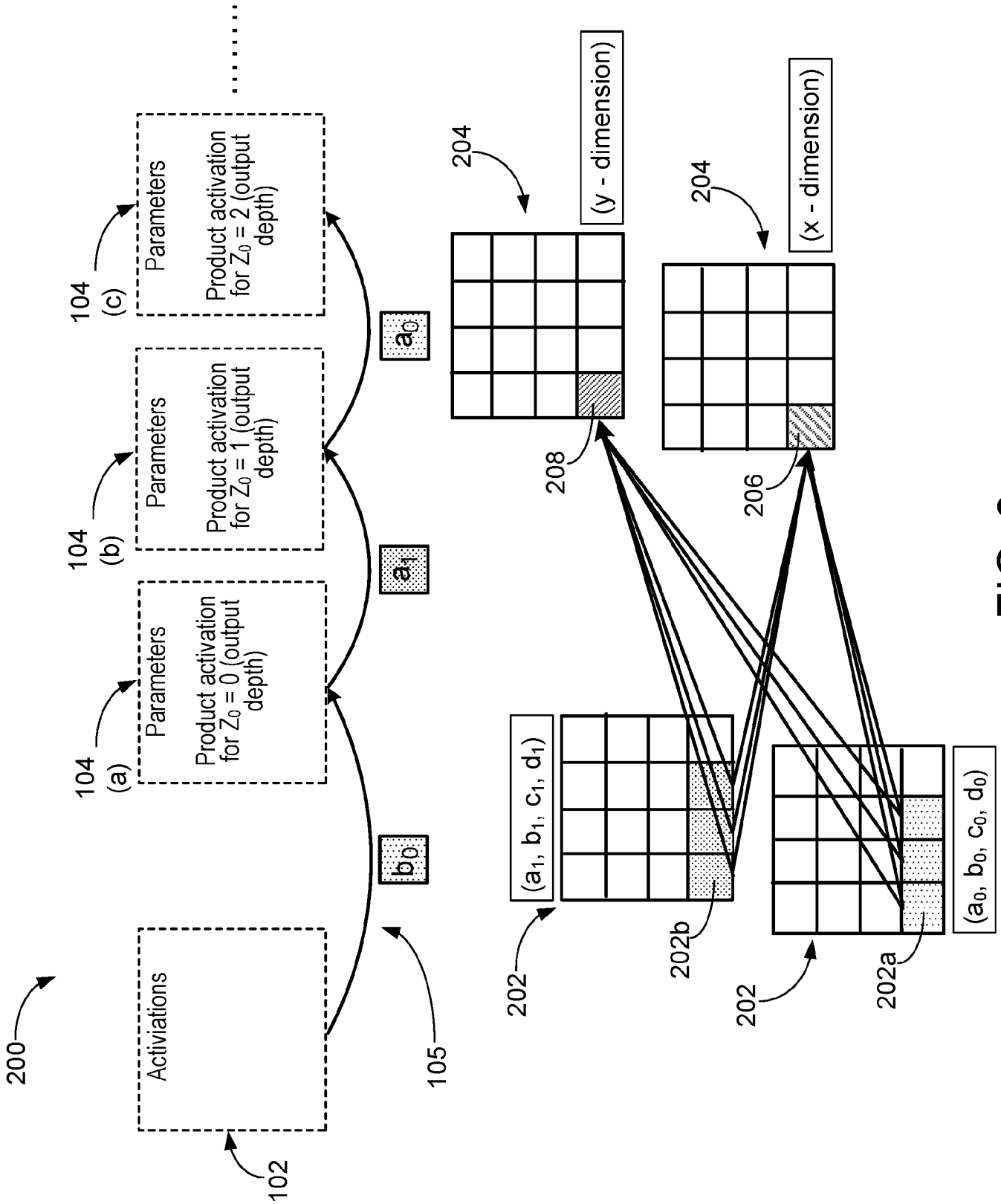
200
Activiations
102
105
104 (a)
Parameters
Product activation for Z_0 = 0 (output depth)
104 (b)
Parameters
Product activation for Z_0 = 1 (output depth)
104 (c)
Parameters
Product activation for Z_0 = 2 (output depth)
b_0
a_1
a_0
204
(y - dimension)
208
204
(x - dimension)
206
202
(a_1, b_1, c_1, d_1)
202b
202
202a
(a_0, b_0, c_0, d_0)

FIG. 2

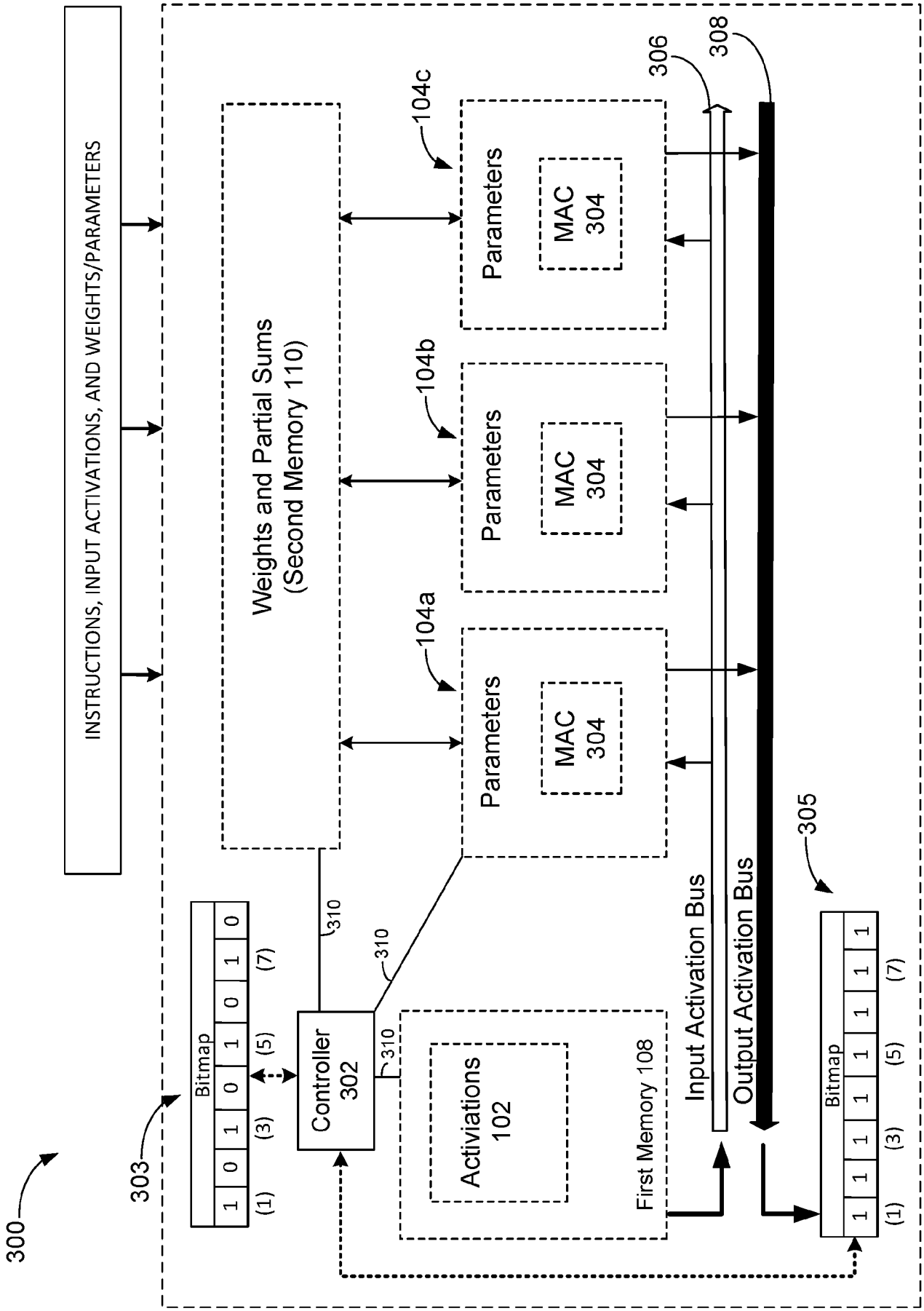
300
INSTRUCTIONS, INPUT ACTIVATIONS, AND WEIGHTS/PARAMETERS
303
Bitmap
1 0 1 0 1 0 1 0
(1) (3) (5) (7)
Controller 302
310
310
310
Weights and Partial Sums (Second Memory 110)
104a
104b
104c
Parameters
MAC 304
Activiations 102
First Memory 108
Input Activation Bus
Output Activation Bus
306
308
305
Bitmap
1 1 1 1 1 1 1 1
(1) (3) (5) (7)

FIG. 3

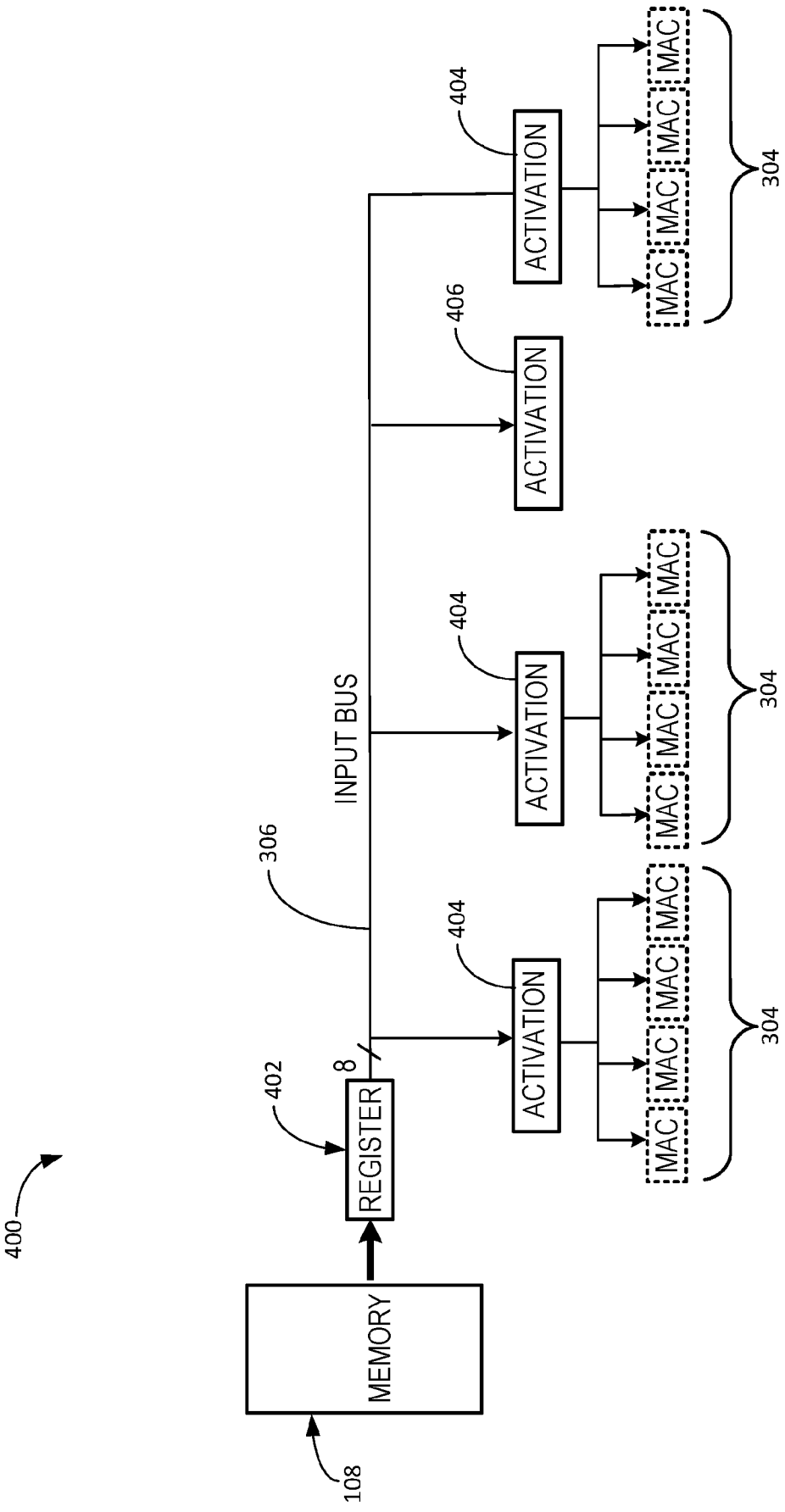
400
MEMORY
108
REGISTER
402
8
INPUT BUS
306
ACTIVATION
404
ACTIVATION
406
ACTIVATION
404
ACTIVATION
404
MAC
304

FIG. 4

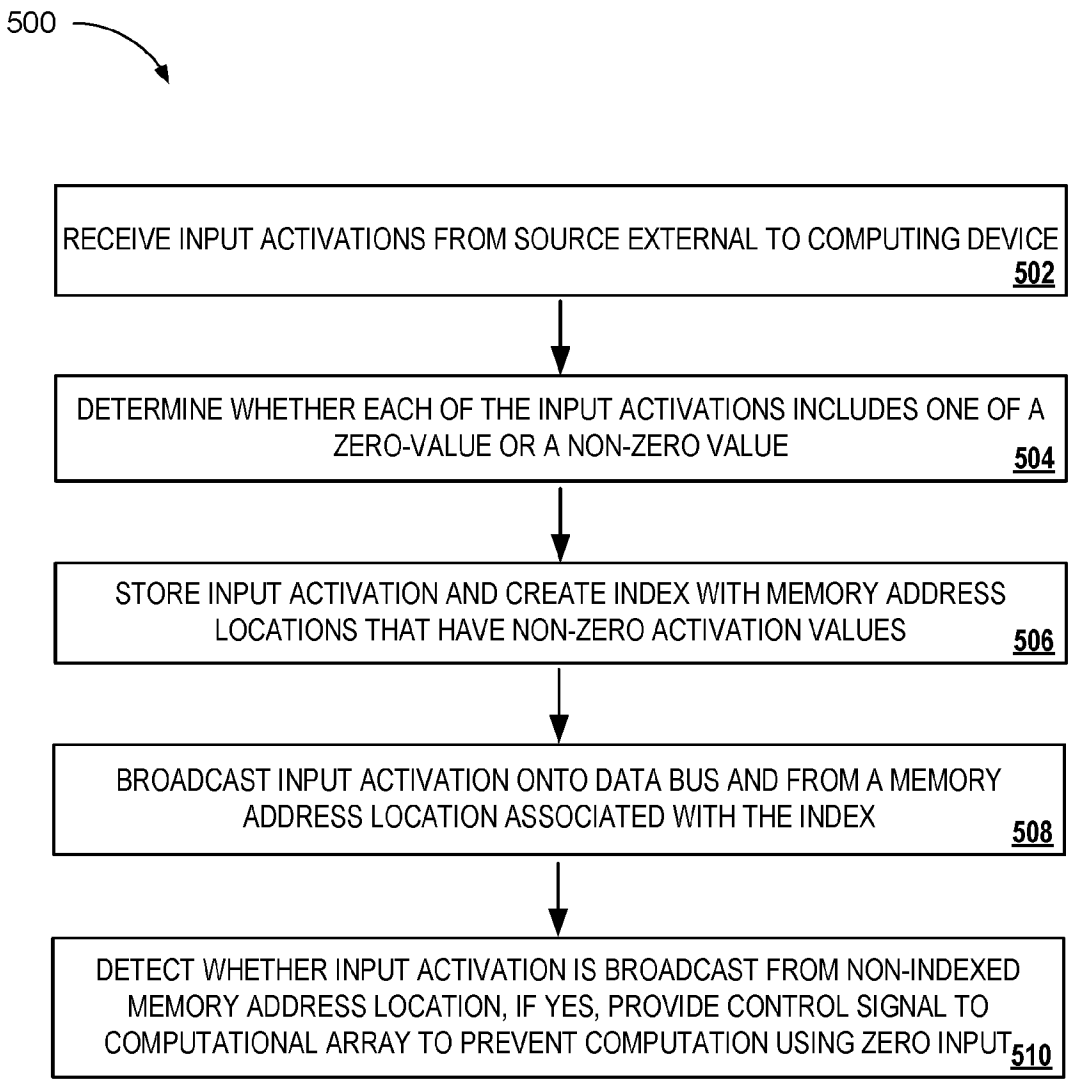
500
RECEIVE INPUT ACTIVATIONS FROM SOURCE EXTERNAL TO COMPUTING DEVICE
502
DETERMINE WHETHER EACH OF THE INPUT ACTIVATIONS INCLUDES ONE OF A ZERO-VALUE OR A NON-ZERO VALUE
504
STORE INPUT ACTIVATION AND CREATE INDEX WITH MEMORY ADDRESS LOCATIONS THAT HAVE NON-ZERO ACTIVATION VALUES
506
BROADCAST INPUT ACTIVATION ONTO DATA BUS AND FROM A MEMORY ADDRESS LOCATION ASSOCIATED WITH THE INDEX
508
DETECT WHETHER INPUT ACTIVATION IS BROADCAST FROM NON-INDEXED MEMORY ADDRESS LOCATION, IF YES, PROVIDE CONTROL SIGNAL TO COMPUTATIONAL ARRAY TO PREVENT COMPUTATION USING ZERO INPUT
510

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2017/047992

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06N3/10
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)
G06N

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	MOONS BERT ET AL: "A 0.3-2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets", 2016 IEEE SYMPOSIUM ON VLSI CIRCUITS (VLSI-CIRCUITS), IEEE, 15 June 2016 (2016-06-15), pages 1-2, XP032969303, DOI: 10.1109/VLSIC.2016.7573525 [retrieved on 2016-09-21] -----	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 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
9 November 2017	16/11/2017

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	Appeltant, Lennert

1