(12) UK Patent (19) GB (11) 2582088 (13) B

(45) Date of B Publication 27.07.2022

(54) Title of the Invention: Counter based resistive processing unit for programmable and reconfigurable artificial-neural-networks

(51) INT CL: ***G06N 3/08*** (2006.01) ***G06N 3/063*** (2006.01)

(21) Application No: **2007432.4**

(22) Date of Filing: **22.11.2018**

Date Lodged: **19.05.2020**

(30) Priority Data:
(31) **15840322** (32) **13.12.2017** (33) **US**

(86) International Application Data:
PCT/IB2018/059228 En 22.11.2018

(87) International Publication Data:
WO2019/116128 En 20.06.2019

(43) Date of Reproduction by UK Office **09.09.2020**

(56) Documents Cited:
US 9779355 B1
US 9715656 B1
US 9646243 B1
US 4874963 A
US 20170109626 A1
US 20110119215 A1
Seyoung Kim et al. "Analog CMOS-based Resistive Processing Unit for Deep Neural Network Training"2017 IEEE 60TH International Midwest Symposium on Circuits and Systems (MWSCAS), 09 August 2017 (2017-08-09), pages 422-425

(58) Field of Search:
As for published application 2582088 A viz:
INT CL **G06F, G06N, G11C**
Other: **CNABS; CNTXT; CNKI;DWPI; SIPOABS; USTXT; WOTXT; EPTXT; SETXT; GBTXT; CATXT; CHTXT; ATTXT; LETXT; IEEE; DNN**
updated as appropriate

Additional Fields
INT CL **H03K**
Other: **WPI, EPODOC**

(72) Inventor(s):
Yulong Li
Siyuranga O Koswatta
Paul Michael Solomon

(73) Proprietor(s):
International Business Machines Corporation (Incorporated in USA - New York)
New Orchard Road, Armonk, New York 10504, United States of America

(74) Agent and/or Address for Service:
IBM United Kingdom Limited
Intellectual Property Department, Hursley Park, Winchester, Hampshire, SO21 2JN, United Kingdom

GB 2582088 B

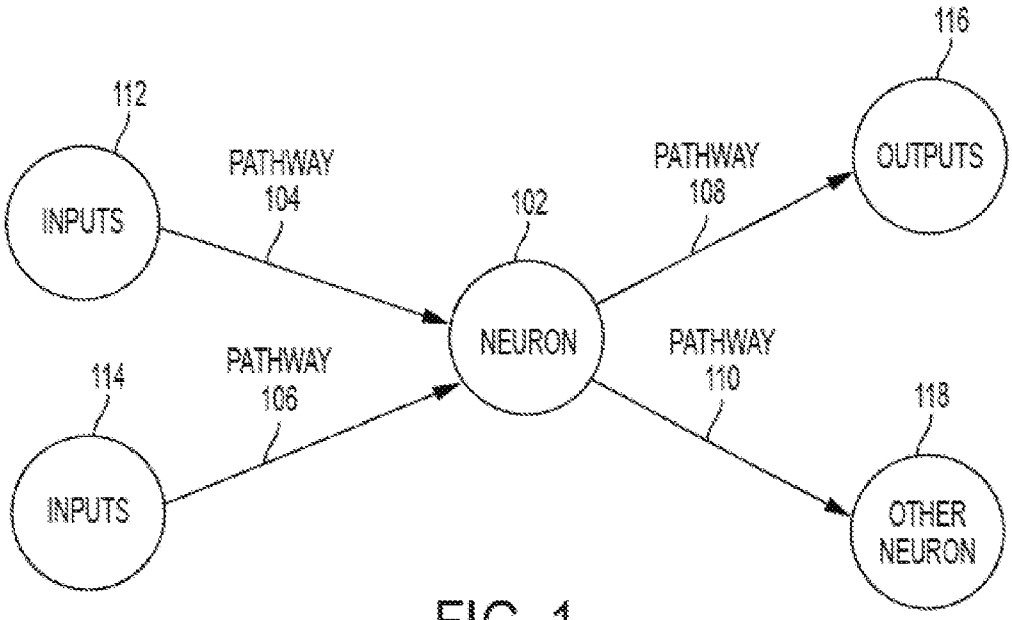
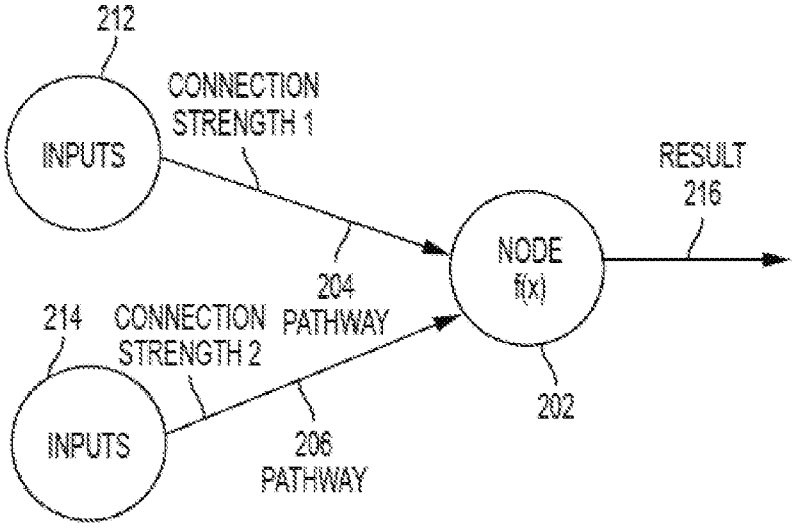
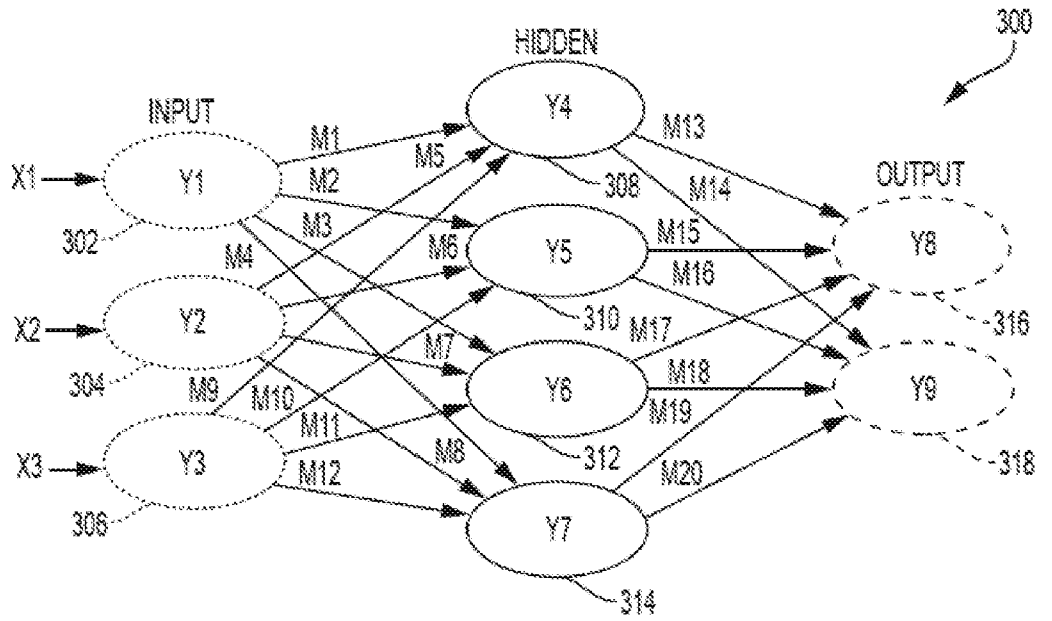


FIG. 1



f(x) = f (INPUT 1 * CONNECTION STRENGTH 1 + INPUT 2 * CONNECTION STRENGTH 2)

FIG. 2



Y1 = F(X1)
Y2 = F(X2)
Y3 = F(X3)

Y4 = F(M1*Y1 + M5*Y2 + M9*Y3)
Y5 = F(M2*Y1 + M6*Y2 + M10*Y3)
Y6 = F(M3*Y1 + M7*Y2 + M11*Y3)
Y7 = F(M4*Y1 + M8*Y2 + M12*Y3)

Y8 = F(M13*Y4 + M15*Y5 + M17*Y6 + M19*Y7)
Y9 = F(M14*Y4 + M16*Y5 + M18*Y6 + M20*Y7)

FIG. 3

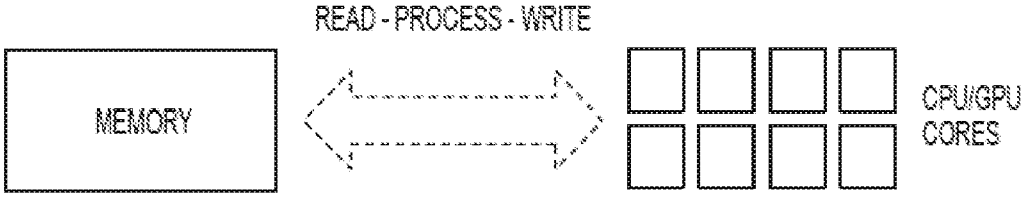


FIG. 4

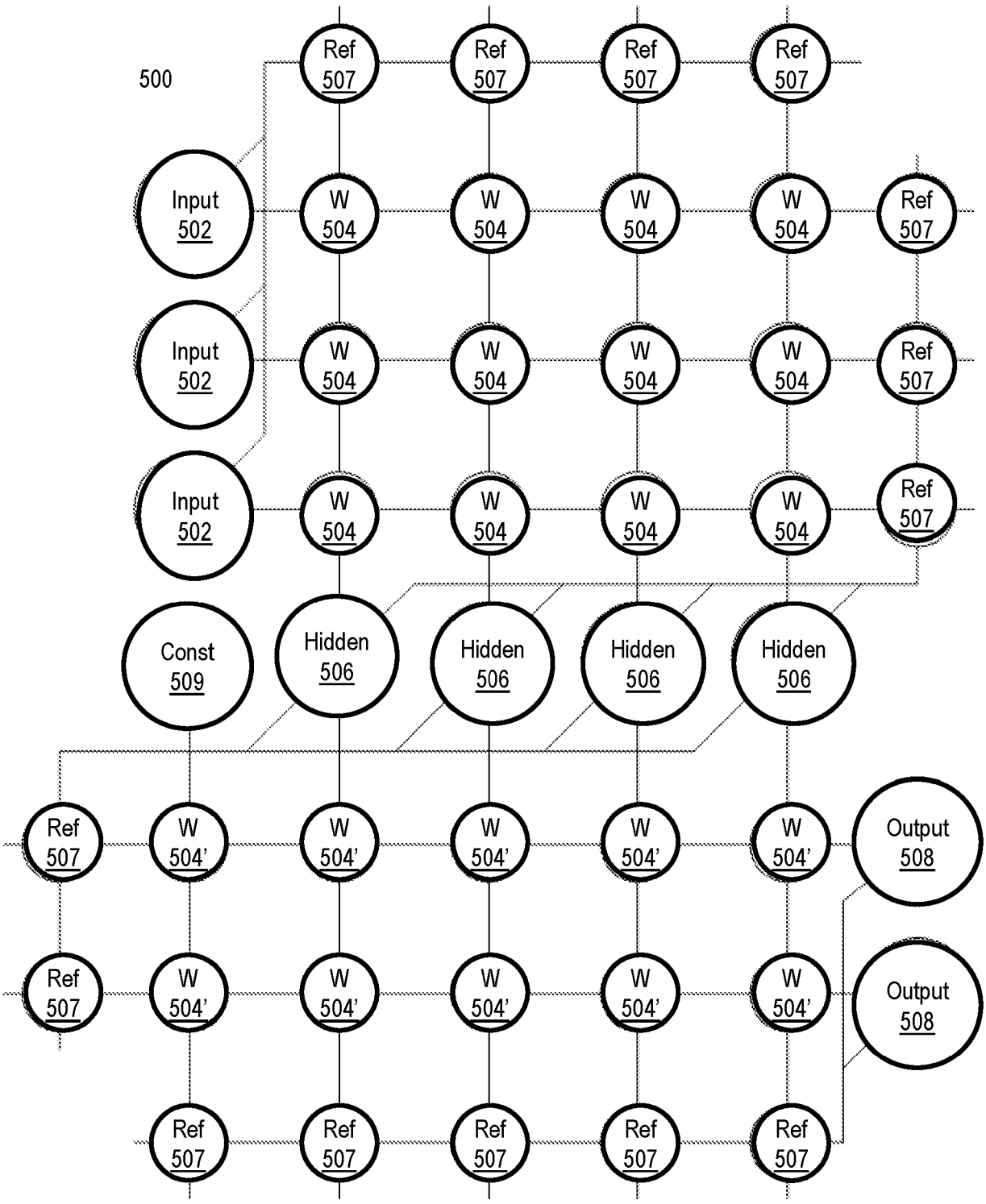

500
Ref 507
Input 502
W 504
Const 509
Hidden 506
W 504'
Output 508

FIG. 5

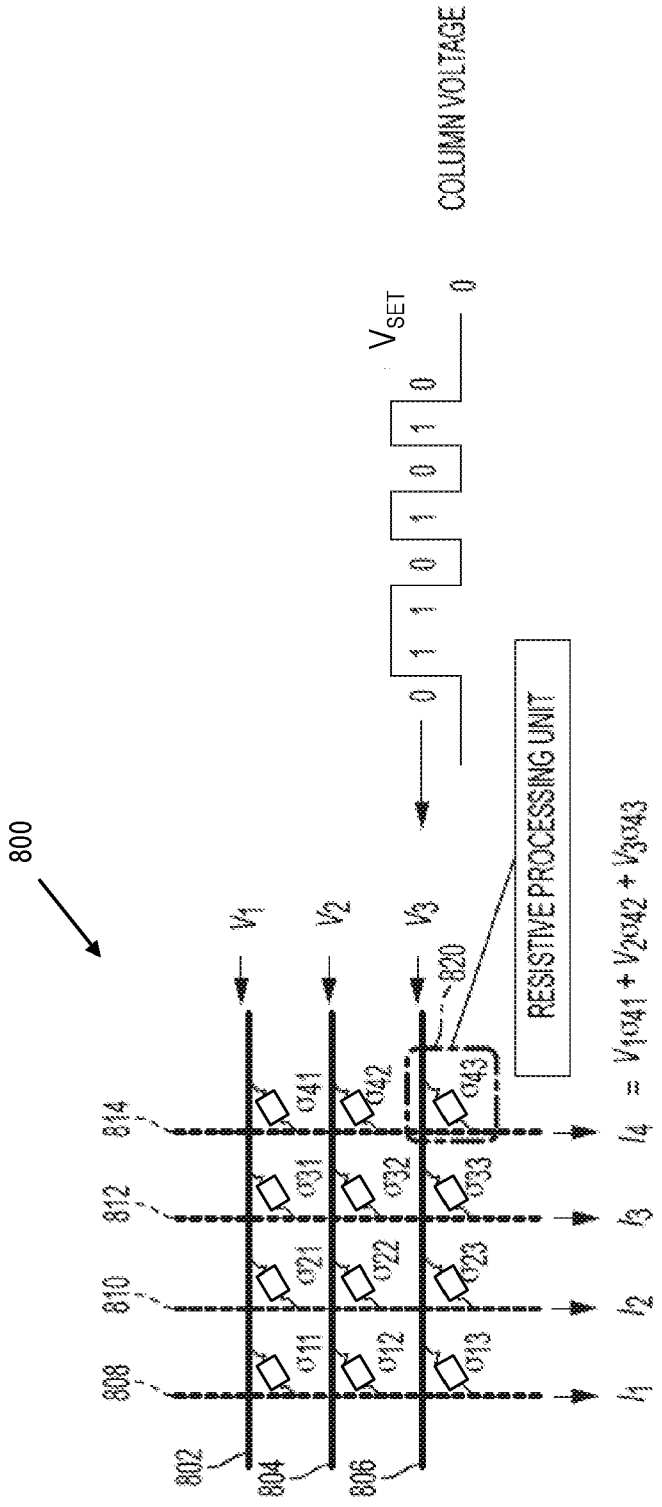

800
COLUMN VOLTAGE
V_{SET}
0
1 1 1 0 1 0 1 0 1 0
V_1
V_2
V_3
820
RESISTIVE PROCESSING UNIT
$I_4 = V_{1041} + V_{2042} + V_{3043}$
I_3
I_2
I_1
802
804
806
808
810
812
814
011
012
013
021
022
023
031
032
033
041
042
043

FIG. 6

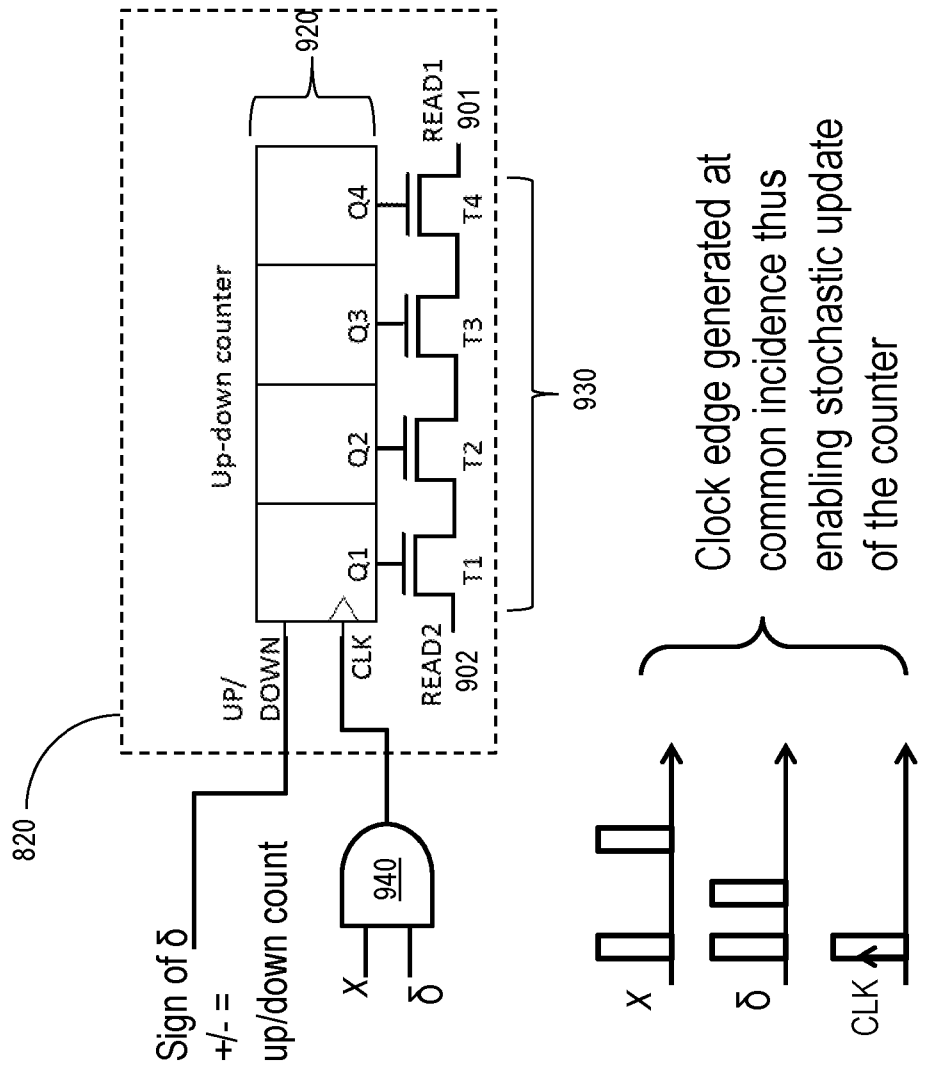

820
920
Up-down counter
UP/
DOWN
CLK
Q1
Q2
Q3
Q4
T1
T2
T3
T4
READ2
902
READ1
901
930
Sign of δ
+/- =
up/down count
940
x
δ
Clock edge generated at
common incidence thus
enabling stochastic update
of the counter
x
δ
CLK

FIG. 7

Bit states for T4, T3, T2, T1	Total stack resistance*
0, 0, 0, 0	$15R_{hi}$
0, 0, 0, 1	$14R_{hi} + R_{low}$
0, 0, 1, 0	$13R_{hi} + 2R_{low}$
0, 0, 1, 1	$12R_{hi} + 3R_{low}$
. . .	
1, 1, 1, 0	$R_{hi} + 14R_{low}$
1, 1, 1, 1	$15R_{low}$



Resistance linearly varies in steps of $(R_{hi} - R_{low})$

* R_{hi} = Off-state resistance of T1 (eg. 100K)
R_{low} = On-state resistance of T1 (eg. 10K)

FIG. 8

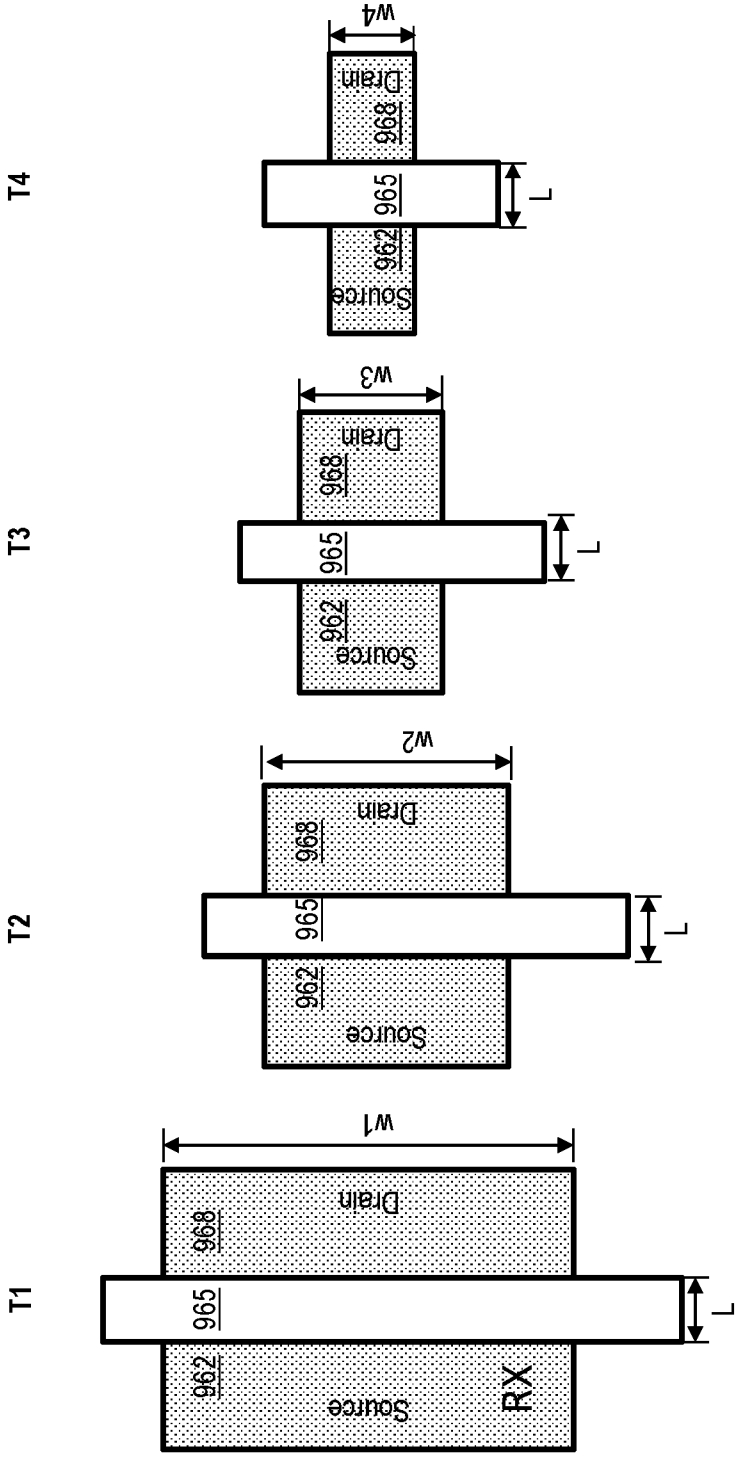


FIG. 9

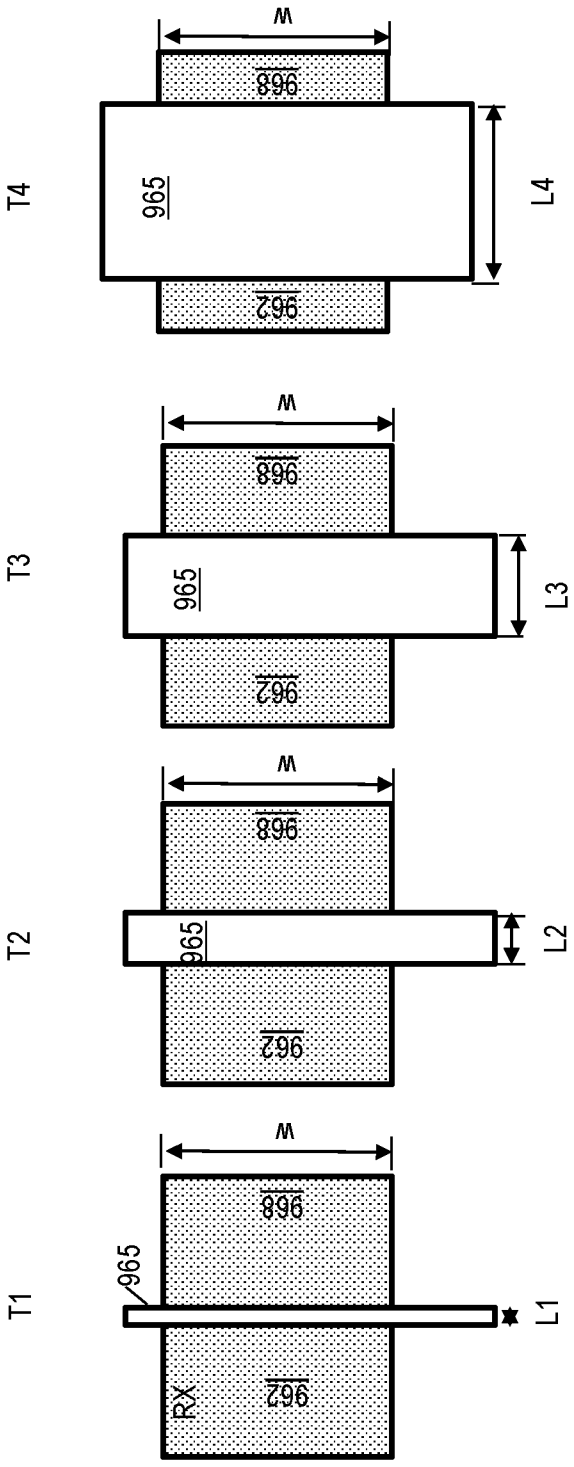

T4
W
968
965
962
L4
T3
W
968
965
962
L3
T2
W
968
965
962
L2
T1
W
968
965
RX
962
L1

FIG. 10

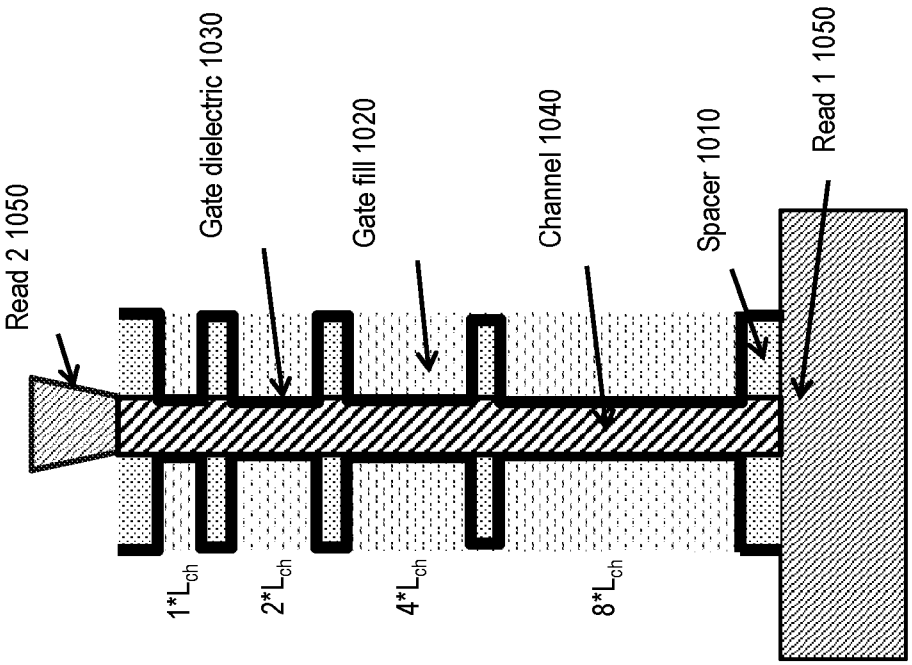

Read 2 1050
Gate dielectric 1030
Gate fill 1020
Channel 1040
Spacer 1010
Read 1 1050
1*Lch
2*Lch
4*Lch
8*Lch

FIG. 11

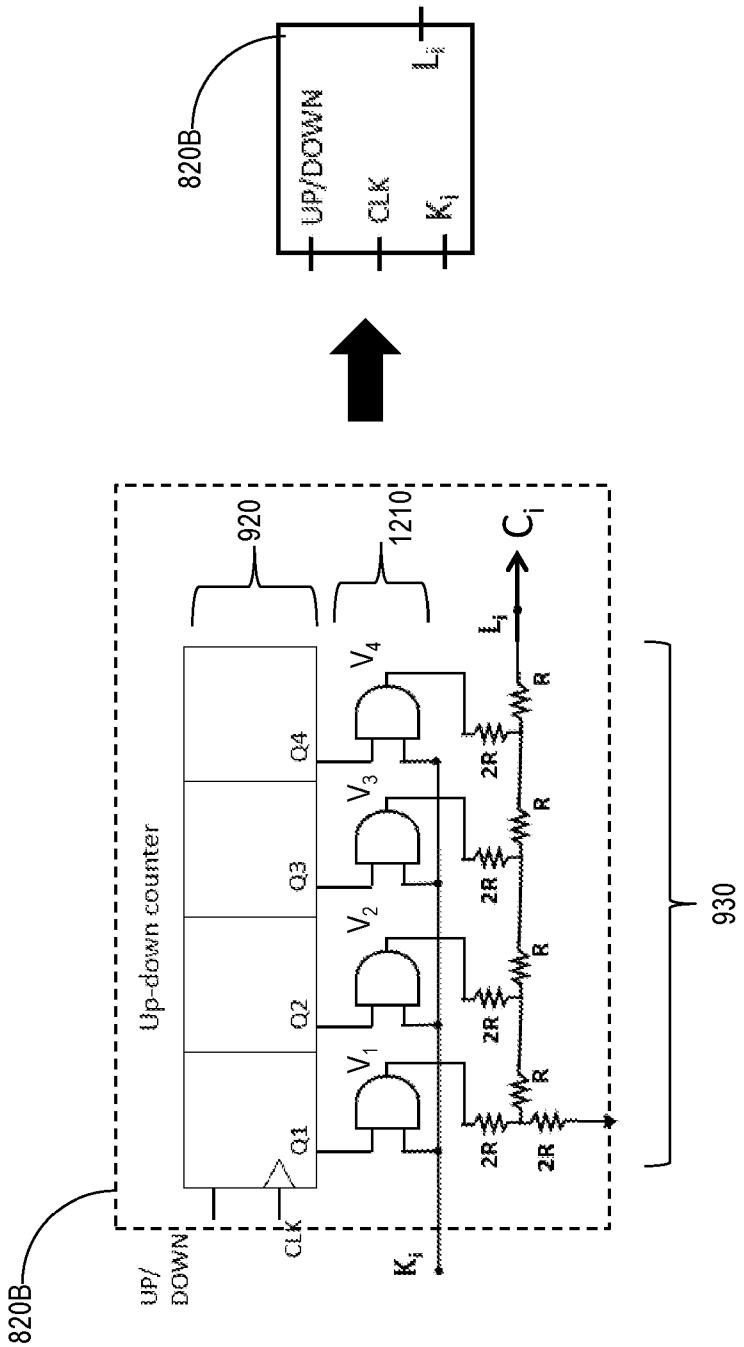
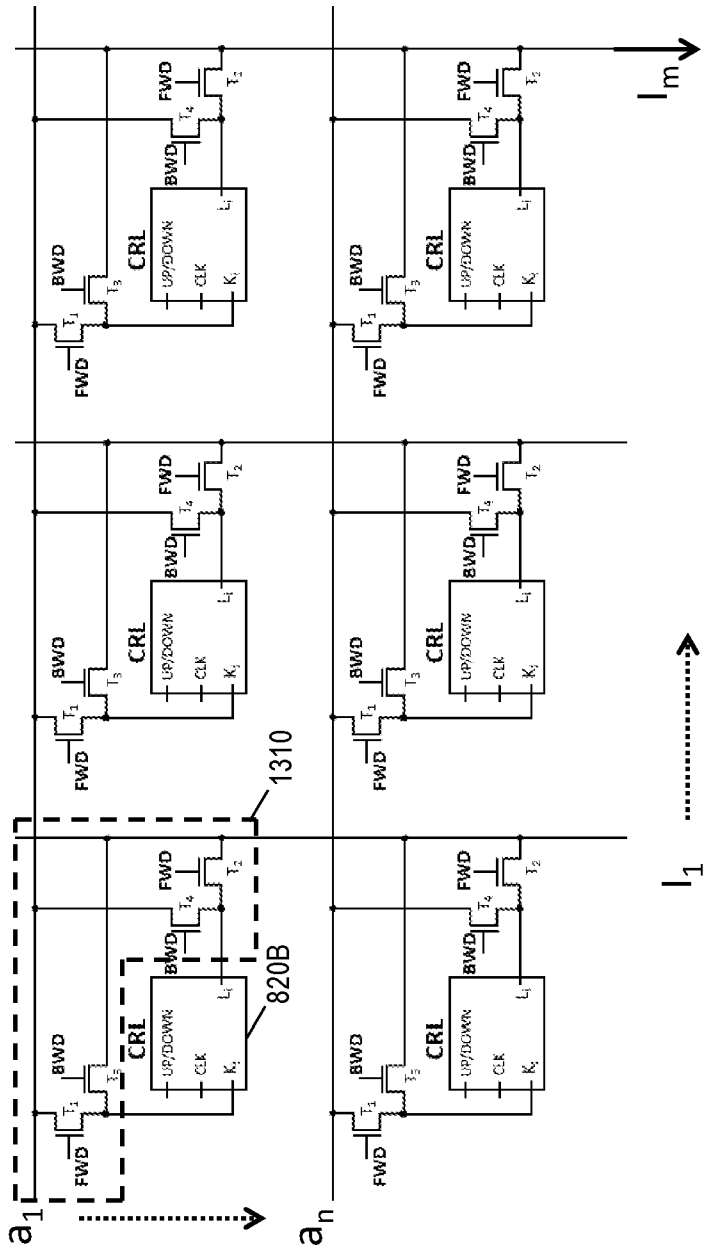

820B
UP/DOWN
CLK
K_i
L_i
820B
Up-down counter
UP/
DOWN
CLK
Q1
Q2
Q3
Q4
920
1210
V_1
V_2
V_3
V_4
K_i
L_i
C_i
2R
R
930

FIG. 12



* $<a_1:a_n>$ = input vector (time modulated)
  $<I_1:I_m>$ = Current from each column (from multiplication + accumulation)

FIG. 13

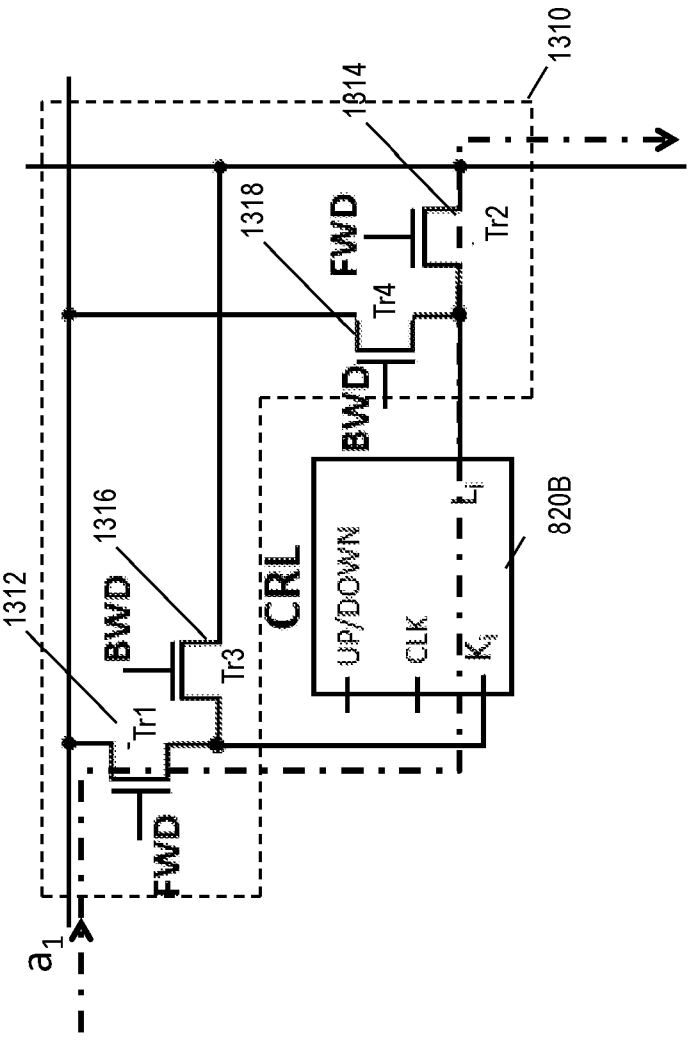

1310
1314
1318
1316
1312
FWD
Tr2
Tr4
BWD
BWD
Tr3
CRL
UP/DOWN
CLK
L_i
K_i
820B
Tr1
FWD
a_1

FIG. 14

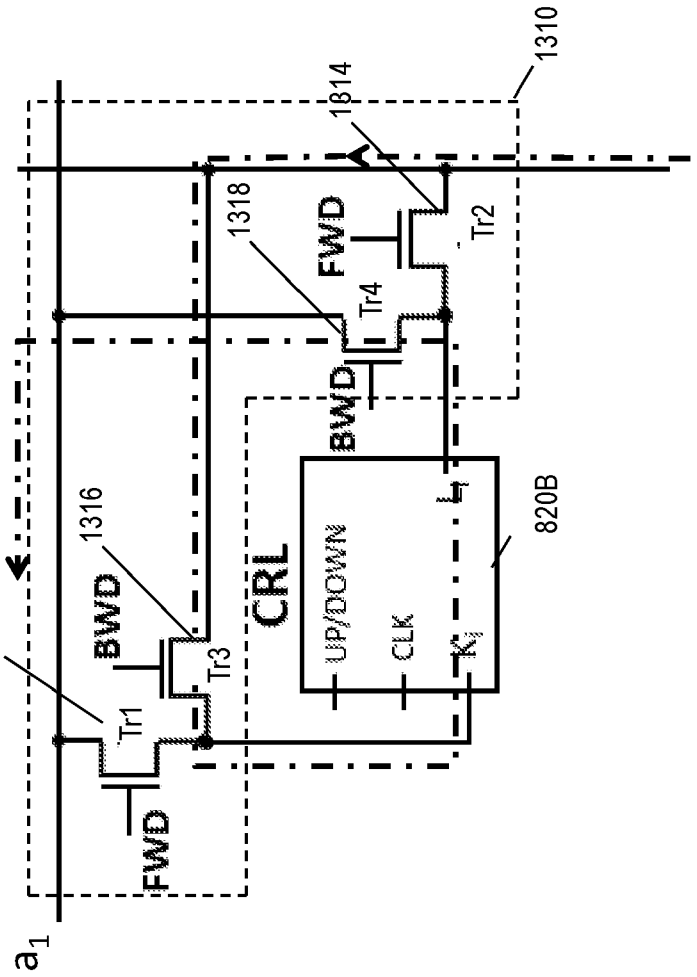

1310
1314
1318
FWD
Tr2
Tr4
BWD
1316
BWD
Tr3
CRL
UP/DOWN
CLK
K_i
820B
1312
Tr1
FWD
a_1

FIG. 15

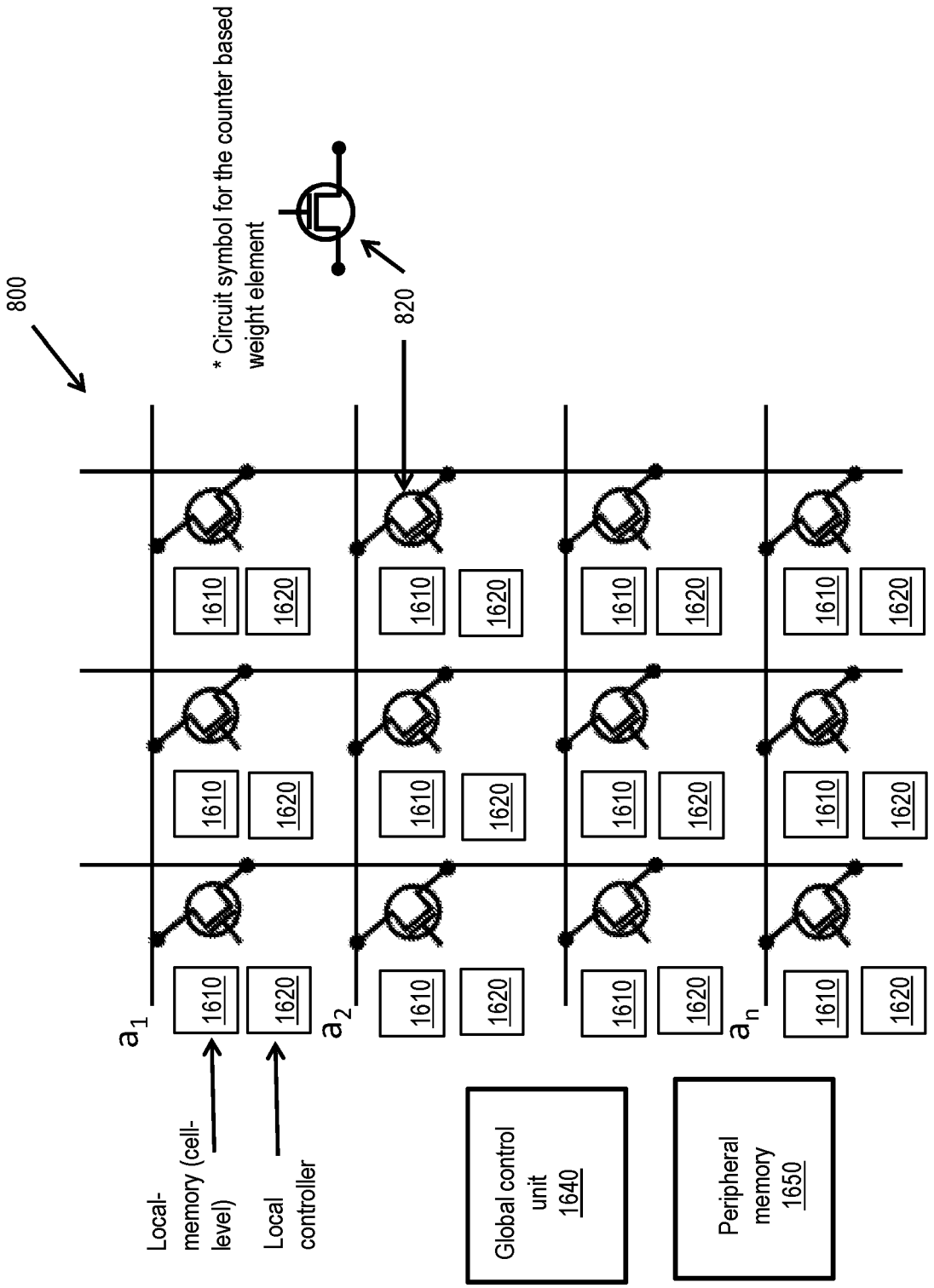

800
* Circuit symbol for the counter based weight element
820
1610
1620
a_1
a_2
a_n
Local-memory (cell-level)
Local controller
Global control unit 1640
Peripheral memory 1650

FIG. 16

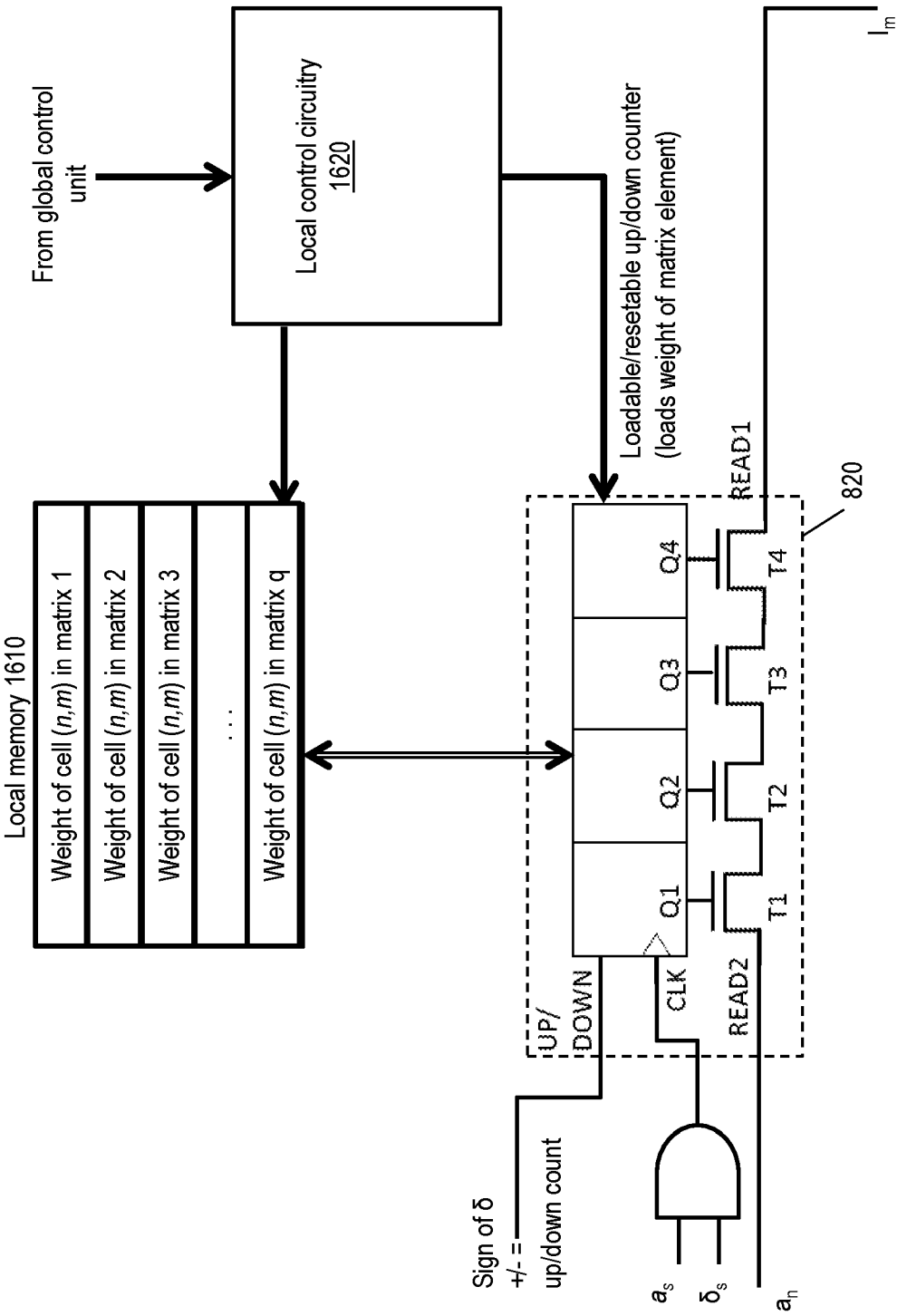

From global control unit
Local control circuitry 1620
Local memory 1610
Weight of cell (n,m) in matrix 1
Weight of cell (n,m) in matrix 2
Weight of cell (n,m) in matrix 3
Weight of cell (n,m) in matrix q
Loadable/resetable up/down counter (loads weight of matrix element)
820
Q1
Q2
Q3
Q4
T1
T2
T3
T4
READ1
READ2
CLK
UP/DOWN
Sign of δ
+/- =
up/down count
a_s
δ_s
a_n
I_m

FIG. 17

COUNTER BASED RESISTIVE PROCESSING UNIT FOR PROGRAMMABLE AND RECONFIGURABLE ARTIFICIAL-NEURAL-NETWORKS

BACKGROUND

[0001] The present invention relates in general to configurations of trainable resistive crosspoint devices, which are referred to herein as resistive processing units (RPUs). More specifically, the present invention relates to artificial neural networks (ANNs) formed from crossbar arrays of resistive processing units (RPUs) that provide local data storage and local data processing without the need for additional processing elements beyond the RPU, thereby accelerating the ANN's ability to implement algorithms such as matrix inversion, matrix decomposition and the like.

[0002] Technical problems such as character recognition and image recognition by a computer are known to be well handled by machine-learning techniques. "Machine learning" is used to broadly describe a primary function of electronic systems that learn from data. In machine learning and cognitive science, ANNs are a family of statistical learning models inspired by the biological neural networks of animals, and in particular the brain. ANNs can be used to estimate or approximate systems and functions that depend on a large number of inputs and are generally unknown. Crossbar arrays are high density, low cost circuit architectures used to form a variety of electronic circuits and devices, including ANN architectures, neuromorphic microchips and ultra-high density nonvolatile memory. A basic crossbar array configuration includes a set of conductive row wires and a set of conductive column wires formed to intersect the set of conductive row wires. The intersections between the two sets of wires are separated by so-called crosspoint devices.

13 04 22

SUMMARY

[0003] According to the present invention there are provided a method, and a crosspoint device according to the independent claims.

[0004] Deleted.

[0005] Deleted.

[0006] Deleted.

[0007] Deleted.

[0008] Additional technical features and benefits are realized through the techniques of the invention. Embodiments of the invention are described in detail herein and are considered a part of the claimed subject matter. For a better understanding, refer to the detailed description and to the drawings.

13 04 22

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The examples described herein will be better understood with reference to the following figures. The components in the figures are not necessarily to scale. Moreover, in the figures, like-referenced numerals designate corresponding parts throughout the different views.

[0010] FIG. 1 depicts a simplified diagram of input and output connections of a biological neuron;

[0011] FIG. 2 depicts a simplified model of the biological neuron shown in FIG. 1;

[0012] FIG. 3 depicts a simplified model of an ANN incorporating the biological neuron model shown in FIG. 2;

[0013] FIG. 4 depicts a simplified block diagram of a weight update methodology;

[0014] FIG. 5 is a diagram of an ANN including arrays of weights in accordance with an embodiment of the invention;

[0015] FIG. 6 depicts a cross bar array of RPU devices according to an embodiment of the invention, along with voltage sequences illustrating the operation of the RPU;

[0016] FIG. 7 depicts a counter-based RPU that can be used as a crosspoint device in an RPU array according to an embodiment of the invention;

[0017] FIG. 8 depicts adjustable resistance of the resistor device based at least in part on the states of the counter;

[0018] FIG. 9 depicts varying the resistance/conductance of FETs based at least in part on width;

[0019] FIG. 10 depicts varying the resistance/conductance of FETs based at least in part on gate length;

[0020] FIG. 11 depicts an example of the resistor device with variable resistance/conductance using vertical geometry;

[0021] FIG. 12 depicts a counter-based RPU that can be used as a crosspoint device in an RPU array according to an embodiment of the invention;

[0022] FIG. 13 depicts how the RPU with resistor ladder is connected as a crosspoint device in the RPU array;

[0023] FIG. 14 depicts a forward propagation pass performed using the RPU and a connection circuit;

[0024] FIG. 15 depicts a backward propagation pass performed using the RPU and the connection circuit;

[0025] FIG. 16 depicts an RPU array using the RPU devices; and

[0026] FIG. 17 depicts a dataflow in an RPU array using the RPU devices.

[0027] The diagrams depicted herein are illustrative. There can be many variations to the diagram or the operations described therein without departing from the scope of the invention. For instance, the actions can be performed in a differing order or actions can be added, deleted or modified. Also, the term "coupled" and variations thereof describes having a communications path between two elements and does not imply a direct connection between the elements with no intervening elements/connections between them. All of these variations are considered a part of the specification.

13 04 22

DETAILED DESCRIPTION

[0028] Various embodiments of the invention are described herein with reference to the related drawings. Alternative embodiments of the invention can be devised without departing from the scope of this invention. Various connections and positional relationships (e.g., over, below, adjacent, etc.) are set forth between elements in the following description and in the drawings. These connections and/or positional relationships, unless specified otherwise, can be direct or indirect, and the present invention is not intended to be limiting in this respect. Accordingly, a coupling of entities can refer to either a direct or an indirect coupling, and a positional relationship between entities can be a direct or indirect positional relationship. Moreover, the various tasks and process steps described herein can be incorporated into a more comprehensive procedure or process having additional steps or functionality not described in detail herein.

[0029] The following definitions and abbreviations are to be used for the interpretation of the claims and the specification. As used herein, the terms "comprises," "comprising," "includes," "including," "has," "having," "contains" or "containing," or any other variation thereof, are intended to cover a non-exclusive inclusion. For example, a composition, a mixture, process, method, article, or apparatus that comprises a list of elements is not necessarily limited to only those elements but can include other elements not expressly listed or inherent to such composition, mixture, process, method, article, or apparatus.

[0030] Additionally, the term "exemplary" is used herein to mean "serving as an example, instance or

illustration." Any embodiment or design described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other embodiments or designs. The terms "at least one" and "one or more" can be understood to include any integer number greater than or equal to one, i.e. one, two, three, four, etc. The terms "a plurality" can be understood to include any integer number greater than or equal to two, i.e. two, three, four, five, etc. The term "connection" can include both an indirect "connection" and a direct "connection."

[0031] The terms "about," "substantially," "approximately," and variations thereof, are intended to include the degree of error associated with measurement of the particular quantity based upon the equipment available at the time of filing the application. For example, "about" can include a range of ± 8% or 5%, or 2% of a given value.

[0032] For the sake of brevity, conventional techniques related to making and using aspects of the invention may or may not be described in detail herein. In particular, various aspects of computing systems and specific computer programs to implement the various technical features described herein are well known. Accordingly, in the interest of brevity, many conventional implementation details are only mentioned briefly herein or are omitted entirely without providing the well-known system and/or process details.

[0033] It is understood in advance that although one or more embodiments of the invention are described in the context of biological neural networks with a specific emphasis on modeling brain structures and functions, implementation of the teachings recited herein are not limited to modeling a particular environment. Rather, embodiments of the invention are capable of modeling any type of environment, including for example, weather patterns, arbitrary data collected from the internet, and the like, as long as the various inputs to the environment can be turned into a vector.

[0034] Artificial neural networks (ANNs) can be used to estimate or approximate systems and functions that depend on a large number of inputs and are generally unknown. Neural networks use a class of algorithms based on a concept of inter-connected "neurons." In a typical neural network, neurons have a given activation function that operates on the inputs. By determining proper connection weights (a process also referred to as "training"), a neural network achieves efficient recognition of a desired patterns, such as images and characters. Oftentimes, these neurons are grouped into "layers" in order to make connections between groups more obvious and to each computation of values. Training the neural network is a computationally intense process.

[0035] ANNs are often embodied as so-called "neuromorphic" systems of interconnected processor elements that act as simulated "neurons" and exchange "messages" between each other in the form of electronic signals. Similar to the so-called "plasticity" of synaptic neurotransmitter connections that carry messages between biological neurons, the connections in ANNs that carry electronic messages between simulated neurons are provided with numeric weights that correspond to the strength or weakness of a given connection. The weights can be adjusted and tuned based on experience, making ANNs adaptive to inputs and capable of learning. For example, an ANN

for handwriting recognition is defined by a set of input neurons which can be activated by the pixels of an input image. After being weighted and transformed by a function determined by the network's designer, the activations of these input neurons are then passed to other downstream neurons, which are often referred to as "hidden" neurons. This process is repeated until an output neuron is activated. The activated output neuron determines which character was read.

[0036] Crossbar arrays, also known as crosspoint arrays or crosswire arrays, are high density, low cost circuit architectures used to form a variety of electronic circuits and devices, including ANN architectures, neuromorphic microchips and ultra-high density nonvolatile memory. A basic crossbar array configuration includes a set of conductive row wires and a set of conductive column wires formed to intersect the set of conductive row wires. The intersections between the two sets of wires are separated by so-called crosspoint devices, which can be formed from thin film material.

[0037] Crosspoint devices, in effect, function as the ANN's weighted connections between neurons. Nanoscale devices, for example memristors having "ideal" conduction state switching characteristics, are often used as the crosspoint devices in order to emulate synaptic plasticity with high energy efficiency. The conduction state (e.g., resistance) of the ideal memristor material can be altered by controlling the voltages applied between individual wires of the row and column wires. Digital data can be stored by alteration of the memristor material's conduction state at the intersection to achieve a high conduction state or a low conduction state. The memristor material can also be programmed to maintain two or more distinct conduction states by selectively setting the conduction state of the material. The conduction state of the memristor material can be read by applying a voltage across the material and measuring the current that passes through the target crosspoint device. So far, however, there have been some practical drawbacks in memristor and other resistive-random-access-memory (RRAM) based crosspoint devices that are detrimental to ANN applications; significant device-to-device variability and the asymmetry between "set (i.e. to increment resistance)" and "reset (i.e. to decrement resistance)" operations are two such main limitations.

[0038] In order to limit power consumption, the crosspoint devices of ANN chip architectures are often designed to utilize offline learning techniques, wherein the approximation of the target function does not change once the initial training phase has been resolved. Offline learning allows the crosspoint devices of crossbar-type ANN architectures to be simplified such that they draw very little power.

[0039] Notwithstanding the potential for lower power consumption, executing offline training can be difficult and resource intensive because it is typically necessary during training to modify a significant number of adjustable parameters (e.g., weights) in the ANN model to match the input-output pairs for the training data. Accordingly, simplifying the crosspoint devices of ANN architectures to prioritize power-saving, offline learning techniques typically means that training speed and training efficiency are not optimized.

[0040] Providing simple crosspoint devices that keep power consumption within an acceptable range, as well as accelerate the speed and efficiency of training ANN architectures, would improve overall ANN performance and allow a broader range of ANN applications. Accordingly, described herein are technical solutions that facilitate using and training crosspoint arrays using counter-based crosspoint devices that can provide adjustable conductance (resistance) representing corresponding weights to be stored in the array. Further, comparing to typical capacitor based implementations for weight storage, the technical solutions described herein address technical challenges such as charge-loss, and stringent requirements on update currents. Also, unlike memristor or RRAM based crosspoint devices, the counter-based crosspoint device has lesser device-to-device variability (afforded by well-established semiconductor technological processes) and ideal symmetry between up-counting vs. down-counting of the weight values.

[0041] The technical solutions described herein further facilitate updating the values stored by the counter-based crosspoint devices based at least in part on specific clock events. The clock events facilitate the crosspoint array that represents a neural network to update the stored weights after forward and/or a backward propagation. The stored weight is then used during subsequent forward and/or backward propagation.

[0042] The use of counter-based crosspoint devices as the RPUs in the RPU array facilitates the RPU array to have robustness of conventional CMOS technology. Further, the use of digital counters facilitates symmetry between up vs. down weight increments. Further yet, the counter-based crosspoint devices facilitates the RPU array to work with stochastic parallel update algorithms for back-propagation.

[0043] In one or more examples, weight elements are stored in digital counters (e.g. J-K flip-flop based counters) of the crosspoint devices. Each bit in the counter is associated with one or more resistive circuit from a stack of resistive circuits of the crosspoint devices. The counter bits control the gate terminals of FETs connected in a series such that the total resistance of the FET stack will be used as the transduction element (i.e. resistor element of each RPU cell). Thus, the counter bits facilitate controlling a total resistance of the stack of resistive circuits as described in detail further.

[0044] Although embodiments of the invention are directed to electronic systems, for ease of reference and explanation various aspects of the electronic systems are described using neurological terminology such as neurons, plasticity and synapses, for example. It will be understood that for any discussion or illustration herein of an electronic system, the use of neurological terminology or neurological shorthand notations are for ease of reference and are meant to cover the neuromorphic, ANN equivalent(s) of the described neurological function or neurological component.

[0045] Instead of utilizing the traditional digital model of manipulating zeros and ones, ANNs create connections between processing elements that are substantially the functional equivalent of the core system

functionality that is being estimated or approximated. For example, IBM™'s SyNapse™ computer chip is the central component of an electronic neuromorphic machine that attempts to provide similar form, function and architecture to the mammalian brain. Although the IBM SyNapse computer chip uses the same basic transistor components as conventional computer chips, its transistors are configured to mimic the behavior of neurons and their synapse connections. The IBM SyNapse computer chip processes information using a network of just over one million simulated "neurons," which communicate with one another using electrical spikes similar to the synaptic communications between biological neurons. The IBM SyNapse architecture includes a configuration of processors (i.e., simulated "neurons") that read a memory (i.e., a simulated "synapse") and perform simple operations. The communications between these processors, which are typically located in different cores, are performed by on-chip network routers.

[0046] A general description of how a typical ANN operates will now be provided with reference to FIGS. 1, 2 and 3. As previously noted herein, a typical ANN models the human brain, which includes about one hundred billion interconnected cells called neurons. FIG. 1 depicts a simplified diagram of a biological neuron 102 having pathways 104, 106, 108, 110 that connect it to upstream inputs 112, 114, downstream outputs 116 and downstream "other" neurons 118, configured and arranged as shown. Each biological neuron 102 sends and receives electrical impulses through pathways 104, 106, 108, 110. The nature of these electrical impulses and how they are processed in biological neuron 102 are primarily responsible for overall brain functionality. The pathway connections between biological neurons can be strong or weak. When a given neuron receives input impulses, the neuron processes the input according to the neuron's function and sends the result of the function to downstream outputs and/or downstream "other" neurons.

[0047] Biological neuron 102 is modeled in FIG. 2 as a node 202 having a mathematical function, f(x) depicted by the equation shown in FIG. 2. Node 202 takes electrical signals from inputs 212, 214, multiplies each input 212, 214 by the strength of its respective connection pathway 204, 206, takes a sum of the inputs, passes the sum through a function, f(x), and generates a result 216, which can be a final output or an input to another node, or both. Herein, an asterisk (*) is used to represent a multiplication. Weak input signals are multiplied by a very small connection strength number, so the impact of a weak input signal on the function is very low. Similarly, strong input signals are multiplied by a higher connection strength number, so the impact of a strong input signal on the function is larger. The function f(x) is a design choice, and a variety of functions can be used. An example design choice for f(x) is the hyperbolic tangent function, which takes the function of the previous sum and outputs a number between minus one and plus one.

[0048] FIG. 3 depicts a simplified ANN model 300 organized as a weighted directional graph, wherein the artificial neurons are nodes (e.g., 302, 308, 316), and wherein weighted directed edges (e.g., m1 to m20) connect the nodes. ANN model 300 is organized such that nodes 302, 304, 306 are input layer nodes, nodes 308, 310, 312, 314 are hidden layer nodes and nodes 316, 318 are output layer nodes. Each node is connected to every

node in the adjacent layer by connection pathways, which are depicted in FIG. 3 as directional arrows having connection strengths m1 to m20. Although only one input layer, one hidden layer and one output layer are shown, in practice, multiple input layers, hidden layers and output layers can be provided.

[0049] Similar to the functionality of a human brain, each input layer node 302, 304, 306 of ANN 300 receives inputs x1, x2, x3 directly from a source (not shown) with no connection strength adjustments and no node summations. Accordingly, y1 = f(x1), y2 = f(x2) and y3 = f(x3), as shown by the equations listed at the bottom of FIG. 3. Each hidden layer node 308, 310, 312, 314 receives its inputs from all input layer nodes 302, 304, 306 according to the connection strengths associated with the relevant connection pathways. Thus, in hidden layer node 308, y4 = f(m1*y1 + m5*y2 + m9*y3), wherein * represents a multiplication. A similar connection strength multiplication and node summation is performed for hidden layer nodes 310, 312, 314 and output layer nodes 316, 318, as shown by the equations defining functions y5 to y9 depicted at the bottom of FIG. 3.

[0050] ANN model 300 processes data records one at a time, and it "learns" by comparing an initially arbitrary classification of the record with the known actual classification of the record. Using a training methodology knows as "backpropagation" (i.e., "backward propagation of errors"), the errors from the initial classification of the first record are fed back into the network and used to modify the network's weighted connections the second time around, and this feedback process continues for many iterations. In the training phase of an ANN, the correct classification for each record is known, and the output nodes can therefore be assigned "correct" values. For example, a node value of "1" (or 0.9) for the node corresponding to the correct class, and a node value of "0" (or 0.1) for the others. It is thus possible to compare the network's calculated values for the output nodes to these "correct" values, and to calculate an error term for each node (i.e., the "delta" rule). These error terms are then used to adjust the weights in the hidden layers so that in the next iteration the output values will be closer to the "correct" values.

[0051] There are many types of neural networks, but the two broadest categories are feed-forward and feedback/recurrent networks. ANN model 300 is a non-recurrent feed-forward network having inputs, outputs and hidden layers. The signals can only travel in one direction. Input data is passed onto a layer of processing elements that perform calculations. Each processing element makes its computation based upon a weighted sum of its inputs. The new calculated values then become the new input values that feed the next layer. This process continues until it has gone through all the layers and determined the output. A threshold transfer function is sometimes used to quantify the output of a neuron in the output layer.

[0052] A feedback/recurrent network includes feedback paths, which mean that the signals can travel in both directions using loops. All possible connections between nodes are allowed. Because loops are present in this type of network, under certain operations, it can become a non-linear dynamical system that changes continuously until it reaches a state of equilibrium. Feedback networks are often used in associative memories and optimization

problems, wherein the network looks for the best arrangement of interconnected factors.

[0053] The speed and efficiency of machine learning in feed-forward and recurrent ANN architectures depend on how effectively the crosspoint devices of the ANN crossbar array perform the core operations of typical machine learning algorithms. Although a precise definition of machine learning is difficult to formulate, a learning process in the ANN context can be viewed as the problem of updating the crosspoint device connection weights so that a network can efficiently perform a specific task. The crosspoint devices typically learn the necessary connection weights from available training patterns. Performance is improved over time by iteratively updating the weights in the network. Instead of following a set of rules specified by human experts, ANNs "learn" underlying rules (like input-output relationships) from the given collection of representative examples. Accordingly, a learning algorithm can be generally defined as the procedure by which learning rules are used to update and/or adjust the relevant weights.

[0054] The three main learning algorithm paradigms are supervised, unsupervised and hybrid. In supervised learning, or learning with a "teacher," the network is provided with a correct answer (output) for every input pattern. Weights are determined to allow the network to produce answers as close as possible to the known correct answers. Reinforcement learning is a variant of supervised learning in which the network is provided with only a critique on the correctness of network outputs, not the correct answers themselves. In contrast, unsupervised learning, or learning without a teacher, does not require a correct answer associated with each input pattern in the training data set. It explores the underlying structure in the data, or correlations between patterns in the data, and organizes patterns into categories from these correlations. Hybrid learning combines supervised and unsupervised learning. Parts of the weights are usually determined through supervised learning, while the others are obtained through unsupervised learning.

[0055] As previously noted herein, in order to limit power consumption, the crosspoint devices of ANN chip architectures are often designed to utilize offline learning techniques, wherein the approximation of the target function does not change once the initial training phase has been resolved. Offline learning allows the crosspoint devices of crossbar-type ANN architectures to be simplified such that they draw very little power.

[0056] Notwithstanding the potential for lower power consumption, executing offline training can be difficult and resource intensive because it is typically necessary during training to modify a significant number of adjustable parameters (e.g., weights) in the ANN model to match the input-output pairs for the training data. FIG. 4 depicts a simplified illustration of a typical read-process-write weight update operation, wherein CPU/GPU cores (i.e., simulated "neurons") read a memory (i.e., a simulated "synapse") and perform weight update processing operations, then write the updated weights back to memory. Accordingly, simplifying the crosspoint devices of ANN architectures to prioritize power-saving, offline learning techniques typically means that training speed and training efficiency are not optimized.

[0057] FIG. 5 illustrates an artificial neural network (ANN) architecture 500. During feed-forward operation, a set of input neurons 502 each provide an input voltage in parallel to a respective row of weights 504. A weight 504 is a crosspoint device, such as an RPU device which is a counter-based weight element described herein. The weights 504 each have a settable resistance value, such that a current output flows from the weight 504 to a respective hidden neuron 506 to represent the weighted input. The current output by a given weight is determined as $I = \frac{V}{r}$, where V is the input voltage from the input neuron 502 and r is the set resistance of the weight 504. The current from each weight adds column-wise and flows to a hidden neuron 506.

[0058] The hidden neurons 506 use the currents from the array of weights 504 to perform some calculation. The hidden neurons 506 then output a voltage of their own to another array of weights 504'. This array performs in the same way, with a column of weights 504' receiving a voltage from their respective hidden neuron 506 to produce a weighted current output that adds row-wise and is provided to the output neuron 508.

[0059] It should be understood that any number of these stages can be implemented, by interposing additional layers of arrays and hidden neurons 506.

[0060] During back propagation, the output neurons 508 provide a voltage back across the array of weights 504'. The output layer compares the generated network response to training data and computes an error. The error is applied to the array as a voltage pulse, where the height and/or duration of the pulse is modulated proportional to the error value. In this example, a row of weights 504' receives a voltage from a respective output neuron 508 in parallel and converts that voltage into a current which adds column-wise to provide an input to hidden neurons 506. The hidden neurons 506 combine the weighted feedback signal with a derivative of its feed-forward calculation and stores an error value before outputting a feedback signal voltage to its respective column of weights 504. It should be noted that the weights 504' operate in the same manner as the weights 504; the labeling is provided to indicate that weights 504 are between layers of neurons 502 and neurons 506, and weights 504' are between layers of neurons 506 and 508. This back propagation travels through the entire network 500 until all hidden neurons 506 and the input neurons 502 have stored an error value.

[0061] During weight updates, the input neurons 502 and hidden neurons 506 apply first weight update voltages to the crosspoint array of weights 504 and the output neurons 508 and hidden neurons 506 apply second weight update voltages to the crosspoint array of weights 504' through the network 500. Further, during the update phase of the weights 504, the input layer 502 applies voltage pulses (row-wise) proportional to the input values, and the layer 506 applies voltage pulses proportional to the error values of the hidden layer 506 (column-wise). Further yet, during update phase of the neurons 504', hidden layer 506 applies voltage pulses proportional to its feed-forward output values (column-wise), and the output layer 508 applies voltage pulses proportional to the error of the output layer 508 (row-wise). The combinations of these voltages create a state change within each weight 504, 504', causing the weight 504, 504' to take on a new counter-value, which in turn changes the resistance value. In

this manner, the weights 504, 504' can be trained to adapt the neural network 500 to errors in its processing. It should be noted that the three modes of operation, feed forward, back propagation, and weight update, do not overlap with one another.

[0062] Turning now to an overview of the invention, an embodiment of the invention is directed to a programmable resistive crosspoint component referred to herein as a crosspoint device, or a resistive processing unit (RPU), which provides local data storage functionality and local data processing functionality. In other words, when performing data processing, the value stored at each RPU is updated in parallel and locally, which eliminate the need to move relevant data in and out of a processor and a separate storage element. Additionally, the local data storage and local data processing provided by the described RPUs accelerate the ANN's ability to implement algorithms such as matrix inversion, matrix decomposition and the like. Accordingly, implementing a machine learning ANN architecture having the described RPU enables the implementation that optimize the speed, efficiency and power consumption of the ANN. The described RPU and resulting ANN architecture improve overall ANN performance and enable a broader range of practical ANN applications.

[0063] The described RPU can be implemented as resistive cross point devices, including a counter and a resistor device that includes a stack of resistive circuits. The counter includes digital circuits to store and output each bit in the counter. Each bit is associated with a respective resistive circuit from the stack of resistive circuits. Each resistive circuit provides a corresponding predetermined conductance (or resistance). In one or more examples, the conductance values of the resistive circuits from the stack in the resistor device have a predetermined relationship, such as a quadratic relationship. Based at least in part on a state of the bit (ON = 1, OFF = 0), the corresponding resistive circuit adds its corresponding conductance to the total conductance of the resistor device. Accordingly, based at least in part on the value stored in the counter, the conductance/resistance of the resistor device can be controlled.

[0064] FIG. 6 depicts a cross bar array of two-terminal, non-linear RPU devices embodying the invention, along with voltage sequences illustrating the operation of the RPU. FIG. 6 is a diagram of a two-dimensional (2D) crossbar array 800 that performs forward matrix multiplication, backward matrix multiplication and weight updates. Crossbar array 800 is formed from a set of conductive row wires 802, 804, 806 and a set of conductive column wires 808, 810, 812, and 814 that intersect the set of conductive row wires 802, 804, and 806. The intersections between the set of row wires and the set of column wires are separated by RPUs, which are shown in FIG 6 as resistive elements each having its own adjustable/updateable resistive weight, depicted as σ_{11}, σ_{21}, σ_{31}, σ_{41}, σ_{12}, σ_{22}, σ_{32}, σ_{42}, σ_{13}, σ_{23}, σ_{33} and σ_{43}, respectively. For ease of illustration, only one RPU 820 is labeled with a reference number in FIG. 6. In forward matrix multiplication, the conduction state (i.e., the stored weights) of the RPU can be read by applying a voltage across the RPU and measuring the current that passes through the RPU. In one or more examples, the crossbar array 800 includes two sets of conductive wires, that is two sets of rows and two sets of columns. In other words, each of 802, 804, 806, 808, 810, 812 and 814 includes two conductive wires.

The forward/backward propagation happen through the first set of wires and the update happens through the second set of wires.

[0065] Input voltages V_1, V_2, V_3 are applied to row wires 802, 804, 806, respectively. Each column wire 808, 810, 812, 814 sums the currents I_1, I_2, I_3, I_4 generated by each RPU along the particular column wire. For example, as shown in FIG. 6, the current I_4 generated by column wire 814 is according to the equation $I_4 = V_1\sigma_{41} + V_2\sigma_{42} + V_3\sigma_{43}$. Thus, array 800 computes the forward matrix multiplication by multiplying the values stored in the RPUs by the row wire inputs, which are defined by voltages V_1, V_2, V_3, and accumulating the current values at each column. The backward matrix multiplication (not shown) is very similar. In backward matrix multiplication, voltages are applied at column wires 808, 810, 812, 814 then currents are read and accumulated from row wires 802, 804, 806. For weight updates, which are described in greater detail below, voltages are applied to column wires and row wires at the same time, and the conductance values stored in the relevant RPU devices all update in parallel. Accordingly, the multiplication and addition operations required to perform weight updates are performed locally at each RPU 820 of array 800 using the RPU device itself plus the relevant row or column wire of array 800.

[0066] Continuing with the diagram of FIG. 6, in accordance with one or more embodiments, the operation of a positive weight update methodology for RPU 820 and its corresponding weight σ_{33} at the intersection of conductive row wire 806 and conductive column wire 812 will now be provided. Update generator circuitry (not shown) is provided at the periphery of crossbar array 800 and used as a peripheral "translator" in order to generate necessary voltage pulses in the form of stochastic bit streams that are applied to all RPUs of 2D crossbar array 800.

[0067] Accordingly, referring to the ANN implemented using a crosspoint array including RPUs as described herein, in the array, the value of the resistance (or conductance) of each node determines the coupling between nodes, where a node is represented by an RPU device in the array. Further, upon training the crosspoint array according to the ANN, the resistance (or conductance) will be different from device to device, depending on the desired coupling. For training a neural network, it is necessary to actively adjust the resistance values. Once the training is complete, the resistance values remain fixed during operation of the crosspoint array circuit, until training begins for a new task.

[0068] FIG. 7 depicts a counter-based RPU that can be used as a crosspoint device in an RPU array according to one or more embodiments. The RPU device 820 includes a pair of terminals 901, 902, which connect the RPU 820 to the crosspoint array 800.

[0069] Further, the counter-based RPU 820 includes a counter 920, which is a set of single bit counters, and resistor device 930, which is a stack of resistive circuits. In the depicted counter-based RPU, the resistive device includes transistors T1, T2, T3, and T4 as the resistive circuits. The transistors can be field-effect transistor (FETs) or any other type of transistors. However, in other embodiments, the resistive device 930 includes other types of

resistive circuits, such as a resistor ladder (FIG. 12) or any other type of circuits that can be stacked together to provide the adjustable/controllable resistance/conductance as provided by the technical solutions described herein.

[0070] The first set of conductive wires in the RPU array 800 is coupled with the READ1 901 and READ2 902 terminals of the crosspoint device 820 for reading the weight values stored in the crosspoint device 820 (as opposed to updating the weight values). The second set of conductive wires in the RPU array 800 is coupled with the logic gate 940 and the UP/DOWN counter 920 of the RPU device 820. Particularly, the second set of wires connect, (a) to the logic-gate inputs of the device, in turn generating the clock pulse under common incidence, and (b) connect to the counter UP/DOWN input to adjust the counter according to the weight value to be stored. The voltages used for the second set of wires are selected for correct logical operation of the logic-gate 940 (e.g. AND gate) and the counter 920 (typically, 0 and VDD). Furthermore, each column wire in the second set includes two wires: one carrying the voltage pulse proportional to the error value (δ) of the subsequent layer-neuron, and the second carrying the corresponding sign of the error value (e.g. 0V = - sign, VDD = +sign) to cause the increment/decrement at the clock pulse.

[0071] As mentioned earlier, forward/backward propagation happen using the first set of column and row wires, and the weight update happens through a second, distinct set of wires (rows and columns). During forward and backward propagation the voltage pulses are applied to wires in the first set of wires which are optimized for current accumulation operation (i.e. the voltage values are chosen such that if the crosspoint device is based on the FET-resistor stack, those FETs operate in the linear-regime (as opposed to saturation-regime)). On the other hand, during the update phase, the second set of wires in the crossbar array 800 is used.

[0072] Each of the single bit counters from the counter 920 corresponds to a respective resistive circuit from the resistor device 930. It should be noted that in the example depicted in FIG. 7, the counter 920 includes four bits mapped respectively with a stack of four resistive circuits from the resistor device 930. However, in other examples, the RPU can include a different number of single counter bits and resistive circuits than those depicted herein.

[0073] The resistance/conductance provided by the resistor device 930 affects the current output by the RPU 820 based at least in part on the voltage applied to the crosspoint. In one or more examples, the stack of resistive circuits includes the resistive circuits being connected in series. Therefore, the resistance of the resistor device 930 is a sum of the resistance of each of the resistive circuits that is activated. A resistive circuit from the resistor device 930 is activated based at least in part on a state of the corresponding single bit counter from the counter 920. For example, if a single bit counter is set (= 1/ON), the corresponding resistive circuit is activated, and if the single bit counter is not set (=0/OFF), the corresponding resistive circuit is not activated; or vice versa.

[0074] Further, in order to create a series of resistive states, the resistive circuits from the resistor device 930

have specific constraints. For example, the resistive circuits have quadratically increasing resistance as depicted in TABLE 1, where each successive resistive circuit has a resistance/conductance that is twice that of a previous resistive circuit in the stack. It should be noted that in other examples, the resistive circuits can have a different predetermined relationship than the example herein.

Table 1

FET	Resistance
T1	1x
T2	2x
T3	4x
T4	8x

[0075] FIG. 8 depicts adjustable resistance of the resistor device based at least in part on the states of the counter according to an embodiment of the invention. Consider that $R1_{hi}$ = Off-state resistance of T1 (e.g. 100K Ω) and $R1_{low}$ = On-state resistance of T1 (e.g. 10K Ω). The resistances of the other resistive circuits T2, T3, and T4, can be determined based at least in part on the relationship between the resistive circuits, for example the quadratic relationship (Table 1). Thus, $R2_{hi} = 2R1_{hi}$; $R3_{hi} = 4R1_{hi}$; and $R4_{hi} = 8R1_{hi}$. Accordingly, if the counter 920 is set to 1, that is, only T1 is set to ON, the total resistance of the resistor device 930 is $R1_{low} + R2_{hi} + R3_{hi} + R4_{hi} = 14R1_{hi} + R1_{low}$. Other configurations of varying the resistance of the resistor device 930 according to the value stored in the counter 920 are depicted in FIG. 8. The resistance of the resistor device 930 thus linearly varies in steps of ($R_{hi} - R_{low}$).

[0076] It should be noted that the relationship between the resistive circuits of the resistor device 930 can be varied to vary the linear step by which the resistor device 930 is adjusted according to the counter value 920. For example, if the resistive circuits used in the resistor device are FETs, the relationship can be achieved by various combinations of width, length, threshold voltage (Vt), and/or bias voltage, of the FETs.

[0077] FIG. 9 depicts varying the resistance/conductance of FETs based at least in part on the width according to an embodiment of the invention. The FETs T1, T2, T3, and T4, have source 962 and drain 968 of varying widths w1, w2, w3, and w4, respectively with gate 965 having substantially the same gate length L. The widths w1, w2, w3, and w4, are varied according to the relationship to vary the resistance/conductance of the resistor device 930.

For example, the width w1 is twice that of w2, w2 is twice of w3, and w3 is twice of w4, where w4 is a predetermined transistor width assigned to T4. In one or more examples, the width is varied by changing the number of fins in case the FETs used as the resistive circuits are FinFETs. For example, table 2 depicts a 4-bit example of varying the resistance/conductance of the resistive circuits from the resistor device 930 by varying the number of fins of the FinFETs used as the resistive circuits.

Table 2

FET	# FINs	Resistance
T1	8	1x
T2	4	2x
T3	2	4x
T4	1	8x

[0078] FIG. 10 depicts varying the resistance/conductance of FETs based at least in part on the gate length according to an embodiment of the invention. In this case, the FETs T1, T2, T3, and T4, have source 962 and drain 968 of a common width w, and gates 965 having different gate lengths L1, L2, L3, and L4, respectively. The lengths L1, L2, L3, and L4, are varied according to the relationship to vary the resistance/conductance of the resistor device 930. For example, the length L4 is twice that of L3, L3 is twice of L2, and L2 is twice of L1, where L1 is a predetermined transistor gate length assigned to L1.

[0079] FIG. 11 depicts an example of the resistor device 930 with variable resistance/conductance using vertical geometry according to an embodiment of the invention. Here, the stack FETs are fabricated in a vertical geometry. The fabrication is performed on a substrate layer. The substrate can be a silicon-on-insulator wafer that includes a top layer or silicon layer on top of an insulating layer such as a BOX (buried oxide) layer that can include silicon-dioxide. The substrate layer can be an N+ silicon substrate layer or any other such material.

[0080] A spacer layer 1010 is used to separate the various FETs in the stack by depositing the spacer layer 1010 between gate fill layers 1020 for the gates of the various FETs. For example, the spacer 1010 can be a SiNx layer and

[0081] The gate fill 1020 can be composed of any type of electrically conductive material such as those

selected from the group consisting of aluminum, cobalt, copper, gold, hafnium, nickel, palladium, platinum, molybdenum, niobium, polysilicon, rhodium, silver, tantalum, tantalum nitride, tin, titanium nitride, titanium tungsten, tungsten, vanadium, zirconium, an electrically conductive polymer, and mixtures thereof.

[0082] Further, a gate dielectric 1030 is used to separate the gate fill layer 1020 from the spacer as well as from a transistor channel 1040. The gate-dielectric 1030 can be composed of any known dielectric material such as those selected from the group consisting of aluminum arsenide, aluminum gallium arsenide, aluminum nitride, aluminum oxide, barium fluoride, barium strontium titanium oxide, barium titanium oxide, calcium fluoride, cerium fluoride, cerium oxide, gallium arsenide, hafnium aluminate, hafnium oxide, hafnium oxynitride, hafnium silicon oxide, hafnium silicon oxynitride, indium oxide, indium gallium nitride, indium phosphide, lanthanum oxide, lanthanum aluminum oxide, lanthanum silicon oxides, lead scandium tantalum oxide, lead zinc niobate, magnesium fluoride, niobium fluoride, niobium oxide, silicon nitride, silicon oxide, strontium bismuth tantalite, strontium titanium oxide, zirconium oxide, zirconium silicon oxide, titanium oxide, titanium dioxide, tantalum dioxide, tantalum pentoxide, tin oxide, zirconium aluminate, zirconium oxide, zirconium oxynitride, zirconium silicon oxynitride, yttrium oxide, yttrium silicon oxides, and admixtures thereof. In at least one example the gate-dielectric 90 exhibits a dielectric constant greater than that of silicon oxide.

[0083] A gate channel 1040 can be doped to provide particular electrical and performance characteristics. In one or more examples, the channel 1040 can be a polysilicon material surrounded by the gate electrodes 1020. Further yet, the vertical FET stack includes the terminals 1050 that provide the Read-1 901 and Read-2 902 terminals for the resistor device 930.

[0084] The gate dielectric 1020 deposition is spaced according to the varied lengths of the gates for the FETs T1, T2, T3, and T4, to create the quadratic (or any other predetermined) relationship between the gate lengths, and hence the resistance/conductance. Thus, the resistor device 930 can be a vertical FET stack with varying gate lengths.

[0085] Referring back to FIG. 7, the counter 920 is updated to store the weight at the crosspoint of the RPU array 800 at a clock edge that is generated at a common incidence of x and δ, thus enabling stochastic update of the counter 920. The x and δ are the voltages that are applied to the rows and columns of the RPU array 800 and include stochastic sequences of the voltages. The x corresponds to the stochastic pulse signal generated by the input to the cross-point device. In one or more examples, x is a stochastic sequence based on the feed-forward output of a given layer, and the δ is a stochastic sequence generated based on the error computed by the subsequent layer of the crossbar array of RPU devices 820.

[0086] Accordingly, when both stochastic sequences x and δ have a common waveform, a clock edge is generated and provided to the counter by a logic gate 940. In one or more examples, the logic gate 940 is an AND

gate that receives the x and δ sequences as input and provides the clock (CLK) as output. Other logic gates can be used in other examples.

[0087] The counter 920 receives, in addition to the CLK as an input, a second input that is indicative of a sign of the δ. Based on the sign, the counter 920 either increments (sign = +) or decrements (sign = -) the count value stored in the counter 920. Based on the increment/decrement, the bits in the single bit counters of the counter 920 change and in turn, the resistive circuits of the resistor device 930 are activated/deactivated to adjust the resistance/conductance of the resistor device 930.

[0088] FIG. 12 depicts a counter-based RPU that can be used as a crosspoint device in an RPU array according to an embodiment of the invention. Here, the RPU 820B, compared to the RPU 820 with a stack of FETs used in the resistor device 930 (FIG. 7), uses a resistor device 930 that includes a resistor ladder as the transduction element. In the example depicted, a 4-bit resistor ladder is shown, however, it should be noted that in other examples a different number of resistive circuits can be used in the resistor ladder to support a different number of bits in the counter 920. The resistor ladder shown includes four resistive circuits associated with corresponding single bit counters respectively of the counter 920.

[0089] Here, when input Ki is pulsed high, the current flowing out from the ladder circuit Ci is proportional to the value (i.e. weight) of the counter 920 (represented by the single bit counters <Q1:Q4>). Subsequently, during forward-propagation the output currents are accumulated column-wise by the RPU array 800. Further, during back-propagation, the output currents are accumulated row-wise by the RPU array 800.

[0090] For the 4-bit example depicted, consider that the voltages at the output nodes of the AND gates 1210 are <V1:V4> (either VL = 0V or VH = 1V, for example). The current flowing out from the circuit Ci into the column of the RPU array 800 is:

$$C_i = \frac{1}{2R}\left(\frac{1}{2^3}V_1 + \frac{1}{2^2}V_2 + \frac{1}{2^1}V_3 + \frac{1}{2^0}V_4\right)$$

[0091] More generally, for the case of r-bits, current flowing out is:

$$C_i = \frac{1}{2R}\left(\frac{1}{2^{r-1}}V_1 + \frac{1}{2^{r-2}}V_2 + \cdots + \frac{1}{2^1}V_{r-1} + \frac{1}{2^0}V_r\right)$$

[0092] Thus, the current is proportional to the digital value of the counter as exemplified in Table 3 for the 4-bit case, where the current varies linearly in steps of 1/16R, where R is a predetermined resistance value.

Table 3

Bit states for V4, V3, V2, V1	**Current flowing out, C_i**
0, 0, 0, 0	0
0, 0, 0, 1	1/16R
0, 0, 1, 0	2/16R
0, 0, 1, 1	3/16R
. . .	. . .
1, 1, 1, 0	14/16R
1, 1, 1, 1	15/16R

[0093] FIG. 13 depicts how the counter-based RPU 820B with resistor ladder is connected as a crosspoint device in the RPU array 800 according to an embodiment of the invention. Connecting the RPU 820B in the RPU array 800 uses a connection circuit 1310. The connection circuit 1310 includes four transistors Tr1 1312, Tr2 1314, Tr3 1316, and Tr4 1318, as depicted and described further. The transistors Tr2 and Tr4 have resistances $<< R/2^r$ where r = number of bits used in the counter 920.

[0094] FIG. 14 depicts a forward propagation pass performed using the RPU 820B and the connection circuit 1310 according to an embodiment of the invention. During forward-propagation, the two FWD transistors (Tr1 1312 and Tr2 1314) at each cross-point are turned ON, and the two BWD transistors (Tr3 1316 and Tr4 1318) are turned OFF.

[0095] FIG. 15 depicts a backward propagation pass performed using the RPU 820B and the connection circuit 1310 according to an embodiment of the invention. During back-propagation, two FWD transistors (Tr1 1312 and Tr2 1314) at each cross-point are turned OFF, and the two BWD transistors (Tr3 1316 and Tr4 1318) are turned ON.

[0096] In both the forward and backward propagation, using the RPU 820B the operation of the RPU array 800 continues as described earlier, with current accumulated column-wise for the forward propagation and row-wise for the backward propagation.

[0097] FIG. 16 depicts an RPU array using the counter-based RPU devices according to an embodiment of the

invention. The RPU array 800 includes the counter-based RPUs 820 at each crosspoint. The RPU devices 820 can use the FET-based or the resistor-ladder-based resistor device 930 as described herein. The RPU array 800 further includes, at each crosspoint, a local memory 1610 and a local controller 1620.

[0098] The local memory 1610 at a specific crosspoint stores multiple weights assigned to the specific crosspoint by each of the one or more ANN matrices, each matrix corresponding to the connection between two adjacent layers of the ANN (For example, matrix1 connects Layer-A to Layer-B, and matrix2 connects Layer-B to Layer-C, and so forth). The local controller 1620 receives an instruction from a global controller 1640 to load the corresponding weights into the RPU devices 820 based on the layers of the ANN being processed. Accordingly, the local controller 1620 accesses the corresponding weight value from the local memory 1610 and loads the weight value into the counter 920 of the RPU device 820 at the specific crosspoint.

[0099] Further, the global controller 1640 instructs the local controller 1620 of a change in the counter value after a weight update operation. In one or more examples, in response, the local controller 1620 reads the updated weight value from the counter 920 and stores the updated value into the local memory 1610 for subsequent use.

[0100] In one or more examples, a peripheral memory 1650 stores information used by the global controller 1640, for example, parameter values indicative of the processing of the ANN, such as which layer is being processed, among others.

[0101] FIG. 17 depicts a dataflow in an RPU array using the counter-based RPU devices according to N embodiment of the invention. The RPU array 800 includes the counter-based RPUs 820 at each crosspoint. The RPU devices 820 can use the FET-based or the resistor-ladder-based resistor device 930 as described herein. The weight elements of the individual cells of the RPU array 800 are stored locally at the cell-level in the local memory 1610 for efficient data transfer. All the different weights that can be assigned to a particular cell (e.g. crosspoint (m, n)) of each ANN weight matrix (e.g. matrix1 between A-B layers, matrix2 between B-C layers, etc.) is stored in the local memory 1610. The weight is loaded into the cell during respective layer operations (i.e. forward propagation, back propagation, and weight update). Further, the value of the cell is stored back into the local memory 1610 after the weight update step.

[0102] As depicted, the local controller 1620 receives an indication/instruction from the global controller 1640 that a particular layer (e.g. A-B) is being processed. Accordingly, the local controller 1620 for a specific crosspoint (m, n) reads the weight value for the cell (m, n) that is stored in the local memory 1610, and updates the counter 920 of the RPU device 820 at the crosspoint (m, n) of the RPU array 800. As described herein, the counter value adjusts the resistance/conductance of the resistor device 930 of the RPU device 820 at the crosspoint (m, n). Accordingly, when a voltage sequence (a_n) is applied to the RPU array 800 at the row *n*, the RPU device 820 passes a current I_m as output based at least in part on the adjusted resistance/conductance according to the

counter value. The currents across the multiple RPU devices 820 in the column are accumulated in forward propagation to provide input value to the subsequent layer. Alternatively, in backward propagation, the current is accumulated across the rows.

[0103] Thus, the RPU array 800 facilitates implementing multiple layers of an ANN using adjustable resistance/conductance of the RPU device 820, the resistance/conductance being adjustable using a counter 920 that activates/deactivates corresponding resistive circuits in the resistor device 930 of the RPU device 820.

[0104] The embodiments of the invention described herein improve typical weight storage elements used in RPU arrays by eliminating charge-loss, and stringent requirements on update currents, which are typically observed when using capacitor based weight storage elements. The embodiments of the invention described herein facilitate using a counter-based RPU device as the weight storage element where the counter bits control gate terminals of FETs connected in a series such that the total resistance of the FET stack is used as the transduction element (i.e. resistor element of each RPU cell). Alternatively, the embodiments of the invention described herein facilitate using a counter-based RPU device in which a resistor ladder is used as the transduction element based at least in part on the counter bits.

13 04 22

[0105] The descriptions of the various embodiments of the invention have been presented for purposes of illustration, but are not intended to be exhaustive or limited to the embodiments of the invention described. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from scope of the invention. The terminology used herein was chosen to best explain the principles of the embodiments of the invention, the practical application, or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments of the invention described herein.

CLAIMS

1. A method for storing weight in a crosspoint device of a resistive processing unit (RPU) array, the method comprising:

reading, from a local memory, a current state of each single bit counter of a plurality of single bit digital counters in the crosspoint device;

updating the current state of each single bit counter from the plurality of single bit counters in the crosspoint device, the updated states of the single bit counters representing the weight to be stored at the crosspoint device, and the crosspoint device being in operable communication with an input layer and a hidden layer of the neural network;

adjusting electrical conductance of a resistor device of the crosspoint device, wherein the resistor device comprises a plurality of resistive circuits, wherein each resistive circuit is associated with a respective single bit counter from the plurality of single bit counters, wherein the electrical conductance is adjusted by activating or deactivating each resistive circuit according to a state of the associated single bit counter; and

updating the local memory to indicate an updated state of each single bit counter of the plurality of single bit counters.

13 04 22

2. The method of claim 1, wherein the resistive circuits are communicatively coupled in series.

3. The method of claim 1, wherein the resistive circuits include a quadratically increasing resistance, a first resistive circuit having a predetermined resistance and each further successive resistive circuit includes a resistance that is twice the value of a previous resistive circuit.

4. The method of claim 3, wherein the resistive circuits comprise field effect transistors (FETs).

5. The method of claim 4, wherein the FETs include quadratically increasing resistance based at least in part on quadratically decreasing transistor width.

6. The method of claim 4, wherein the FETs include quadratically increasing resistance based at least in part on quadratically increasing gate length.

7. The method of claim 6, wherein the FETs are part of a vertical stack of FETs.

8. The method of claim 3, wherein the resistive circuits comprise a resistor ladder, each of the resistive circuit comprising:

a logic gate; and

a series of resistors; and

wherein, the logic gate is activated based at least in part on the state of the corresponding single bit counter.

9. A crosspoint device for storing a weight value at a crosspoint in a resistive processing unit (RPU) array, the crosspoint device comprising:

a local memory for storing a current weight for each single bit digital counter of a plurality of single bit counters in the crosspoint device, wherein the weight corresponds to a layer of the neural network;

a counter comprising a plurality of single bit counters, states of the single bit counters representing the weight to be stored at the crosspoint device; and

a resistor device comprising a plurality of resistive circuits, each resistive circuit associated with a respective single bit counter from the plurality of single bit counters, the resistive circuits for being activated or deactivated according to a state of the associated single bit counter, and an electrical conductance of the resistor device for being adjusted based at least in part on the resistive circuits that are activated.

10. The crosspoint device of claim 9, wherein the resistive circuits are connected in series.

11. The crosspoint device of claim 9, wherein the resistive circuits include a quadratically increasing resistance, a first resistive circuit having a predetermined resistance and each further successive resistive circuit having a resistance that is twice of a previous resistive circuit.

12. The crosspoint device of claim 11, wherein the resistive circuits are field effect transistors (FETs).

13. The crosspoint device of claim 12, wherein the FETs include quadratically increasing resistance based at least in part on quadratically decreasing transistor width.

14. The crosspoint device of claim 12, wherein the FETs include quadratically increasing resistance based at least in part on quadratically increasing gate length.

15. The crosspoint device of claim 11, wherein the resistive circuits are a resistor ladder, each of the resistive circuit comprising:

a logic gate; and

a series of resistors; and

wherein, the logic gate is activated based at least in part on the state of the corresponding single bit counter.

13 04 22