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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 September 2022 (22.09.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/197325 A1

(51) International Patent Classification:
G06F 17/15 (2006.01) *G06N 3/063* (2006.01)

(21) International Application Number:
PCT/US2021/053281

(22) International Filing Date:
03 October 2021 (03.10.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/162,516 18 March 2021 (18.03.2021) US
17/480,282 21 September 2021 (21.09.2021) US

(71) Applicant: **MICROCHIP TECHNOLOGY INC.** [US/US]; 2355 W. Chandler Blvd., Chandler, Arizona 85224 (US).

(72) Inventors: **DONTHU, Sathishkumar**; Flat No-305, Block-A, Sai keerthi estates, Friends colony, chandanagar, Hyderabad 500050 (IN). **REDDY, Battu Prakash**; 103, Plot 291, Eden homes, Road 23, Sardar Patel Nagar, Kukatpally, Hyderabad 500085 (IN).

(74) Agent: **GLASS, Kenneth**; Glass & Associates, 236 N. Santa Cruz Ave., Suite 243, Los Gatos, California 95030 (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, IT, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

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

(54) **Title:** METHOD AND APPARATUS FOR PERFORMING CONVOLUTION NEURAL NETWORK OPERATIONS USING 3X3 CONVOLUTION MATRIX

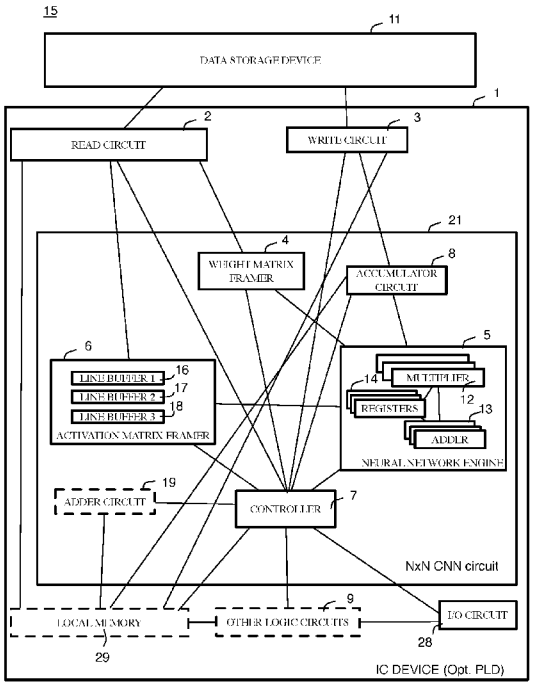

FIG. 1

(57) **Abstract:** A method and apparatus for performing a convolution of a NxN matrix. A weights matrix for a NxN Convolutional Neural Network (CNN) is received and is divided into 3x3 weights matrixes. Lines of image values are read and are stored in a buffer as sets of image values. A 3x3 convolution is performed to generate a 3x3 convolution value. All 3x3 convolution values associated with a particular NxN convolution and a particular set of image values are summed. The 3x3 convolutions and the summing are repeated until all columns in the set of image values have been processed; and the reading, the storing, the performing 3x3 convolutions, the summing and the repeating performing 3x3 convolutions are repeated until all lines of image values have been processed. The sums associated with a particular NxN convolution are added together to generate an NxN convolution value for each of the NxN convolutions.

WO 2022/197325 A1

Method and Apparatus for Performing Convolution Neural Network Operations Using 3x3 Convolution Matrix

CROSS-REFERENCE TO RELATED APPLICATIONS

[001] The present application claims priority to United States Provisional Patent Application Serial No. 63/162,516 filed on March 18, 2021 and United States Non-Provisional Patent Application Serial No. 17/480,282 filed on September 21, 2021, the contents of which are incorporated by reference herein in their entirety.

BACKGROUND OF THE INVENTION

[002] Convolution Neural Networks (CNNs) used in computer vision have multiple layers and use convolution as a basic operation. The convolution matrix is usually of the size 3x3, 5x5, 7x7, 9x9 or 11x11. CNN's are typically implemented using a Programmable Logic Device (PLD) such as a Field Programmable Gate Array (FPGA) or using an Application Specific Integrated Circuit (ASIC) having specialized circuitry to achieve fast processing time. However, the number of memory blocks required to frame a matrix from the input image and the number of multipliers is directly proportional to the size of the matrix. Also, the number of registers required to hold the convolution weights matrix values is directly proportional to the size of the matrix. The large number of memory blocks, multipliers, and registers requires significant power and adds expense, making implementation of larger CNN's (CNN's having a size greater than 3x3) difficult to implement on small size FPGA's and ASIC's.

[003] For example, a system for computing a convolution of a 11x11 matrix can require 11 memory blocks, 121 registers for convolution weights matrix, 121 registers for weights values and 121 parallel multipliers.

[004] Accordingly, what is needed in the art is a method and apparatus that allows for implementing large CNN's that provides lower cost and lower power than current systems.

SUMMARY OF THE INVENTION

[005] A method for performing NxN convolutions on an image file that includes lines of image values is disclosed that includes receiving a weights matrix for an NxN convolution and dividing the weights matrix into a plurality of 3x3 weights matrixes. Configuration files for a 3x3 CNN are loaded into an Integrated Circuit (IC) device to form a 3x3 CNN core in a neural network engine of the IC device. One or more lines of image values are read from the image file and are stored in a buffer of the integrated circuit device as a set of image values. A 3x3 convolution is performed on the set of image values stored in the buffer by: coupling weights values from one of the 3x3 weights matrixes into a neural network engine of the integrated circuit device; loading image values from the buffer into the neural network engine; performing a 3x3 CNN operation of the CNN core, utilizing the loaded image values and coupled weights values from one of the 3x3 weights matrixes as input to the 3x3 CNN operation to generate a 3x3 convolution value. All 3x3 convolution values that are associated with a particular NxN convolution and a particular set of image values are summed together to obtain a sum of 3x3 values. The performing 3x3 convolutions and the summing are repeated until all columns in the set of image values have been processed; and the reading, the storing one or more lines of image values in a buffer, the performing 3x3 convolutions, the summing and the repeating the performing 3x3 convolutions are repeated until all lines of image values in the image file have been processed. Each of the sums associated with a particular NxN convolution is added to the other stored sums associated with the particular NxN convolution to generate an NxN convolution value for each of the NxN convolutions.

[006] An IC device is disclosed that includes a write circuit and a read circuit to read one or more lines of image values from an image file and to perform a read to obtain a weights matrix for a N by N convolution neural network. An NxN CNN circuit of the IC device includes a neural network engine, a controller coupled to the neural network engine, the read circuit and the write circuit, a weight matrix framer coupled to the read circuit and to the neural network engine, an activation matrix framer coupled to the read circuit and to the neural network engine and an accumulator circuit coupled to the neural network engine and to the write circuit. The controller is to divide the weights matrix into a plurality of 3x3 weights matrixes and to load configuration files for a 3x3 CNN into the IC device to form a 3x3 CNN core in the neural network engine of the IC device. The weight matrix framer is to load one of the 3x3 weights matrixes into the

neural network engine. The activation matrix framer includes buffers to store the one or more lines of image values as sets of image values and is to load image values from each of the sets of image values into the neural network engine. Upon receiving the one of the 3x3 weights matrixes and the image values the neural network engine is to perform a 3x3 CNN operation of the 3x3 CNN core to generate a 3x3 convolution value. The accumulator circuit is to sum together all 3x3 convolution values that are associated with a particular NxN convolution and a particular set of image values and is to store each sum of 3x3 convolution values. The IC device is further to add each of the stored sums associated with a particular NxN convolution to the other stored sums associated with the particular NxN convolution to generate an NxN convolution value for each of the NxN convolutions.

[007] A PLD is disclosed that includes a write circuit and a read circuit. The read circuit to read one or more lines of image values from an image file and to perform a read to obtain a weights matrix for a N by N convolution neural network. An NxN CNN circuit included in the PLD includes a neural network engine, a controller coupled to the neural network engine, the read circuit and the write circuit, a weight matrix framer coupled to the read circuit and to the neural network engine, an activation matrix framer coupled to the read circuit and to the neural network engine and an accumulator circuit coupled to the neural network engine and to the write circuit. The controller is to divide the weights matrix into a plurality of 3x3 weights matrixes and to load configuration files for a 3x3 CNN into the PLD to form a 3x3 CNN core in the neural network engine of the PLD. The weight matrix framer is to load one of the 3x3 weights matrixes into the neural network engine. The activation matrix framer includes buffers is to store the one or more lines of image values as sets of image values and is to load image values from each of the sets of image values into the neural network engine. Upon receiving the one of the 3x3 weights matrixes and the image values, the neural network engine is to perform a 3x3 CNN operation of the 3x3 CNN core to generate a 3x3 convolution value. The accumulator circuit is to sum together all 3x3 convolution values that are associated with a particular NxN convolution and a particular set of image values and is to store each sum of 3x3 convolution values. The PLD is further to add each of the stored sums associated with a particular NxN convolution to the other stored sums associated with the particular NxN convolution to generate an NxN convolution value for each of the NxN convolutions.

[008] The method and apparatus of the present invention allows for implementing large CNN's and provides lower cost and lower power than conventional systems in which the number of memory blocks required to frame a matrix from the input image and the number of multipliers is directly proportional to the size of the matrix.

BRIEF DESCRIPTION OF THE DRAWINGS

[009] The accompanying drawings, which are incorporated in and form a part of this specification, illustrate various embodiments and, together with the Description of Embodiments, serve to explain principles discussed below. The drawings referred to in this brief description should not be understood as being drawn to scale unless specifically noted.

[0010] FIG. 1 is a system that includes an IC device and a data storage device that is coupled to the IC device.

[0011] FIG. 2 is a block diagram illustrating a method for performing NxN convolutions.

[0012] FIG. 3A-3F illustrate an example in which a 9x9 convolution operation is performed on 12 rows and 12 columns of image values with a stride of 1.

[0013] FIG. 4A-4B are flow diagrams illustrating use of a neural network engine to add the row vectors from the example shown in FIGS 3A-3F.

DETAILED DESCRIPTION OF THE INVENTION

[0014] Figure 1 shows a system 15 that includes an IC device 1 in communication with a data storage device 11 that is a discreet data storage device such as a Dynamic Random Access Memory (DRAM) that is coupled to IC device 1 (e.g., by traces on a circuit board on which both data storage device 11 and IC device 1 are mounted). IC device 1 may be a PLD such as an FPGA, or an ASIC. IC device 1 includes a read circuit 2 to read data from data storage device 11, a write circuit 3 to write data into data storage device 11, an NxN CNN circuit 21, Input and Output (I/O) circuit 28, other logic circuits 9 and optionally includes local memory 29. Read circuit 2 and write circuit 3 are coupled to data storage device 11.

[0015] NxN CNN circuit 21 includes a weight matrix framer 4, a neural network engine 5, an activation matrix framer 6, a controller 7, an accumulator circuit 8 and an optional adder circuit 19. Controller 7 is coupled to read circuit 2, write circuit 3, weight matrix framer 4, neural network engine 5, activation matrix framer 6 and accumulator circuit 8, and is optionally coupled

to I/O circuit 28, local memory 29, optional adder circuit 19 and other logic circuits 9. Activation matrix framer 6 is coupled to neural network engine 5 and to read circuit 2. Weight matrix framer 4 is further coupled to read circuit 2 and neural network engine 5. Write circuit 3 is further coupled to accumulator circuit 8 and to data storage device 11. Accumulator circuit 8 is further coupled to neural network engine 5. Optional local memory 29 is further coupled to accumulator circuit 8 to optional other logic circuits 9, to read circuit 2, to write circuit 3 and to optional adder circuit 19. I/O circuit 28 is further coupled to other logic circuits 9.

[0016] FIG. 2 illustrates a method 100 for performing NxN convolutions on an image file that includes lines of image values. The method includes receiving (101) a weights matrix for an NxN convolution neural network. In the present application weights matrixes having a size of N by N are referred to using the style "NxN." Accordingly, a matrix indicated herein as a "NxN" matrix has N rows and N columns.

[0017] In one example, the NxN weights matrix for the NxN convolution is stored in data storage device 11 by other devices that are coupled to data storage device 11 (not shown) and read circuit 2 receives the weights matrix by reading an NxN weights matrix for an NxN convolution stored on data storage device 11. Alternatively, the NxN weights matrix for the NxN convolution is received at I/O circuit 28. In one example, the NxN weights matrix for the NxN convolution is received at I/O circuit 28 from an external source and write circuit 3 stores the received image file in data storage device 11 or in local memory 29. Read circuit 2 obtains the NxN weights matrix for the NxN convolution by reading data storage device 11 or local memory 29.

[0018] The received NxN weights matrix is divided (102) into a plurality of 3x3 weights matrixes. If N is not an integer multiple of 3, then zeros are appended to the NxN weights matrix to make the matrix size an integer multiples of 3. For example, when a 4x4, 5x5 or 6x6 convolution is to be performed, the weights matrix received in step 101 is divided into four 3x3 weights matrixes. For 4x4 and 5x5 convolutions, zero values are added to the 4x4 or 5x5 weights matrixes so that the size of the resultant matrix is the next integer multiple of 3, which is 6x6 in the case of 4x4 or 5x5 matrices. Similarly, when the received matrix is a 7x7, 8x8 or 9x9 the received weights matrix is divided into nine 3x3 weights matrixes. For 7x7 and 8x8 convolutions, zero values added to the 7x7 and 8x8 weights matrixes so that the size of the resultant matrix is the next integer multiple of 3, which is 9x9 in the case of 7x7 or 8x8 matrices.

When a 10x10, 11x11 or 12x12 convolution is to be performed, the weights matrix received in step 101 is divided into sixteen 3x3 weights matrixes. For the 10x10 and 11x11 convolutions, zero values are added to the 10x10 and 11x11 weights matrixes so that the size of the resultant matrix is the next integer multiple of 3 which is 12x12 in the case of 10x10 and 11x11 matrices. In FIG. 1, controller 7 divides the received weights matrix into the plurality of 3x3 weights matrixes, add zero weights as required, and to store the resulting plurality of 3x3 weights matrixes in data storage device 11, in local memory 29 or in weight matrix framer 4.

[0019] Configuration files for a 3x3 CNN are loaded (103) into an IC device to form a 3x3 CNN core in a neural network engine of the IC device. Controller 7 instructs read circuit 2 to read configuration files for the 3x3 CNN from data storage device 11 or local memory 29, and controller 7 loads the configuration files for the 3x3 CNN into neural network engine 5 to form the 3x3 CNN in neural network engine 5.

[0020] In FIG. 1, some of the configuration files for the 3x3 CNN can be loaded into neural network engine 5 and some configuration values for the 3x3 CNN and/or instructions from the configuration files can be loaded into the other components of NxN CNN circuit 21 (e.g., to configure the other components of NxN CNN circuit 21).

[0021] FIGS 3A-3F illustrate an example (Example A) in which the steps of method 100 are used to perform a 9x9 convolution on an image file containing 12 rows and 12 columns of image values using a stride of 1. A weights matrix for a 9x9 convolution neural network is received in step 101 and is divided into 9 3x3 weights matrixes W11, W12, W13, W21, W22, W23, W31, W32 and W33 in step 102, where the 3x3 weight matrixes are illustrated in FIG. 3A with a first number indicating row and a second number indicating column. In this example, the 3x3 weight matrixes W11, W12, W13, W21, W22, W23, W31, W32 and W33 are stored in data storage device 11 or in local memory 29 and read by read circuit 2 and coupled to weight matrix framer 4 as required for the operation of weight matrix framer 4. Configuration files for a 3x3 CNN are loaded in step 103 into IC device 1 to form a 3x3 CNN core in neural network engine 5.

[0022] One or more lines of image values are read (104) from an image file. The term “image file”, as used in the present application, includes all types of files that include values relating to an image, and includes, without limitation, Red Green Blue (RGB) image files, greyscale image files, Cyan Magenta Yellow Black (CMYK) image files, Hue Saturation and Lightness (HSL), Hue Saturation Value (HSV) image files and image files resulting from the performance of one

or more activation function (sometimes referred to as activation image files, activation maps or feature maps), and specifically includes video recognition, recommender systems, natural language processing and brain-computer interface and financial time series image files.

[0023] In FIG. 1 the image file that is to be processed using the NxN convolution is stored in data storage device 11 (or in local memory 29) by a user of system 15. In one example, the image file is stored in data storage device 11 by other devices that are coupled to data storage device 11 (not shown). In another example, the image file is received at I/O circuit 28 from an external source and write circuit 3 stores the received image file in data storage device 11 (or in in local memory 29) prior to performing step 104. In FIG. 1 read circuit 2 reads one or more lines of image values from the image file stored in data storage device 11 (or in local memory 29).

[0024] The one or more lines of image values are stored (105) in a buffer of the IC device as a set of image values. In one example the buffer is a three-line buffer and each set of image values stored in the three-line buffer consists of three lines of the image values and in iterations following the first iteration of step 105 additional lines of image values are stored in the 3-line buffer by overwriting one or more lines of image values previously stored in the three-line buffer.

[0025] In the example of FIGS 3A-3F, in step 104 a first line of the image file 26 is read and stored in line buffer 16 of FIG. 1, a second line of the image file is read and is stored in line buffer 17 and a third line of the image file is read and is stored in line buffer 18 to form a first set of image values 10 in line buffers 16-18, as illustrated in FIG. 3C. In the following discussion of the processing of image values stored in buffers 16-18, the first set of image values 10 (three lines in buffers 16-18) are illustrated as rectangles and the rows and columns of image values that are framed (e.g., by framer and output to neural network engine 5 are shown in bold and referenced with the labels 10a-10l. In FIGS. 3D-3F instances of image values in line buffers 16-18 are illustrated in the same manner.

[0026] 3x3 convolutions are performed (106) on the set of image values stored in the buffer. Each of the 3x3 convolutions includes: coupling weights values from one of the 3x3 weights matrixes into the neural network engine 5 of the IC device 1; loading a plurality of image values from the buffer into the neural network engine; and performing a 3x3 CNN operation of the CNN core to generate a 3x3 convolution value. The input to each 3x3 CNN operation thus

includes the plurality of image values from the sets of image values and weights values from one of the 3x3 weights matrixes.

[0027] In FIG. 1, read circuit 2 reads the weights values of one or more of the 3x3 weights matrixes and couple the weights values to weight matrix framer 4; and weight matrix framer 4 loads the weights values from the 3x3 weights matrixes into neural network engine 5. The weights values of all of the 3x3 weights matrixes can be stored in a local memory of weight matrix framer 4 (e.g., registers within weight matrix framer 4), with weight matrix framer 4 operable to provide the weights values of each respective 3x3 weights matrix to neural network engine 5 as required to perform method 100. Alternatively, all of the 3x3 weights matrixes are stored in data storage device 11 (or local memory 29) and individual 3x3 weights matrixes are read by read circuit 2 and coupled to weight matrix framer 4 as required to perform method 100.

[0028] In FIG. 1 read circuit 2 is operable to read three lines of the image values that are to be processed using the CNN and to couple the read three lines of the image values to activation matrix framer 6. It is appreciated that, typically convolutions are processed on a row-by-row basis beginning with the first row. Accordingly, the first three lines will be the first three rows of the image file. However, alternatively, the convolution could be processed column-by-column, beginning with the first column. In the following discussion it will be assumed that processing will begin with the first row and proceed row-by-row.

[0029] Activation matrix framer 6 includes buffers store the set of three lines of image values, and activation matrix framer 6 loads image values from the set of image values into neural network engine 5. More particularly, activation matrix framer 6 includes a first line buffer 16 for storing a first line of image values, a second line buffer 17 for storing a second line of image values and a third line buffer 18 for storing a third line of image values. Line buffers 16-18 can be Static Random-Access Memory (SRAM) blocks that can be individually accessed by read circuit 2.

[0030] Activation matrix framer 6 couples three columns of the image values stored in line buffers 16-18 to neural network engine 5. More particularly, before each 3x3 CNN operation nine image values are loaded into the neural network engine 5, three from each of the line buffers 16-18. More particularly, activation matrix framer 6 is configured for loading three image values from the first line buffer, three image values from the second line buffer and three image values from the third line buffer into the neural network engine prior to each 3x3 CNN operation.

[0031] The input to each 3x3 CNN operation thus includes image values from one of the sets of image values and weights values from one of the 3x3 weights matrixes provided from weight matrix framer 4.

[0032] In FIG. 1, upon receiving the one of the 3x3 weights matrixes provided from weight matrix framer 4 and the image values from activation matrix framer 6 at neural network engine 5, neural network engine 5 performs a 3x3 CNN operation of the 3x3 CNN core to generate a convolution value.

[0033] In one example performing the 3x3 CNN operation further includes multiplying each of the image values received at the neural network engine 5 with a corresponding one of the weights values received at the neural network engine 5 to obtain a 3x3 CNN product. In one example the configuration files indicate one or more bias values (optionally received in step 103) and how the one or more bias values are to be added to one or more of the 3x3 CNN products to generate the 3x3 convolution value. In FIG. 1, one or more bias value is stored in data storage device 11 (or local memory 29) that is read by read circuit 2, and controller 7 is operable to load the one or more bias value into neural network engine 5. In one example, a set of bias values are added to the result of convolution. It is appreciated that NxN convolution does not require the use of bias values, or may require only a single bias value for each NxN convolution to be performed. In such instances a bias value is only provided for one of the 3x3 neural network operations on a particular NxN convolution.

[0034] In FIG. 1, neural network engine 5 includes: a plurality of parallel multipliers 12; a plurality of adders 13; and a plurality of registers 14 that are operably coupled together. Registers 14 include registers for receiving the image values from activation matrix framer 6, registers for receiving the weights values from weight matrix framer 4, and registers for intermediate products and/or sums. Each of parallel multipliers 12 is coupled to one of the registers 14 for receiving an image value and to one of the registers 14 for receiving a weights value and the output of each of the parallel multipliers 12 (a 3x3 CNN product) is coupled to a respective one of the adders 13. Neural network engine 5 can include nine parallel multipliers 12 that multiply respective image values with weight matrix values and couple the product to adders 13. Adders 13 add the outputs from multipliers 12 together to obtain a single 3x3 convolution value.

[0035] Neural network operations are only performed on certain columns of the input image values for each 3x3 convolution of each 9x9 convolution 48a-48p during the processing of each

set of image values. FIG. 3A shows examples of the rows and columns of image file 26 that the 3x3 weights matrixes W11, W12, W13, W21, W22, W23, W31, W32 and W33 operate on to perform each 9x9 convolution 48a-48p. For example in FIG. 3A, the weights matrix $W11_4$ does not operate on the last 6 columns of the image values.

[0036] In the example of FIGS 3A-3F, in step 106 3x3 weights matrix W11 and rows/columns of image values 10a (the first three columns of the first three rows) in set of image values 10 are loaded in step 106 into the neural network engine 5 and a first 3x3 neural network operation is performed using as input to the neural network operation the image values in the first three rows and columns in line buffers 16-18 and 3x3 weights matrix W11 to generate 3x3 convolution value $W11_1$ as illustrated in FIG. 3C.

[0037] All 3x3 convolution values that are associated with a particular NxN convolution, and a particular set of image values are summed (107) and each sum of 3x3 convolution values is optionally stored (108). In FIG. 1, accumulator circuit 8 receives the convolution values from neural network engine 5 and sum together all 3x3 convolution values that are associated with a particular convolution and a particular set of image values. The arrays are stored (108) by coupling the sums of convolution values (e.g., as an array) to write circuit 3 that stores the sums of convolution values (e.g., the arrays) in data storage device 11. Alternatively, write circuit 3 stores the arrays in local memory 29.

[0038] The output of neural network operations is illustrated below as being combined to form an "array." This is not intended to be limiting, but rather is a convenient way to illustrate a series of values that are associated with each other, whether those values are stored in the same storage location, sequentially, as comma separated values, or in non-sequential storage locations. The representation of the output as an "array" containing a single line of numerical values is used for simplicity of illustration, and the results of the neural network operations can also be represented as a corresponding matrix, or simply as individual numerical values.

[0039] Steps 106-108 are repeated (109) until 3x3 CNN operations have been performed on all columns of the set of image files.

[0040] In the example of FIGS 3A-3F, the process continues to the next column and a 3x3 convolution is performed by loading rows/columns of image values 10b (the second through fourth columns of the first three rows of image 26 in line buffers 16-18 – associated with convolution 48b) and 3x3 weights matrix W11 into neural network engine 5 and performing a

CNN operation to generate 3x3 convolution values $W11_2$. A third convolution operation is performed by loading rows/columns of image values 10c (the third through fifth columns of the first three rows of image 26 in line buffers 16-18 - associated with convolution 48c) and weights matrix W11 into neural network engine 5 and performing a 3x3 CNN operation to generate 3x3 convolution value $W11_3$. A fourth convolution operation is performed by loading rows/columns of image values 10d (the fourth through sixth columns of the first three rows of image 26 in line buffers 16-18 - associated with convolution 48d) and 3x3 weights matrix W11 into neural network engine 5 and performing a 3x3 CNN operation to generate 3x3 convolution values $W11_4$. The output from the 3x3 convolution operations utilizing 3x3 weights matrix W11 may be represented as array 31. Subsequent columns do not operate on 3x3 weights matrix W11 as illustrated by the positioning of weights matrixes W11 in 9x9 convolutions 48a-d of FIG. 3A, so the process moves to the next weights matrix W12. 3x3 weights matrix W12 and rows/columns of image values 10e, associated with convolution 48a, are loaded into neural network engine 5 and a CNN operation is performed to generate 3x3 convolution value $W12_1$. 3x3 weights matrix W12 and rows/columns of image values 10f, associated with convolution 48b, are loaded into neural network engine 5 and a 3x3 CNN operation to generate 3x3 convolution value $W12_2$. 3x3 weights matrix W12 and rows/columns of image values 10g, associated with convolution 48c, are loaded into neural network engine 5 and a 3x3 CNN operation is performed using 3x3 weights matrix W12 to generate convolution value $W12_3$. 3x3 weights matrix W12 and rows/columns of image values 10h, associated with convolution 48d, are loaded and a 3x3 CNN operation is performed to generate convolution values $W12_4$. The output from the processing 3x3 weights matrix W12 can be represented as array 32. The process continues with the loading of 3x3 weights matrix W13 and rows/columns of image values 10i, - associated with convolution 48a, into neural network engine 5 and a 3x3 CNN operation is performed to generate 3x3 convolution value $W13_1$. 3x3 weights matrix W13 and rows/columns of image values 10j, associated with convolution 48b, are loaded into neural network engine 5 and a 3x3 CNN operation is performed to generate 3x3 convolution value $W13_2$. 3x3 weights matrix W13 and rows/columns of image values 10k, associated with convolution 48c, are loaded into neural network engine 5 and a 3x3 CNN operation is performed to generate 3x3 convolution values $W13_3$. 3x3 weights matrix W13 and rows/columns of image values 10l, associated with convolution 48d, are loaded into neural network engine 5 and a 3x3 CNN operation is performed

to generate convolution value $W13_4$. The output from the 3x3 convolutions performed using weights matrix W13 can be represented as array 33.

[0041] In the example of FIGS 3A-3F, at_step 107, all 3x3 convolution values that are associated with a particular convolution and set of image values 10 are summed together by accumulator circuit 8 as they are output. More particularly, convolution values from neural network operations on 3x3 weights matrix W11, W12 and W13, for each convolution, are summed by accumulator circuit 8.The sum of neural network operations on the set of image values 10 is represented by array 71 that includes the sum values R1-1 (the sum of the convolution values associated with set of image values 10 and the convolution 48a), R1-2 (the sum of the convolution values associated with set of image values 10 and the convolution 48b), R1-3 (the sum of the convolution values associated with set of image values 10 and the convolution 48c) and R1-4 (the sum of the convolution values associated with set of image values 10 and the convolution 48d). Array 71 is stored in step 108, in which: R1-1 = $W11_1+W12_1+W13_1$; R1-2 = $W11_2+W12_2+W13_2$; R1-3 = $W11_3+W12_3+W13_3$; and R1-4 = $W11_4+W12_4+W13_4$. At this point CNN operations have been performed on all columns of the first 3 rows. In this example NxN convolutions 48a-48p are performed by performing individual 3x3 convolutions, with each calculation of a 3x3 convolution indicated using a subscript corresponding to the particular NxN convolution. For example, referring now to FIG. 3A, for first NxN convolution 48a, each of the corresponding 3x3 weights matrixes have a subscript of 1, indicting it is a first 3x3 calculation of that particular 3x3 weights matrix, and weights matrixes 48b, that correspond to a second NxN convolution have a subscript of 2, indicating a second 3x3 CNN calculation of the particular matrix. The sums from performing steps 106-107 on the first set of image values are illustrated as R1-1 for NxN convolution 48a, R1-2 for NxN convolution 48b, R1-3 for NxN convolution 48c and R 1-4 for NxN convolution 48d.

[0042] Steps 104-109 are repeated (110) until all of the lines of image values in the image file have been processed. In one example a stride for the NxN convolution is received in step 101, and the repeating the reading one or more lines of image values of step 104 includes reading a number of additional lines corresponding to the stride.

[0043] In the example shown in FIGS 3A-3F, in the second iteration of steps 104-109, the process moves to the next row. Operations on the next row are illustrated by convolutions 48e-48h shown in FIGS. 3A-3B. The fourth row of image values in image file 26 is read in step 104

and loaded into one of line buffers 16-18, while the first row of image values is discarded from one of line buffers 16-18 to form a set of image values 20 in line buffers 16-18 shown in FIG. 3D. 3x3 weights matrix W11 is loaded into neural network engine 5. With 3x3 weights matrix W11 loaded: the 3x3 CNN operation on rows/columns of image values 20a, associated with convolution 48e, of step 107 generates 3x3 convolution value $W11_5$; the 3x3 CNN operation on rows/columns of image values 20b, associated with convolution 48f, generates 3x3 convolution value $W11_6$; the 3x3 CNN operation on rows/columns of image values 20c, associated with convolution 48g, generates 3x3 convolution value $W11_7$; and the 3x3 CNN operation on rows/columns of image values 20d, associated with convolution 48h, generates 3x3 convolution value $W11_8$. The output from the processing of 3x3 weights matrix W11 can be represented as array 34.

[0044] The process continues with the loading of 3x3 weights matrix W12 into neural network engine 5. With 3x3 weights matrix W12 loaded: the 3x3 CNN operation on rows/columns of image values 20e, associated with convolution 48e, in step 106 generates convolution value $W12_5$; the 3x3 CNN operation on rows/columns of image values 20f, associated with convolution 48f, generates convolution value $W12_6$; the 3x3 CNN operation on rows/columns of image values 20g, associated with convolution 48g, generates convolution value $W12_7$; and the CNN operation on rows/columns of image values 20h, associated with convolution 48h, generates convolution values $W12_8$. The output from the processing 3x3 weights matrix W12 can be represented as array 35. In step 108 3x3 convolution values from 3x3 convolution operations on 3x3 weights matrixes W11 and W12 are summed by accumulator circuit 8.

[0045] The process continues with 3x3 convolutions using weights matrix W13. With 3x3 weights matrix W13 loaded into neural network engine 5: in step 106 the 3x3 CNN operation on rows/columns of image values 20i, associated with convolution 48e, generates convolution value $W13_5$; the 3x3 CNN operation on rows/columns of image values 20j, associated with convolution 48f, generates convolution value $W13_6$; the CNN operation on rows/columns of image values 20k, associated with convolution 48g, generates convolution value $W13_7$; and the CNN operation on rows/columns of image values 20l, associated with convolution 48h, generates 3x3 convolution value $W13_8$. The output from the processing 3x3 weights matrix W13 can be represented as array 36. The sum of neural network operations on the set of image values 20 (107) is represented by array 72 that includes the sum values R2-1 (the sum of the

convolution values associated with set of image values 20 and the convolution 48e), R2-2 (the sum of the convolution values associated with set of image values 20 and the convolution 48f), R2-3 (the sum of the convolution values associated with set of image values 20 and the convolution 48g) and R2-4 (the sum of the convolution values associated with set of image values 20 and the convolution 48h). Array 72 is stored in step 108, in which: R2-1= $W11_5$ +$W12_5$ + $W13_5$; R2-2= $W11_6$ +$W12_6$ + $W13_6$; R2-3= $W11_7$ +$W12_7$ + $W13_7$; and R2-4= $W11_8$+$W12_8$+ $W13_8$.

[0046] In the next iteration of steps 104-109 the process moves to the third row, illustrated by the first three rows of convolutions 48i-48l in FIG. 3A, with the fifth row of image values loaded into one of the line buffers 16-18, while the second line of image values in image 26 is discarded to form a set of image values 30 in line buffers 16-18. As shown in FIG. 3E, 3x3 convolution operations are performed on set of image values 30 using 3x3 weights matrix W11 on rows/columns of image values 30a, 30b, 30c and 30d to generate 3x3 convolution values $W11_9$, $W11_{10}$, $W11_{11}$, $W11_{12}$, that may be represented as array 37; performed using 3x3 weights matrix W12 on rows/columns of image values 30e, 30f, 30g and 30h to generate 3x3 convolution values $W12_9$, $W12_{10}$, $W12_{11}$, $W12_{12}$, that may be represented as array 38 and performed using 3x3 weights matrix W13 on rows/columns of image values 30i, 30j, 30k and 30l to generate 3x3 convolution value $W13_9$, $W13_{10}$, $W13_{11}$, $W13_{12}$ that may be represented as array 39. The outputs from neural network operations on rows 3-5 are summed together (107) to generate stored array 73 (that includes sums R3-1, R3-2, R3-3 and R3-4) in which: R3-1= $W11_9$ +$W12_9$ + $W13_9$; R3-2= $W11_{10}$ +$W12_{10}$ + $W13_{10}$; R3-3= $W11_{11}$ +$W12_{11}$ + $W13_{11}$; R3-4= $W11_{12}$+$W12_{12}$+ $W13_{12}$.

[0047] In the next iteration of steps 104-109 the process moves to the fourth row, illustrated by the first three rows of 9x9 convolutions 48m-48p in FIG. 3A, with the sixth row of image values from image file 26 loaded into one of line buffers 16-18, while the third line of image values 26 is discarded to form a set of image values 40 in buffers 16-18. Referring now to FIG. 3F, 3x3 CNN operations are performed on set of image values 40 using weights matrix W11 on rows/columns of image values 40a, 40b, 40c and 40d to generate output $W11_{13}$, $W11_{14}$, $W11_{15}$, $W11_{16}$, that may be represented as array 41; performed using 3x3 weights matrix W12 on rows/columns of image values 40e, 40f, 40g and 40h to generate 3x3 convolution values $W12_{13}$, $W12_{14}$, $W12_{15}$, $W12_{16}$, that may be represented as array 42; and performed using 3x3 weights matrix W13 on rows/columns of image values 40i, 40j, 40k and 40l to generate 3x3 convolution values $W13_{13}$, $W13_{14}$, $W13_{15}$, $W13_{16}$, that may be represented as array 43. The outputs from

neural network operations set of image values 40 for convolutions 48m-p are summed together to generate stored array 74 (that includes R4-1, R4-2, R4-3 and R4-4) that is stored in step 108; where R4-1= $W11_{13}+W12_{13}+W13_{13}$; R4-2= $W11_{14}+W12_{14}+W13_{14}$; R4-3= $W11_{15}+W12_{15}+W13_{15}$; and R4-4= $W11_{16}+W12_{16}+W13_{16}$. (e g stored in local memory 29 or data storage device 11).

[0048] Neural network operations are performed using weights matrix W21 on rows/columns of image values 40a-40d to generate 3x3 convolution values $W21_1$-$W21_4$, that may be represented as array 44; neural network operations performed using weights matrix W22 on rows/columns of image values 40e-40h to generate 3x3 convolution value $W22_1$-$W22_4$, that may be represented as array 45 and neural network operations performed using weights matrix W23 on rows/columns of image values 40i-40l to generate 3x3 convolution values $W23_1$-$W23_4$, that may be represented as array 46. The 3x3 convolution values from 3x3 CNN operations on set of image values 40 and convolutions 48a-d are summed together and stored (108) as array 75 (that includes sums R4-5, R4-6, R4-7 and R4-8), where: R4-5= $W21_1+W22_1+W23_1$; R4-6= $W21_2+W22_2+W23_2$; R4-7= $W21_3+W22_3+W23_3$; and R4-8= $W21_4+W22_4+W23_4$. Similarly R7-1, R7-2, R7-3 and R7-4 are computed.

[0049] In the example of Figs 3A-3F, steps 104-109 continue to be repeated to process subsequent rows until all rows and columns have been processed. FIG. 3B illustrates how the values generated from the summation of neural network operations in step 108 map to each convolution 48a-48p, with the values R1-1 through R10-4 stored in local memory 29 or data storage device 11.

[0050] Each of the stored sums associated with a particular NxN convolution is added to the other stored sums associated with the particular NxN convolution (111) to generate an NxN convolution value for each of the NxN convolutions.

[0051] In one example the adding each of the sums of step 111 is performed using a dedicated adder circuit. In FIG. 1, optional adder circuit 19 adds the sums.

[0052] In one example neural network operations are performed on the sums associated with a particular convolution to add the sums in step 111. More particularly, a plurality of the sums associated with a particular convolution and a weights matrix that includes weights values having a value of 1 are coupled to the neural network engine 5, and a neural network operation is performed so as to multiply each of the sums associated with a particular convolution with 1 and

add the resulting products together to obtain the NxN convolution value. In this example, IC device 1 does not included dedicated circuitry for adding the sums associated with a particular convolution together (e.g., does not include adder circuit 19), but rather neural network engine 5 is used for performing the adding of step 111. In this example, weight matrix framer 4 couples to neural network engine 5 weights values having a value of "1" and read circuit 2 reads the sums associated with a particular NxN convolution and store them into buffers 16-18 of activation matrix framer 6. Activation matrix framer 6 couples the sums associated with a particular NxN convolution to neural network engine 5, that in turn multiplies each of the sums associated with the particular convolution with 1 and add the resulting products together to obtain the respective NxN convolution value.

[0053] The NxN convolution values are optionally stored (112). The NxN convolution values from step 111 may be in the form of individual values, comma separated values, a vector or a matrix that is stored by write circuit 3 in data storage device 11. Alternatively, accumulator circuit 8 stores the NxN convolution values in local memory 29 so that they can be accessed by other logic circuits 9. The NxN convolution values may also be output via I/O circuit 28.

[0054] When IC device 1 is a PLD, programming of the PLD forms NxN CNN circuit 21 that is coupled to other logic circuits 9 of the PLD. Furthermore, the PLD can be programmed as a single fixed-size NxN CNN having a predetermined size and predetermined weights values. The fact that NxN CNN circuit 21 is not a full NxN CNN may be transparent to the user such that the user is unaware that the NxN CNN circuit 21 processes convolutions using a 3x3 convolutions.

[0055] Steps 101-112 can also be performed after programming the PLD. More particularly, after programming of the PLD, the user may provide a weights matrix in step 101, the image file and optionally the stride. IC device 1 is then operable to perform steps 103-109 and optionally step 111.

[0056] Alternatively, IC device 1 can be a variable-size CNN (e g , a PLD configured to perform any size of convolution), with the size of the CNN indicated along with the stride in step 101 and coupled to controller 7. Controller 7 is then operable to configure the other components of NxN CNN circuit 21 to perform the correct convolution. The term "stride" as used in the present application refers to the number of rows or columns between adjoining convolutions.

[0057] In step 111 optional adder circuit 19 can be used to add the stored sums to generate an NxN convolution value for each of the NxN convolutions for each of convolutions 48a-48p.

However, using a dedicated circuit to perform the addition adds to the number of gates and the complexity of the NxN CNN circuit 21. In one example, neural network engine 5 is used to perform the addition and NxN CNN circuit 21 does not include an adder circuit 19. More particularly, row vectors to be added are input to neural network engine 5 along with a weights matrix having weights values of a first column having a value of "1" and weights values of other columns having a value "0"as illustrated in FIG. 4A-4B. For large matrixes with more than three sums to be added together, the NxN convolution values output from the neural network operations can be fed back into the neural network 5 as many times as necessary to obtain as output a single NxN convolution value for each of the NxN convolutions to be computed.

[0058] In the example illustrated in FIGS. 4A-4B, sums R4-5 through R10-4 are generated in the same manner as the above example relating to sums R1-1 through R 3-4,

[0059] In the example illustrated in FIGS. 4A-4B, for sums R1-1, R4-5 and R7-1 (that are generated in the same manner as in the examples shown above) weights matrix 83 is loaded into neural network engine 5 to generate the 9x9 convolution values for convolution 48a; sums R1-2, R4-6 and R7-2 and weights matrix 83 are loaded into neural network engine 5 to generate the 9x9 convolution values for convolution 48b. Similarly, neural network operations are performed using weights matrix 83 in which: sums R1-3, R4-7 and R7-3 generate the 9x9 convolution values for convolution 48c; sums R1-4, R4-8 and R7-4 generate the 9x9 convolution values for convolution 48d; sums R2-1, R5-1 and R8-1 generate the 9x9 convolution values for convolution 48e; sums R2-2, R5-2 and R8-2 generate the 9x9 convolution values for convolution 48f; sums R2-3, R5-3 and R8-3 generate the 9x9 convolution values for convolution 48g; and sums R2-4, R5-4 and R8-4 generate the 9x9 convolution values for convolution 48h.

[0060] Referring now to FIG. 4B, neural network operations are performed using weights matrix 83 in which: sums R3-1, R6-1 and R9-1 generate the 9x9 convolution values for convolution 48i; sums R3-2, R6-2 and R9-2 generate the 9x9 convolution values for convolution 48j; sums R3-3, R6-6 and R9-3 generate the 9x9 convolution values for convolution 48k; sums R3-4, R6-4 and R9-4 generate the 9x9 convolution values for convolution 48l; sums R4-1, R7-5 and R10-1 generate the 9x9 convolution values for convolution 48m; and sums R4-2, R7-6 and R10-2 generate the 9x9 convolution values for convolution 48n; sums R4-3, R7-7 and R10-3 generate the 9x9 convolution values for convolution 48o; and sums R4-4, R7-8 and R10-4 generate the 9x9 convolution values for convolution 48p.

[0061] The methods and apparatus of the present invention reduce the number of memory blocks, registers for convolution weights and multipliers required for performing NxN convolutions of large matrixes. For example, in embodiments of the present invention an 11x11 convolution can be computed using only 3 memory blocks (one for each line buffer 16-18), 9 registers for convolution weights matrix, 9 registers for weights values, 9 parallel multipliers. Accordingly, the method and apparatus disclosed herein allows for running large NxN convolutions at lower cost and lower power than current systems that require N parallel multipliers, NxN registers for convolution weights values and a corresponding number of adders.

CLAIMS

What is claimed is:

1. A method for performing an NxN convolution on an image file that includes lines of image values, where N is greater than 3, the method comprising:

receiving a weights matrix for an NxN convolution neural network (CNN);

dividing the weights matrix into a plurality of 3x3 weights matrixes;

loading configuration files for a 3x3 Convolutional Neural Network (CNN) into an integrated circuit device to form a 3x3 CNN core in a neural network engine of the integrated circuit device;

storing one or more lines of image values in a buffer of the integrated circuit device as a set of image values;

performing a 3x3 convolution on the set of image values stored in the buffer by:

coupling weights values from one of the 3x3 weights matrixes into a neural network engine of the integrated circuit device;

loading the set of image values from the buffer into the neural network engine;

performing a 3x3 CNN operation of the CNN core, utilizing the loaded set of image values and coupled weights values from one of the 3x3 weights matrixes as input to the 3x3 CNN operation to generate a 3x3 convolution value;

summing together all 3x3 convolution values that are associated with a particular NxN convolution and a particular set of image values to obtain a sum of 3x3 values;

repeating the performing 3x3 convolutions and the summing until all columns in the set of image values have been processed;

repeating the reading, the storing one or more lines of image values in a buffer, the performing 3x3 convolutions, the summing and the repeating the performing 3x3 convolutions until all lines of image values in the image file have been processed; and

adding each of the sums associated with a particular NxN convolution to the other stored sums associated with the particular NxN convolution to generate an NxN convolution value for each of the NxN convolutions.

2. The method of claim 1 further comprising storing each sum of 3x3 convolution values and storing each of the NxN convolution values.

3. The method of claim 1 wherein the buffer is a three-line buffer and each of the sets of image values consists of three lines of the image values.

4. The method of claim 3 wherein the storing the lines of image values in the buffer after a first set of image files have been stored comprises overwriting one or more lines of image values previously stored in the three-line buffer.

5. The method of claim 1 wherein the configuration files indicate one or more bias values and how the one or more bias values are to be added to a 3x3 CNN product to generate the 3x3 convolution value.

6. The method of claim 1 comprising:

receiving a stride for the NxN convolution, and

wherein the repeating the reading one or more lines of image values comprises reading a number of additional lines corresponding to the stride.

7. The method of claim 1 wherein the adding each of the sums associated with the particular NxN convolution to the other stored sums associated with the particular NxN convolution comprises adding each of the sums using a dedicated adder circuit.

8. The method of claim 1 wherein the adding each of the sums associated with the particular NxN convolution to the other stored sums associated with the particular NxN convolution comprises performing neural network operations on the sums associated with the particular NxN convolution.

9. The method of claim 1 wherein the adding each of the sums associated with the particular NxN convolution to the other stored sums associated with the particular NxN convolution comprises:

coupling a plurality of the sums associated with the particular convolution and a weights matrix that includes weights values of a first column having a value of “1” and weights values of other columns having a value “0”to the neural network engine; and

performing a neural network operation so as to multiply each of the sums associated with the particular convolution with 1 and add the resulting products together to generate the NxN convolution value for each of the NxN convolutions.

10. An integrated circuit device comprising:

a read circuit to read one or more lines of image values from an image file and to perform a read to obtain a weights matrix for an N by N convolution neural network (CNN), where N is greater than 3;

an NxN CNN circuit comprising:

a neural network engine;

a controller coupled to the neural network engine and the read circuit, the controller to divide the weights matrix into a plurality of 3x3 weights matrixes, load configuration files for a 3x3 CNN into the integrated circuit device to form a 3x3 CNN core in the neural network engine;

a weight matrix framer coupled to the read circuit and to the neural network engine, the weight matrix framer to load one of the 3x3 weights matrixes into the neural network engine;

an activation matrix framer coupled to the read circuit and to the neural network engine, the activation matrix framer including buffers to store the one or more lines of image values as sets of image values, the activation matrix framer to load image values from each of the sets of image values into the neural network engine;

wherein, upon receiving the one of the 3x3 weights matrixes and the image values, the neural network engine is to perform a 3x3 CNN operation of the 3x3 CNN core to generate a 3x3 convolution value; and

an accumulator circuit coupled to the neural network engine, the accumulator circuit to sum together all 3x3 convolution values that are associated with a particular NxN convolution and associated with a particular set of image values and to store each sum of 3x3 convolution values, and

wherein the integrated circuit device is further to add each of the stored sums associated with a particular NxN convolution to the other stored sums associated with the particular NxN convolution to generate an NxN convolution value for each of the NxN convolutions.

11. The integrated circuit device of claim 10, wherein the buffer includes a first line buffer to store a first line of image values from the image file, a second line buffer to store a second line of image values from the image file and a third line buffer to store a third line of image values from the image file, the activation matrix framer for loading three image values from the first line

buffer, three image values from the second line buffer and three image values from the third line buffer into the neural network engine prior to each of the 3x3 CNN operations.

12. The integrated circuit device of claim 10 further comprising an adder circuit coupled to the accumulator circuit for adding each of the sums associated with a particular NxN convolution to the other sums associated with the particular NxN convolution to generate the NxN convolution value for each of the NxN convolutions.

13. The integrated circuit device of claim 10 wherein the neural network engine is to add each of the sums associated with a particular NxN convolution to the other sums associated with the particular NxN convolution by performing neural network operations in which some of the weights values of a first column having a value of "1" and weights values of other columns having a value "0".

14. The integrated circuit device of claim 10 wherein the neural network engine comprises:

a plurality of parallel multipliers;

a plurality of adders; and

a plurality of registers including registers for receiving the image values and registers for receiving the weights values,

wherein each of the parallel multipliers is coupled to one of the registers for receiving the image values and one of the registers for receiving one of the weights values, and the output of each of the parallel multipliers is coupled to one of the adders.

15. The integrated circuit device of claim 10 comprising a write circuit to write each sum of 3x3 convolution values to a data storage device or to a local memory so as to store each sum of 3x3 convolution values in the data storage device or the local memory.

16. The integrated circuit device of claim 10 wherein the read circuit is to read the one or more lines of image values from an image file stored on a data storage device and is to perform a read of the data storage device to obtain the weights matrix for an N by N convolution neural network (CNN).

17. The integrated circuit device of claim 10 wherein the read circuit is to read the one or more lines of image values from an image file stored in local memory and is to perform a read of local memory to obtain the weights matrix for an N by N convolution neural network (CNN).

18. A programmable logic device comprising:

a read circuit to read one or more lines of image values from an image file and to perform a read to obtain a weights matrix for a N by N convolution neural network (CNN), where N is greater than 3;

a write circuit ;

an NxN CNN circuit comprising:

a neural network engine;

a controller coupled to the neural network engine, the read circuit and the write circuit, the controller to divide the weights matrix into a plurality of 3x3 weights matrixes, load configuration files for a 3x3 CNN into the integrated circuit device to form a 3x3 CNN core in the neural network engine of the integrated circuit device;

a weight matrix framer coupled to the read circuit and to the neural network engine, the weight matrix framer to load one of the 3x3 weights matrixes into the neural network engine;

an activation matrix framer coupled to the read circuit and to the neural network engine, the activation matrix framer including buffers to store the one or more lines of image values as sets of image values, the activation matrix framer to load image values from each of the sets of image values into the neural network engine;

wherein, upon receiving the one of the 3x3 weights matrixes and the image values, the neural network engine is to perform a 3x3 CNN operation of the 3x3 CNN core to generate a 3x3 convolution value; and

an accumulator circuit coupled to the neural network engine and coupled to the write circuit, the accumulator circuit to sum together all 3x3 convolution values that are associated with a particular NxN convolution and associated with a particular set of image values and store each sum of 3x3 convolution values, and

wherein the programmable logic device is further to add each of the stored sums associated with a particular NxN convolution to the other stored sums associated with the particular NxN convolution to generate an NxN convolution value for each of the NxN convolutions.

19. The programmable logic device of claim 18 wherein the neural network engine is to add each of the sums associated with the particular NxN convolution to the other sums associated with the

particular convolution to generate the NxN convolution value for each of the NxN convolutions by performing neural network operations in which the weights values are set to a value of 1.

20. The programmable logic device of claim 18 further comprising:

other configurable logic circuits external to the NxN CNN circuit;

local memory external to the NxN CNN circuit, the local memory coupled to the other configurable logic circuits and the NxN CNN circuit,

an input and output (I/O) circuit coupled to the controller and to the other configurable logic circuits, wherein the I/O circuit is to receive the NxN convolution values for each of the NxN convolutions and the write circuit is to store the NxN convolution values for each of the NxN convolutions in a data storage device or in local memory.

21. The programmable logic device of claim 20 wherein the read circuit is to read the one or more lines of image values from an image file stored on a data storage device and is to perform a read of the data storage device to obtain the weights matrix for an N by N CNN.

22. The programmable logic device of claim 20 wherein the read circuit is to read the one or more lines of image values from an image file stored in local memory and is to perform a read of local memory to obtain the weights matrix for an N by N CNN.

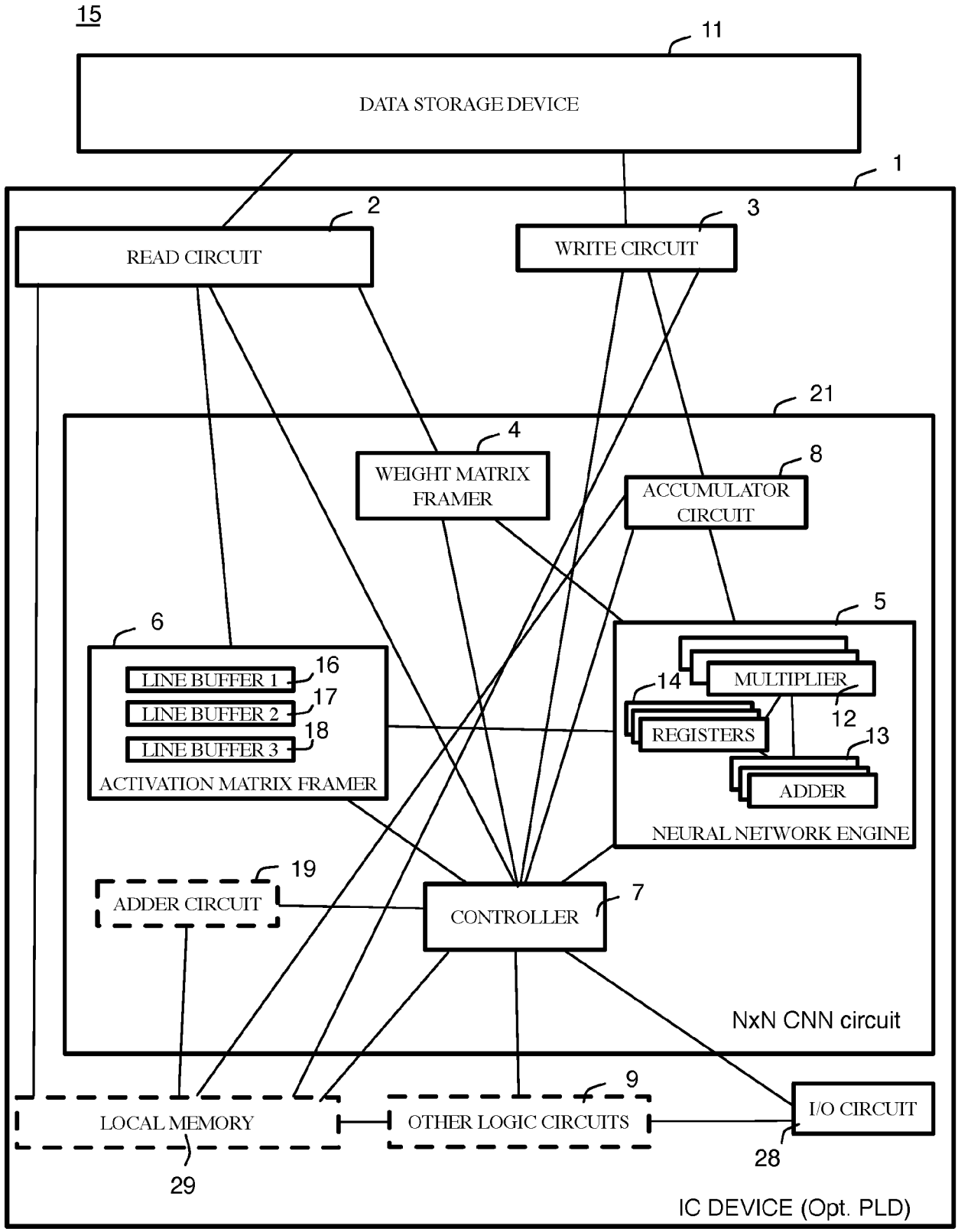
15
11
DATA STORAGE DEVICE
1
2
READ CIRCUIT
3
WRITE CIRCUIT
21
4
WEIGHT MATRIX FRAMER
8
ACCUMULATOR CIRCUIT
5
6
LINE BUFFER 1
16
LINE BUFFER 2
17
LINE BUFFER 3
18
ACTIVATION MATRIX FRAMER
14
MULTIPLIER
12
REGISTERS
13
ADDER
NEURAL NETWORK ENGINE
19
ADDER CIRCUIT
CONTROLLER
7
NxN CNN circuit
9
LOCAL MEMORY
OTHER LOGIC CIRCUITS
I/O CIRCUIT
28
29
IC DEVICE (Opt. PLD)

FIG. 1

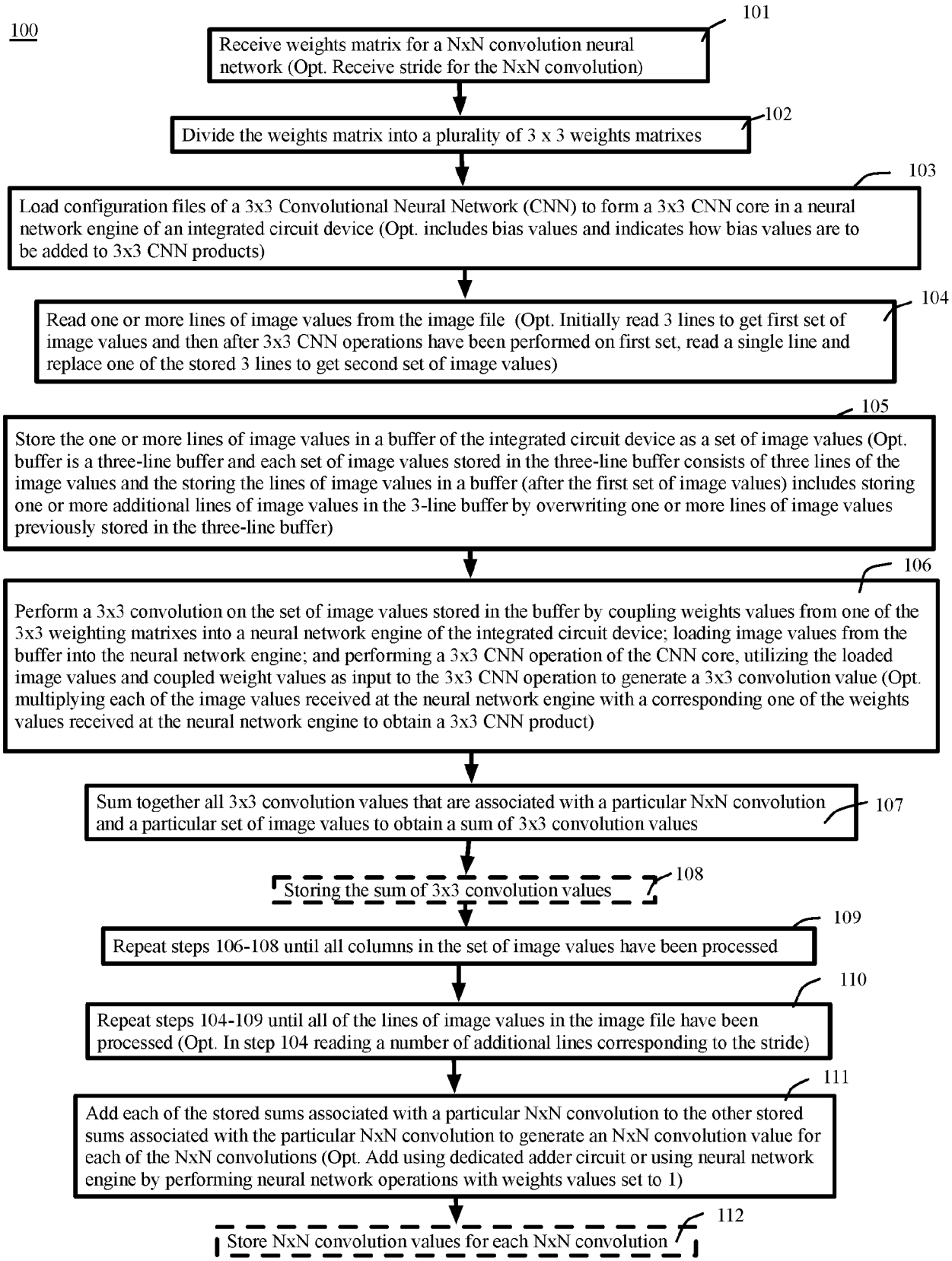
100
101
Receive weights matrix for a NxN convolution neural network (Opt. Receive stride for the NxN convolution)
102
Divide the weights matrix into a plurality of 3 x 3 weights matrixes
103
Load configuration files of a 3x3 Convolutional Neural Network (CNN) to form a 3x3 CNN core in a neural network engine of an integrated circuit device (Opt. includes bias values and indicates how bias values are to be added to 3x3 CNN products)
104
Read one or more lines of image values from the image file (Opt. Initially read 3 lines to get first set of image values and then after 3x3 CNN operations have been performed on first set, read a single line and replace one of the stored 3 lines to get second set of image values)
105
Store the one or more lines of image values in a buffer of the integrated circuit device as a set of image values (Opt. buffer is a three-line buffer and each set of image values stored in the three-line buffer consists of three lines of the image values and the storing the lines of image values in a buffer (after the first set of image values) includes storing one or more additional lines of image values in the 3-line buffer by overwriting one or more lines of image values previously stored in the three-line buffer)
106
Perform a 3x3 convolution on the set of image values stored in the buffer by coupling weights values from one of the 3x3 weighting matrixes into a neural network engine of the integrated circuit device; loading image values from the buffer into the neural network engine; and performing a 3x3 CNN operation of the CNN core, utilizing the loaded image values and coupled weight values as input to the 3x3 CNN operation to generate a 3x3 convolution value (Opt. multiplying each of the image values received at the neural network engine with a corresponding one of the weights values received at the neural network engine to obtain a 3x3 CNN product)
107
Sum together all 3x3 convolution values that are associated with a particular NxN convolution and a particular set of image values to obtain a sum of 3x3 convolution values
108
Storing the sum of 3x3 convolution values
109
Repeat steps 106-108 until all columns in the set of image values have been processed
110
Repeat steps 104-109 until all of the lines of image values in the image file have been processed (Opt. In step 104 reading a number of additional lines corresponding to the stride)
111
Add each of the stored sums associated with a particular NxN convolution to the other stored sums associated with the particular NxN convolution to generate an NxN convolution value for each of the NxN convolutions (Opt. Add using dedicated adder circuit or using neural network engine by performing neural network operations with weights values set to 1)
112
Store NxN convolution values for each NxN convolution

FIG. 2

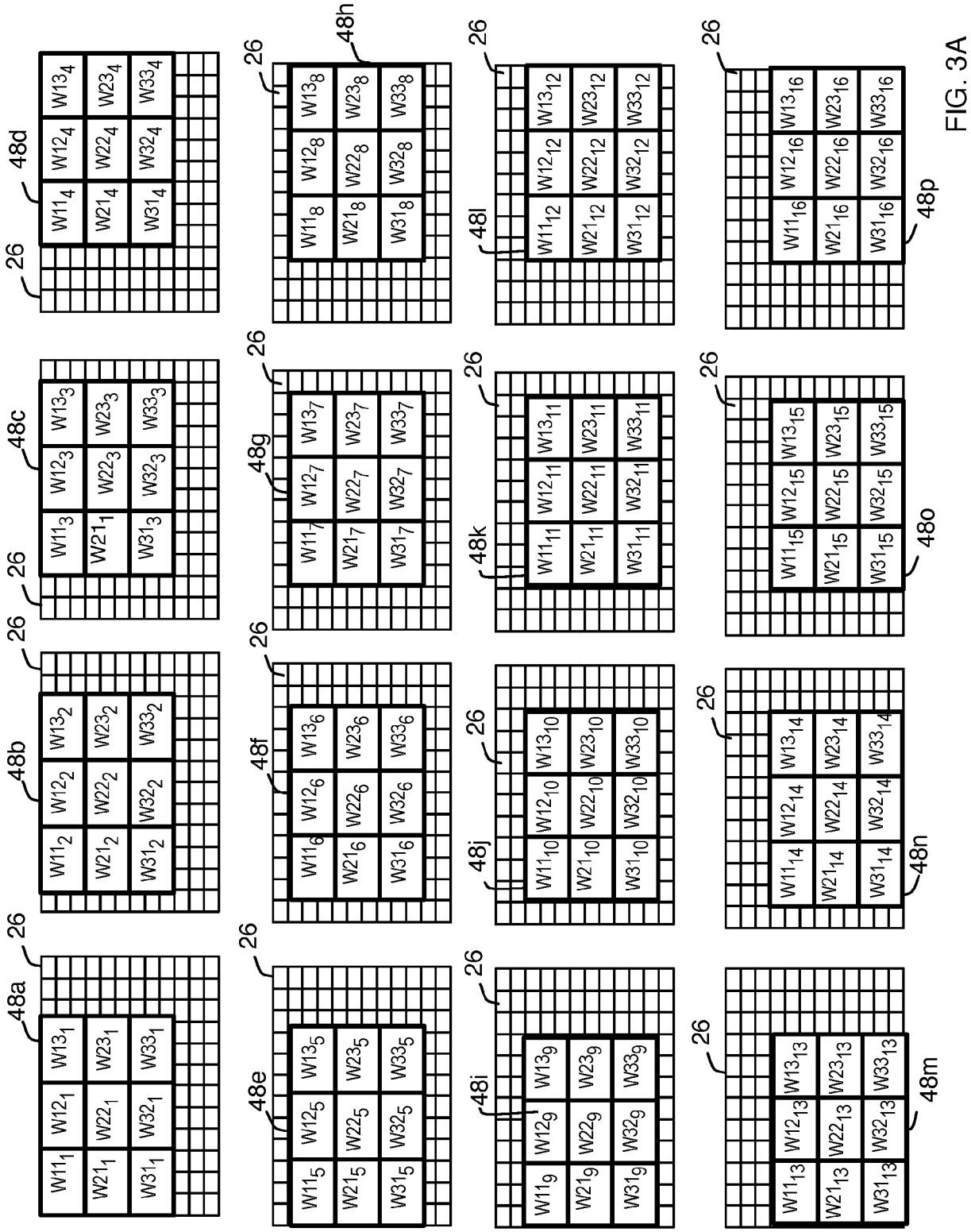
48a
48b
48c
48d
48e
48f
48g
48h
48i
48j
48k
48l
48m
48n
48o
48p
26

FIG. 3A

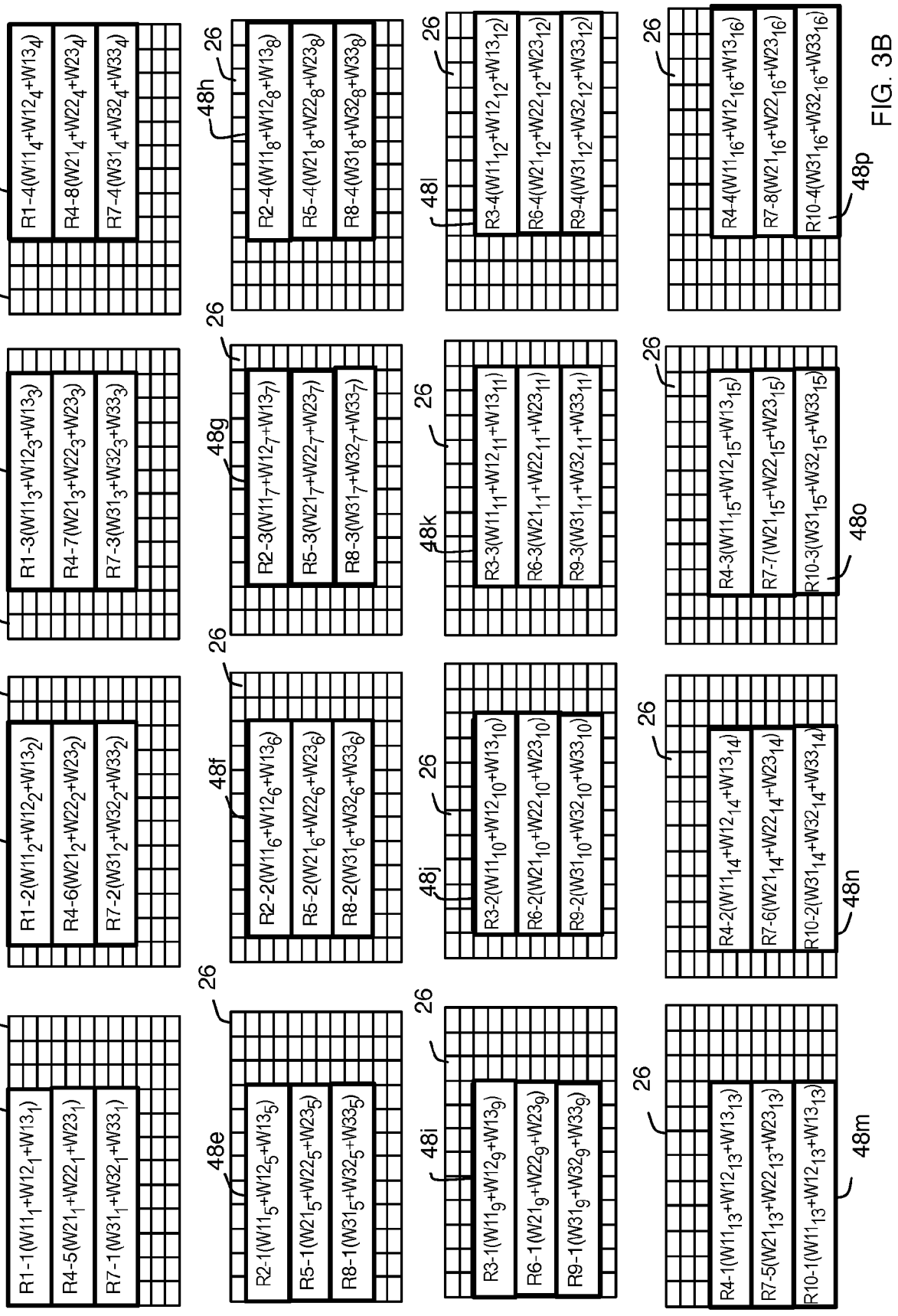
48a
R1-1(W11$_1$+W12$_1$+W13$_1$)
R4-5(W21$_1$+W22$_1$+W23$_1$)
R7-1(W31$_1$+W32$_1$+W33$_1$)
48b
R1-2(W11$_2$+W12$_2$+W13$_2$)
R4-6(W21$_2$+W22$_2$+W23$_2$)
R7-2(W31$_2$+W32$_2$+W33$_2$)
48c
R1-3(W11$_3$+W12$_3$+W13$_3$)
R4-7(W21$_3$+W22$_3$+W23$_3$)
R7-3(W31$_3$+W32$_3$+W33$_3$)
48d
R1-4(W11$_4$+W12$_4$+W13$_4$)
R4-8(W21$_4$+W22$_4$+W23$_4$)
R7-4(W31$_4$+W32$_4$+W33$_4$)
48e
R2-1(W11$_5$+W12$_5$+W13$_5$)
R5-1(W21$_5$+W22$_5$+W23$_5$)
R8-1(W31$_5$+W32$_5$+W33$_5$)
48f
R2-2(W11$_6$+W12$_6$+W13$_6$)
R5-2(W21$_6$+W22$_6$+W23$_6$)
R8-2(W31$_6$+W32$_6$+W33$_6$)
48g
R2-3(W11$_7$+W12$_7$+W13$_7$)
R5-3(W21$_7$+W22$_7$+W23$_7$)
R8-3(W31$_7$+W32$_7$+W33$_7$)
48h
R2-4(W11$_8$+W12$_8$+W13$_8$)
R5-4(W21$_8$+W22$_8$+W23$_8$)
R8-4(W31$_8$+W32$_8$+W33$_8$)
48i
R3-1(W11$_9$+W12$_9$+W13$_9$)
R6-1(W21$_9$+W22$_9$+W23$_9$)
R9-1(W31$_9$+W32$_9$+W33$_9$)
48j
R3-2(W11$_{10}$+W12$_{10}$+W13$_{10}$)
R6-2(W21$_{10}$+W22$_{10}$+W23$_{10}$)
R9-2(W31$_{10}$+W32$_{10}$+W33$_{10}$)
48k
R3-3(W11$_{11}$+W12$_{11}$+W13$_{11}$)
R6-3(W21$_{11}$+W22$_{11}$+W23$_{11}$)
R9-3(W31$_{11}$+W32$_{11}$+W33$_{11}$)
48l
R3-4(W11$_{12}$+W12$_{12}$+W13$_{12}$)
R6-4(W21$_{12}$+W22$_{12}$+W23$_{12}$)
R9-4(W31$_{12}$+W32$_{12}$+W33$_{12}$)
48m
R4-1(W11$_{13}$+W12$_{13}$+W13$_{13}$)
R7-5(W21$_{13}$+W22$_{13}$+W23$_{13}$)
R10-1(W11$_{13}$+W12$_{13}$+W13$_{13}$)
48n
R4-2(W11$_{14}$+W12$_{14}$+W13$_{14}$)
R7-6(W21$_{14}$+W22$_{14}$+W23$_{14}$)
R10-2(W31$_{14}$+W32$_{14}$+W33$_{14}$)
48o
R4-3(W11$_{15}$+W12$_{15}$+W13$_{15}$)
R7-7(W21$_{15}$+W22$_{15}$+W23$_{15}$)
R10-3(W31$_{15}$+W32$_{15}$+W33$_{15}$)
48p
R4-4(W11$_{16}$+W12$_{16}$+W13$_{16}$)
R7-8(W21$_{16}$+W22$_{16}$+W23$_{16}$)
R10-4(W31$_{16}$+W32$_{16}$+W33$_{16}$)
26

FIG. 3B

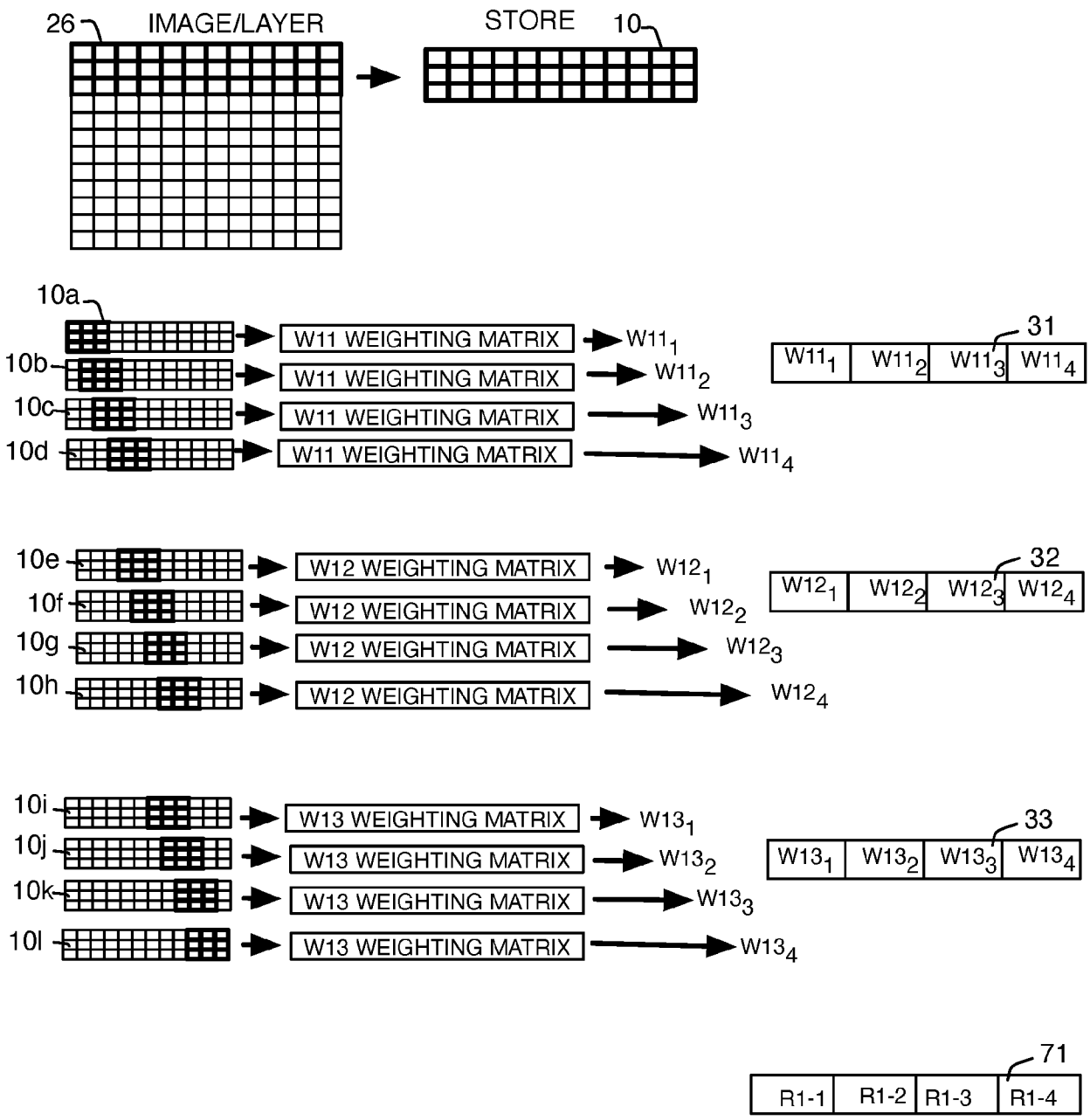
26
IMAGE/LAYER
STORE
10
10a
10b
10c
10d
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11$_1$
W11$_2$
W11$_3$
W11$_4$
31
W11$_1$ W11$_2$ W11$_3$ W11$_4$
10e
10f
10g
10h
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12$_1$
W12$_2$
W12$_3$
W12$_4$
32
W12$_1$ W12$_2$ W12$_3$ W12$_4$
10i
10j
10k
10l
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13$_1$
W13$_2$
W13$_3$
W13$_4$
33
W13$_1$ W13$_2$ W13$_3$ W13$_4$
71
R1-1 R1-2 R1-3 R1-4

FIG. 3C

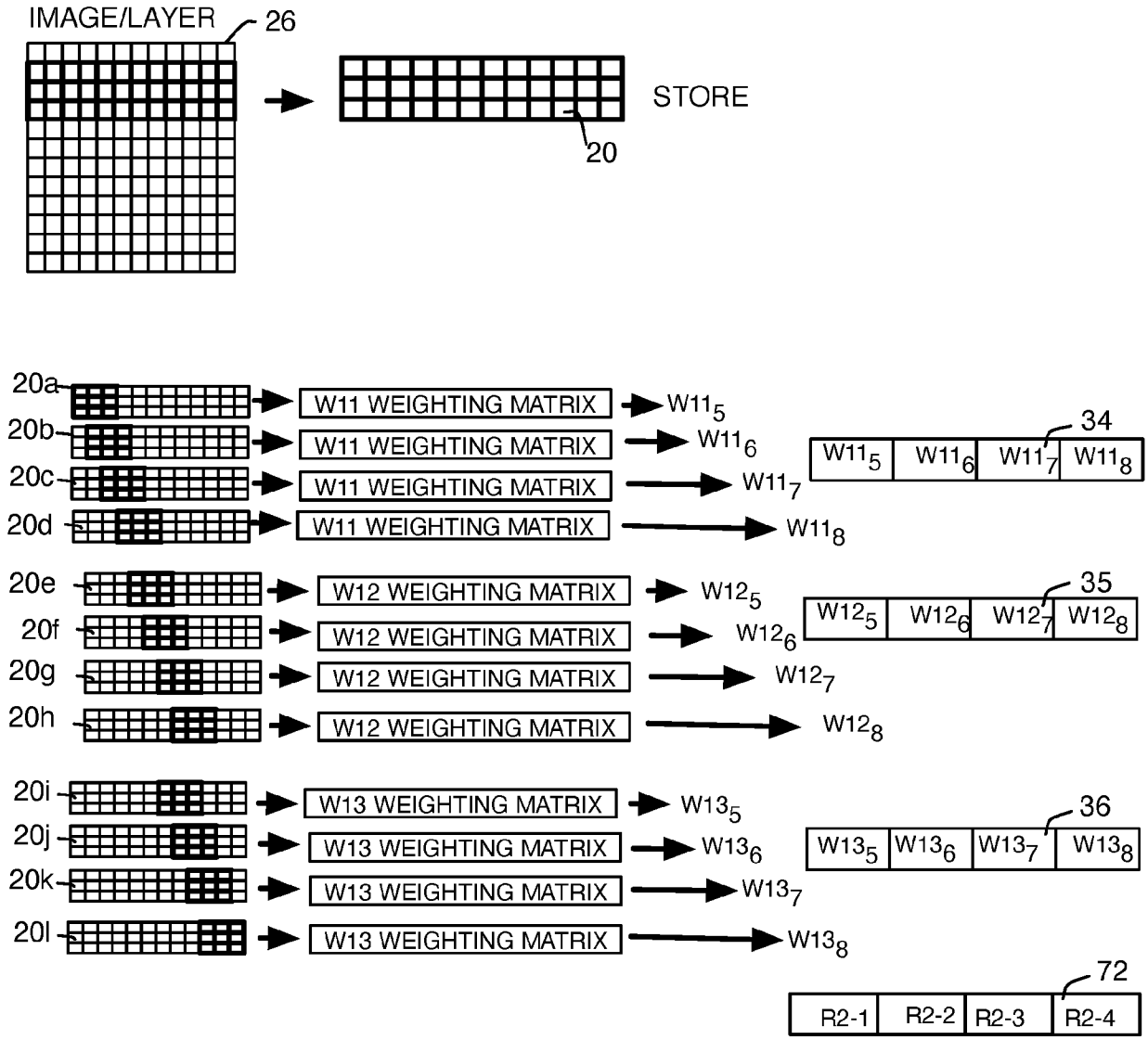
IMAGE/LAYER
26
STORE
20
20a
20b
20c
20d
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11$_5$
W11$_6$
W11$_7$
W11$_8$
34
W11$_5$ W11$_6$ W11$_7$ W11$_8$
20e
20f
20g
20h
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12$_5$
W12$_6$
W12$_7$
W12$_8$
35
W12$_5$ W12$_6$ W12$_7$ W12$_8$
20i
20j
20k
20l
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13$_5$
W13$_6$
W13$_7$
W13$_8$
36
W13$_5$ W13$_6$ W13$_7$ W13$_8$
72
R2-1 R2-2 R2-3 R2-4

FIG. 3D

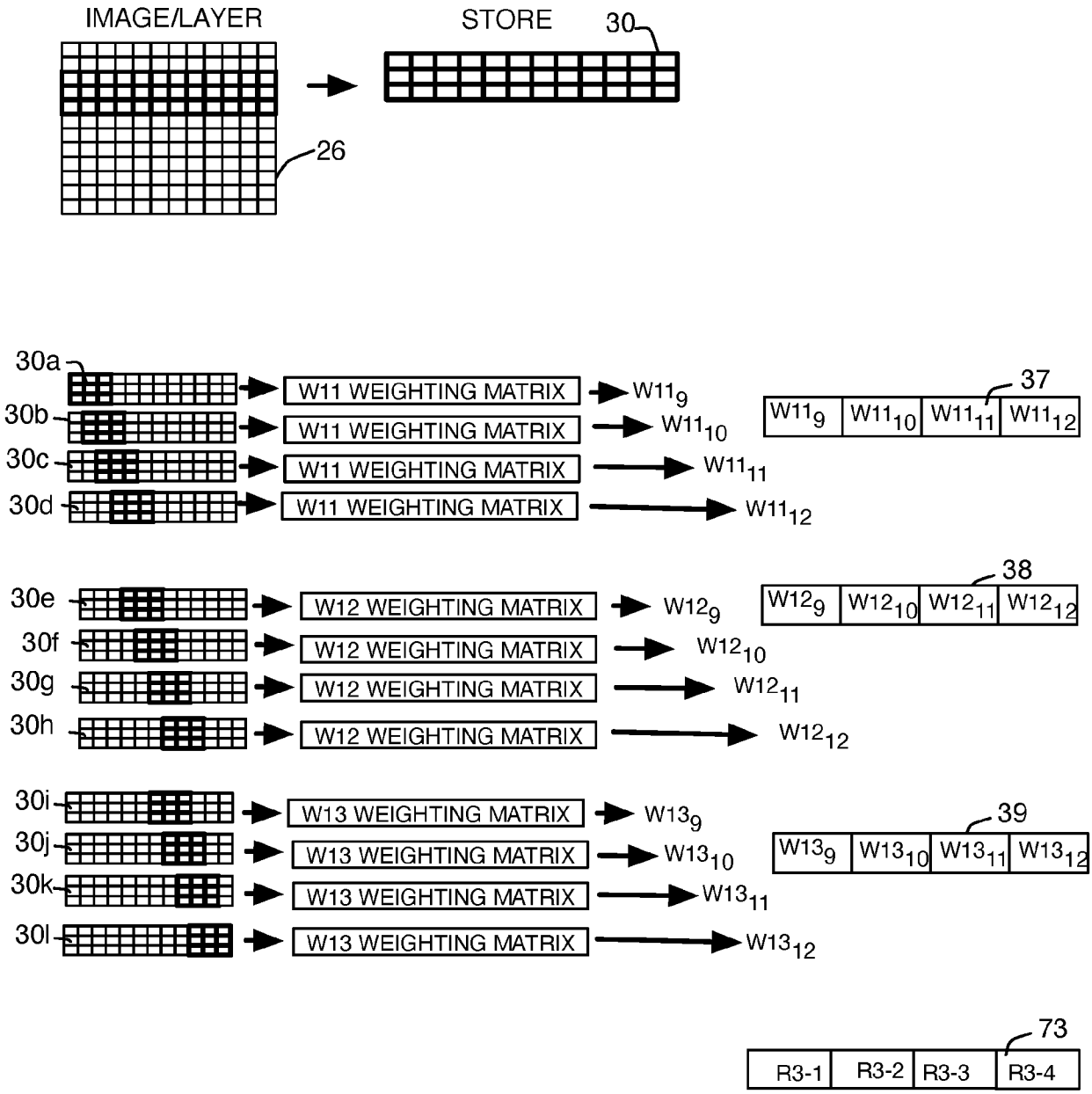
IMAGE/LAYER
STORE
30
26
30a
30b
30c
30d
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11$_9$
W11$_{10}$
W11$_{11}$
W11$_{12}$
37
W11$_9$ W11$_{10}$ W11$_{11}$ W11$_{12}$
30e
30f
30g
30h
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12$_9$
W12$_{10}$
W12$_{11}$
W12$_{12}$
38
W12$_9$ W12$_{10}$ W12$_{11}$ W12$_{12}$
30i
30j
30k
30l
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13$_9$
W13$_{10}$
W13$_{11}$
W13$_{12}$
39
W13$_9$ W13$_{10}$ W13$_{11}$ W13$_{12}$
73
R3-1 R3-2 R3-3 R3-4

FIG. 3E

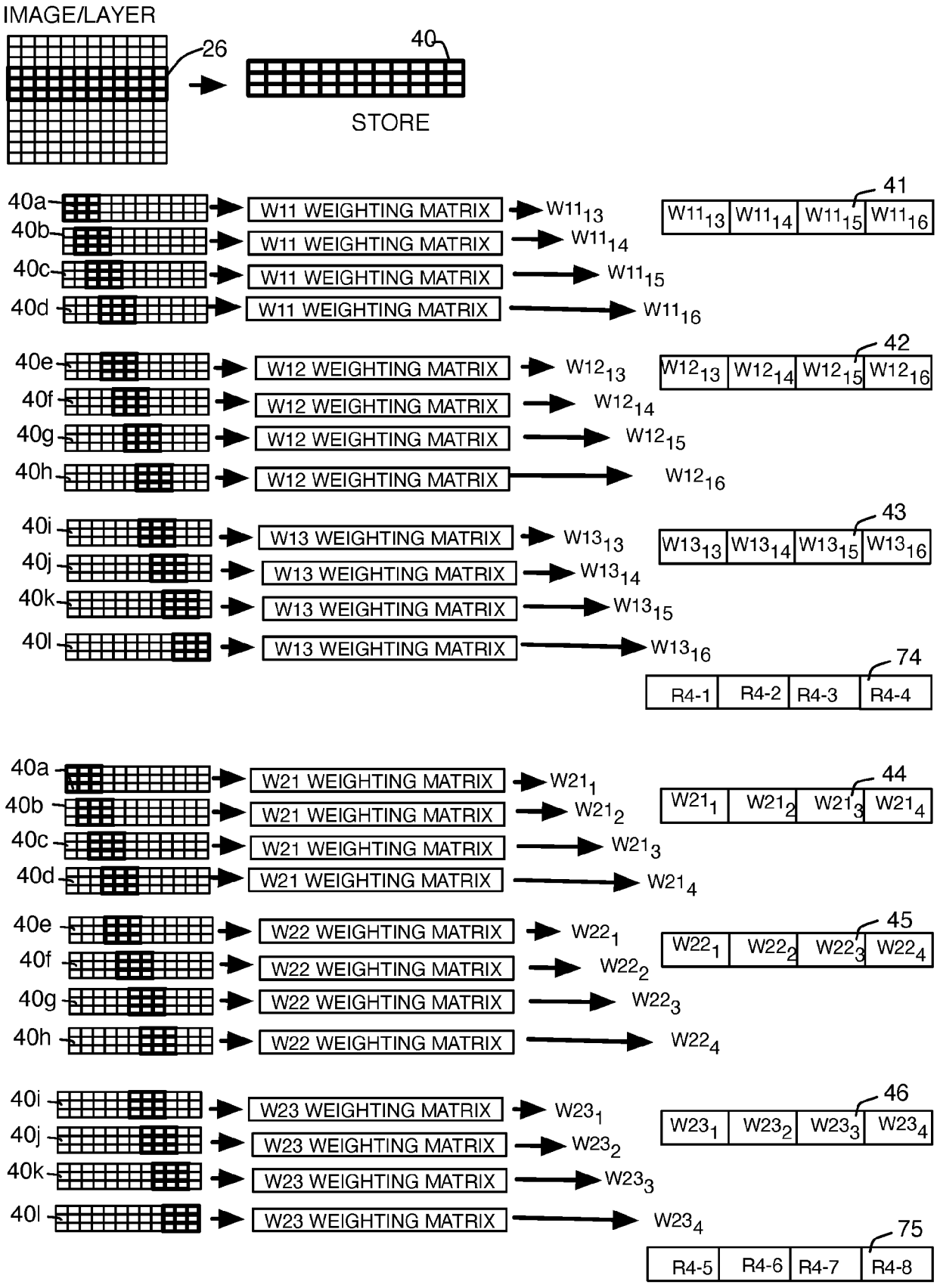
IMAGE/LAYER
26
40
STORE
40a
40b
40c
40d
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11 WEIGHTING MATRIX
W11$_{13}$
W11$_{14}$
W11$_{15}$
W11$_{16}$
41
W11$_{13}$ W11$_{14}$ W11$_{15}$ W11$_{16}$
40e
40f
40g
40h
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12 WEIGHTING MATRIX
W12$_{13}$
W12$_{14}$
W12$_{15}$
W12$_{16}$
42
W12$_{13}$ W12$_{14}$ W12$_{15}$ W12$_{16}$
40i
40j
40k
40l
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13 WEIGHTING MATRIX
W13$_{13}$
W13$_{14}$
W13$_{15}$
W13$_{16}$
43
W13$_{13}$ W13$_{14}$ W13$_{15}$ W13$_{16}$
74
R4-1 R4-2 R4-3 R4-4
40a
40b
40c
40d
W21 WEIGHTING MATRIX
W21 WEIGHTING MATRIX
W21 WEIGHTING MATRIX
W21 WEIGHTING MATRIX
W21$_1$
W21$_2$
W21$_3$
W21$_4$
44
W21$_1$ W21$_2$ W21$_3$ W21$_4$
40e
40f
40g
40h
W22 WEIGHTING MATRIX
W22 WEIGHTING MATRIX
W22 WEIGHTING MATRIX
W22 WEIGHTING MATRIX
W22$_1$
W22$_2$
W22$_3$
W22$_4$
45
W22$_1$ W22$_2$ W22$_3$ W22$_4$
40i
40j
40k
40l
W23 WEIGHTING MATRIX
W23 WEIGHTING MATRIX
W23 WEIGHTING MATRIX
W23 WEIGHTING MATRIX
W23$_1$
W23$_2$
W23$_3$
W23$_4$
46
W23$_1$ W23$_2$ W23$_3$ W23$_4$
75
R4-5 R4-6 R4-7 R4-8

FIG.3F

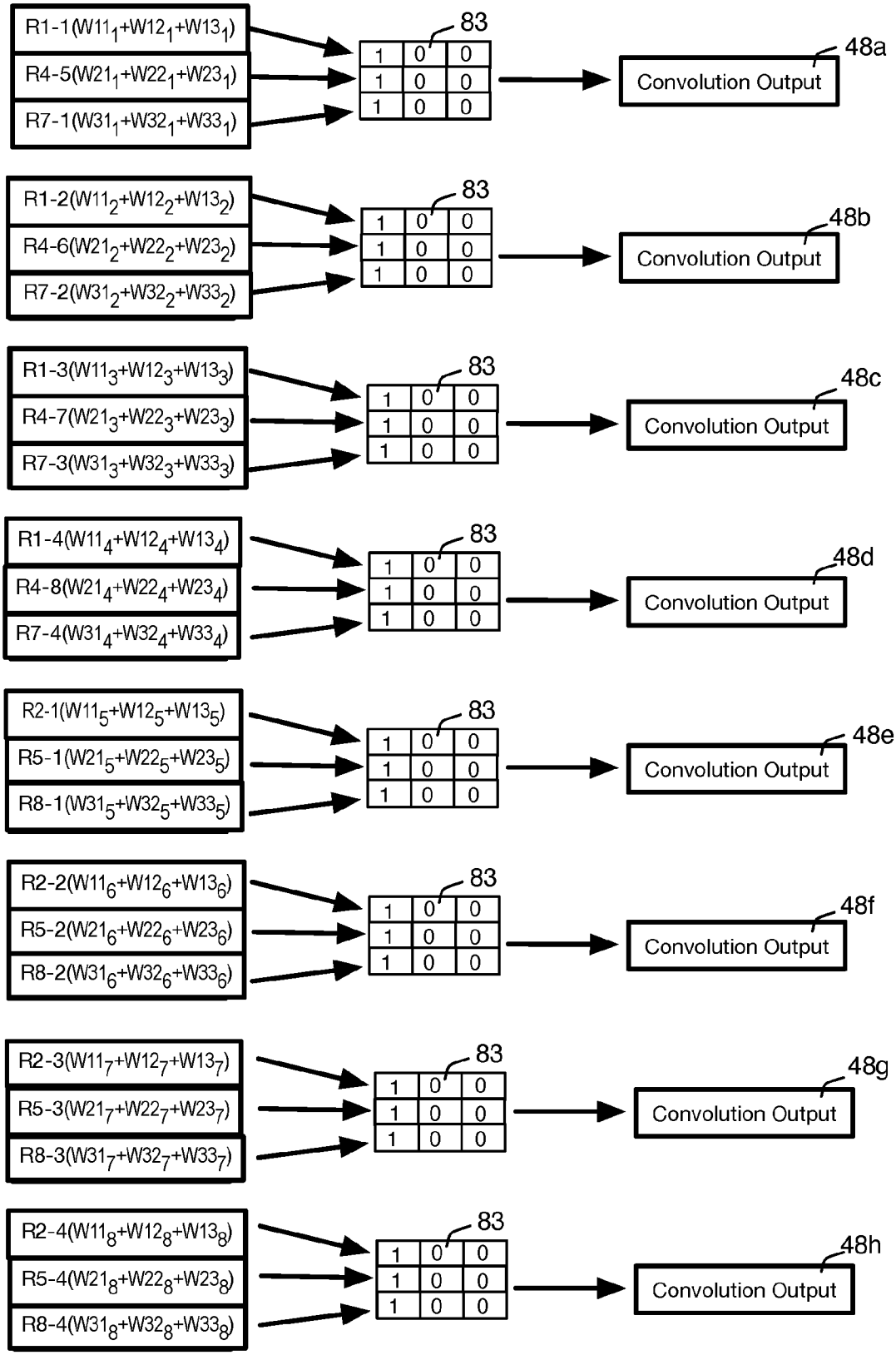
R1-1(W11$_1$+W12$_1$+W13$_1$)
R4-5(W21$_1$+W22$_1$+W23$_1$)
R7-1(W31$_1$+W32$_1$+W33$_1$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48a
R1-2(W11$_2$+W12$_2$+W13$_2$)
R4-6(W21$_2$+W22$_2$+W23$_2$)
R7-2(W31$_2$+W32$_2$+W33$_2$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48b
R1-3(W11$_3$+W12$_3$+W13$_3$)
R4-7(W21$_3$+W22$_3$+W23$_3$)
R7-3(W31$_3$+W32$_3$+W33$_3$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48c
R1-4(W11$_4$+W12$_4$+W13$_4$)
R4-8(W21$_4$+W22$_4$+W23$_4$)
R7-4(W31$_4$+W32$_4$+W33$_4$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48d
R2-1(W11$_5$+W12$_5$+W13$_5$)
R5-1(W21$_5$+W22$_5$+W23$_5$)
R8-1(W31$_5$+W32$_5$+W33$_5$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48e
R2-2(W11$_6$+W12$_6$+W13$_6$)
R5-2(W21$_6$+W22$_6$+W23$_6$)
R8-2(W31$_6$+W32$_6$+W33$_6$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48f
R2-3(W11$_7$+W12$_7$+W13$_7$)
R5-3(W21$_7$+W22$_7$+W23$_7$)
R8-3(W31$_7$+W32$_7$+W33$_7$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48g
R2-4(W11$_8$+W12$_8$+W13$_8$)
R5-4(W21$_8$+W22$_8$+W23$_8$)
R8-4(W31$_8$+W32$_8$+W33$_8$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48h

FIG. 4A

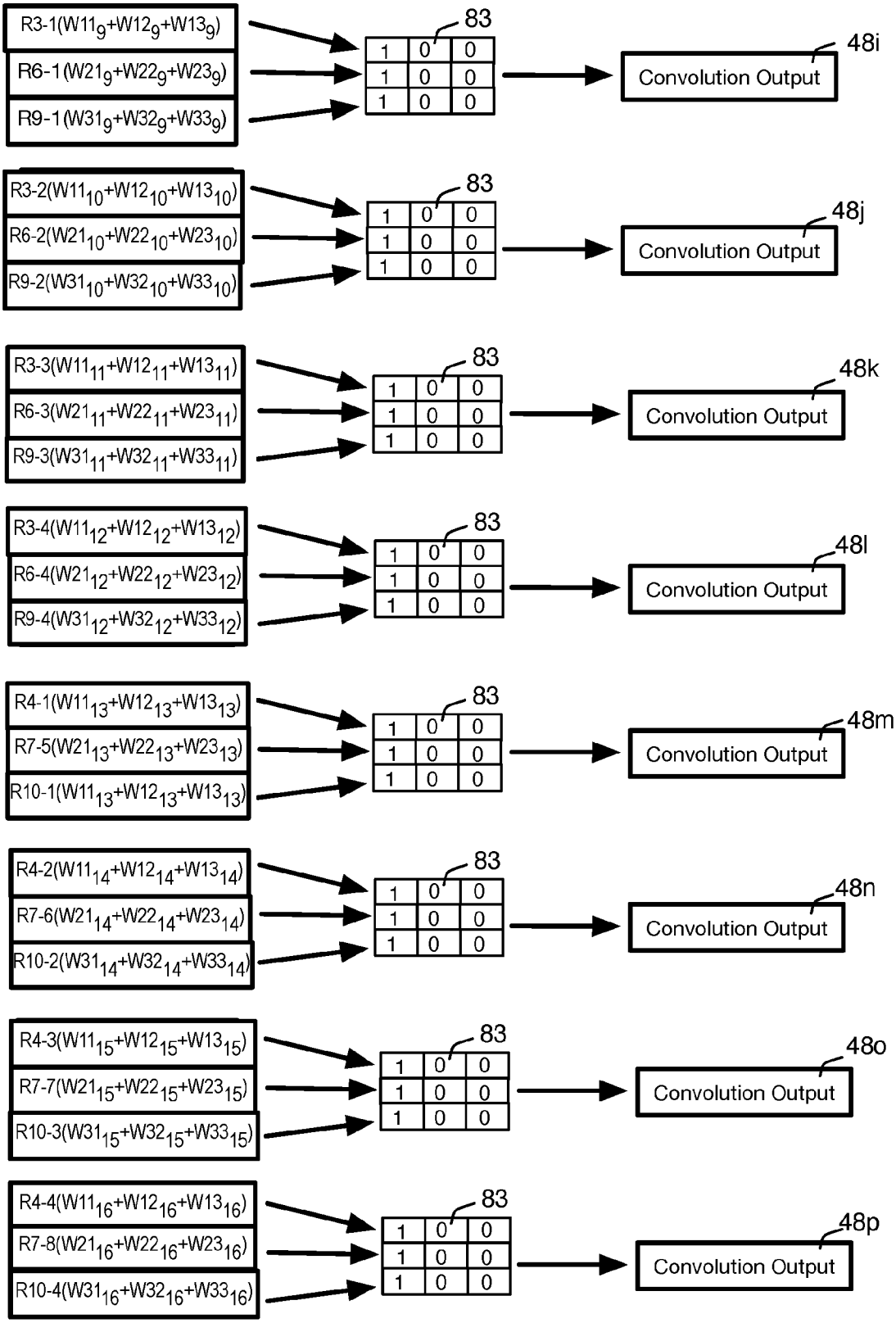
R3-1(W11$_9$+W12$_9$+W13$_9$)
R6-1(W21$_9$+W22$_9$+W23$_9$)
R9-1(W31$_9$+W32$_9$+W33$_9$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48i
R3-2(W11$_{10}$+W12$_{10}$+W13$_{10}$)
R6-2(W21$_{10}$+W22$_{10}$+W23$_{10}$)
R9-2(W31$_{10}$+W32$_{10}$+W33$_{10}$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48j
R3-3(W11$_{11}$+W12$_{11}$+W13$_{11}$)
R6-3(W21$_{11}$+W22$_{11}$+W23$_{11}$)
R9-3(W31$_{11}$+W32$_{11}$+W33$_{11}$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48k
R3-4(W11$_{12}$+W12$_{12}$+W13$_{12}$)
R6-4(W21$_{12}$+W22$_{12}$+W23$_{12}$)
R9-4(W31$_{12}$+W32$_{12}$+W33$_{12}$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48l
R4-1(W11$_{13}$+W12$_{13}$+W13$_{13}$)
R7-5(W21$_{13}$+W22$_{13}$+W23$_{13}$)
R10-1(W11$_{13}$+W12$_{13}$+W13$_{13}$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48m
R4-2(W11$_{14}$+W12$_{14}$+W13$_{14}$)
R7-6(W21$_{14}$+W22$_{14}$+W23$_{14}$)
R10-2(W31$_{14}$+W32$_{14}$+W33$_{14}$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48n
R4-3(W11$_{15}$+W12$_{15}$+W13$_{15}$)
R7-7(W21$_{15}$+W22$_{15}$+W23$_{15}$)
R10-3(W31$_{15}$+W32$_{15}$+W33$_{15}$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48o
R4-4(W11$_{16}$+W12$_{16}$+W13$_{16}$)
R7-8(W21$_{16}$+W22$_{16}$+W23$_{16}$)
R10-4(W31$_{16}$+W32$_{16}$+W33$_{16}$)
83
1 0 0
1 0 0
1 0 0
Convolution Output
48p

FIG.4B

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2021/053281

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/15 G06N3/063
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)
G06F 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	CONTI FRANCESCO ET AL: "A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters", 2015 DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION (DATE), EDAA, 9 March 2015 (2015-03-09), pages 683-688, XP032765876, DOI: 10.7873/DATE.2015.0404 [retrieved on 2015-04-22] page 685 - page 686; figures 2,4	1-22

-/--

[x] 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
17 January 2022	26/01/2022

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	Huguet Serra, G

1

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2021/053281

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	DAVID GSCHWEND: "ZynqNet: An FPGA-Accelerated Embedded Convolutional Neural Network", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 14 May 2020 (2020-05-14), XP081673689, the whole document	1-22
A	SUN BAOHUA ET AL: "Ultra Power-Efficient CNN Domain Specific Accelerator with 9.3TOPS/Watt for Mobile and Embedded Applications", 2018 IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION WORKSHOPS (CVPRW), IEEE, 18 June 2018 (2018-06-18), pages 1758-17588, XP033475521, DOI: 10.1109/CVPRW.2018.00219 [retrieved on 2018-12-13] the whole document	1-22
A	MELONI PAOLO ET AL: "CNN hardware acceleration on a low-power and low-cost APSoC", 2019 CONFERENCE ON DESIGN AND ARCHITECTURES FOR SIGNAL AND IMAGE PROCESSING (DASIP), IEEE, 16 October 2019 (2019-10-16), pages 7-12, XP033750957, DOI: 10.1109/DASIP48288.2019.9049213 [retrieved on 2020-03-26] the whole document	1-22