

**NEURAL PROCESSING CORE, METHOD OF PROGRAMMING A SYNAPTIC  
MEMORY ARRAY THEREOF AND METHOD OF PERFORMING NEURAL  
NETWORK INFERENCE THEREON**

5

**TECHNICAL FIELD**

**[0001]** The present invention generally relates to a neural processing core configured to perform neural network computations for a neural network, a method of programming a synaptic memory array of the neural processing core, and a method of performing neural network inference on the neural processing core.

10

**BACKGROUND**

**[0002]** In relation to the implementation of analog neuromorphic computing for neural networks, one of the important attributes of neuromorphic computing is non-von-Neumann architecture, where storage (e.g., for synaptic weights) and computation (e.g., of membrane potential) are unified as opposed to loading the data (e.g., synaptic weights) from a centrally defined storage and then passing them through a centralized data link (the von-Neumann bottleneck) to the algorithmic logic units (ALUs) for computation. A highly efficient approach towards achieving this goal is via analog computing, with one of the most common forms being analog current summing of synaptic memory cells for a given neuron, where each synaptic memory cell (which may simply be referred to herein as a memory cell) is typically represented by a memristor cell (e.g., a programmable resistor). The change of membrane potential in a neuron (e.g., denoted as  $\Delta P$ ), whether in a spiking neuron (which is more biologically inspired), or an artificial neuron (used in deep learning), typically follow Equation (1) below, as a sum of axon inputs weighted by its synapses (which may simply be referred to as a weighted sum):

20

25

$$\Delta P = \sum_{i=0}^{n-1} w_i * x_i$$

(Equation 1)

**[0003]** In Equation (1),  $n$  denotes the number of (incoming) synapses for the neuron,  $w_i$  denotes the weight for the  $i$ -th synapse, and  $x_i$  denotes the value of  $i$ -th input axon. For spiking neuron, input axon generally takes on binary values of 0 or 1 (though in some cases, may be -1 or 1). For an artificial neuron, input axon may take on a floating-point or multi-bit integer number. After adding the membrane potential with  $\Delta P$ , in a spiking neuron, the membrane

30

potential is subject to a certain leakage model (e.g., an exponential leak), and then compared against a firing threshold. If the membrane potential is above the firing threshold, the neuron will fire and emit a spike (e.g., of 1) to the next layer and reset its membrane potential to a preset value (e.g., 0). If the membrane potential is below the threshold, the neuron will wait for the next cycle of axon inputs on its synapses, and the above-described process starts all over again. In an artificial neuron, the membrane potential always start as 0, and the neuron always fire and emit a value that is a pre-determined function (referred to as an activation function) of  $\Delta P$ , to the next layer. In this regard, the thresholding semantics in a spiking neuron may also be viewed as an activation function which outputs a 1 if input is above the threshold, and outputs a 0 (which is generally not passed onto the next layer) otherwise.

**[0004]** FIG. 1 illustrates an example analog computing in neural network inference with analog current summing based on a synaptic memory array 100 (which may also be referred to as a synaptic memory matrix, or simply, a memory matrix). When a neural network layer is mapped onto a memory matrix 100, such as shown in FIG. 1, if an input activation voltage  $v_i = Act_i$  is applied on input axon  $i$ , and each column corresponds to a neuron, then all columns (neurons) in the memory matrix 100 may compute simultaneously by draining each column's current simultaneously, allowing very fast computation. Accordingly, both the energy efficiency and speed of computing the weighted sum in such an analog computing mode may be significantly better than the digital computing approach, which would require reading each synapse's weight value individually and then performing multiply-and-accumulate (MAC) or synaptic operation (SOP) in the digital domain. Note that for spiking neural network (SNN) / SOP, a synapse need not be read if the input axon value is 0.

**[0005]** One of the limitations in analog computing is the achievable precision. For example, if a memory cell (e.g., memristor cell) can only express 4-bit (i.e., 16 linear states), then the synaptic weight can only have a 4-bit precision, and this may limit the accuracy achievable in the neural network. Although there have been disclosed conventional techniques for "concatenating" multiple memristor cells to form a higher precision synapse, such conventional techniques only work in the digital computing mode, where the cell values are read out first and then digitally concatenated. In particular, various embodiments of the present invention note that, in analog computing mode, the above-mentioned conventional techniques of "concatenating" two or more memristor cells are generally impractical or ineffective, because the error in the highest-order-bit(s) (which may also be referred to as the most-significant-bit(s) or MSB(s)) cell will be dominant and it will be essentially useless to add any low-order-bit(s)

cell(s) (e.g., the last lowest-order-bit(s) cell may be referred to as the least-significant-bit(s) or LSB(s) cell). In other words, various embodiments of the present invention note that although the above-mentioned conventional techniques may attempt to increase the number of bits for representing a synaptic weight by “concatenating” two or more memristor cells, the weight precision (or synapse precision) achieved by such conventional techniques may not be improved (or increased), or at least not improved effectively.

**[0006]** A need therefore exists to provide a method of programming a synaptic memory array of a neural processing core, the corresponding neural processing core and a method of performing neural network inference thereon, that seek to overcome, or at least ameliorate, one or more of deficiencies associated with conventional methods of programming a synaptic memory array of a neural processing core and the corresponding conventional neural processing cores, and more particularly, effectively improving (or increasing) synapse precision (or weight precision) of the synaptic memory array (or synaptic memory cells therein) of the neural processing core in representing synaptic weights for implementing analog neuromorphic computing for neural networks. It is against this background that the present invention has been developed.

### SUMMARY

**[0007]** According to a first aspect of the present invention, there is provided a method of programming a synaptic memory array of a neural processing core configured to perform neural network computations for a neural network, the method comprising:

writing an intended synaptic weight value for a group of synaptic memory cells of the synaptic memory array to a first synaptic memory cell of the group of synaptic memory cells of the synaptic memory array;

obtaining a first residual synaptic weight value associated with the group of synaptic memory cells based on the intended synaptic weight value and an obtained first synaptic weight value of the group of synaptic memory cells, the obtained first synaptic weight value being obtained based on a written state value of the first synaptic memory cell from said writing the intended synaptic weight value to the first synaptic memory cell; and

writing a first intended residual synaptic weight value determined based on the obtained first residual synaptic weight value to a second synaptic memory cell of the group of synaptic memory cells, the second synaptic memory cell being immediately succeeding the first synaptic memory cell in the group of synaptic memory cells, wherein

the group of synaptic memory cells collectively constitutes a combined synaptic memory cell representing a synaptic weight of a corresponding synapse of the neural network, and

5 the group of synaptic memory cells corresponds to a plurality of ordered bits that are ordered from highest order bit(s) to lowest order bit(s), the first synaptic memory cell corresponding to the highest order bit(s).

**[0008]** According to a second aspect of the present invention, there is provided a neural processing core configured to perform neural network computations for a neural network, the neural processing core comprising:

10 a synaptic memory array configured to perform in-memory matrix computations;  
a plurality of input axons coupled to the synaptic memory array and configured to receive a plurality of axon inputs, respectively, to the synaptic memory array; and

a plurality of output neurons coupled to the synaptic memory array and configured to output a plurality of neuron outputs, respectively, from the synaptic memory array, wherein

15 a first synaptic memory cell of a group of synaptic memory cells of the synaptic memory array has a written state value based on an intended synaptic weight value for the group of synaptic memory cells of the synaptic memory array written to the first synaptic memory cell,

a second synaptic memory cell of the group of synaptic memory cells has a written state value based on a first intended residual synaptic weight value written to the second synaptic memory cell, the second synaptic memory cell being immediately succeeding the first synaptic memory cell in the group of synaptic memory cells, and the first intended residual synaptic weight value being determined based on a first residual synaptic weight value associated with the group of synaptic memory cells,

20 the first residual synaptic weight value is obtained based on the intended synaptic weight value and an obtained first synaptic weight value of the group of synaptic memory cells, the obtained first synaptic weight value being obtained based on the written state value of the first synaptic memory cell from the intended synaptic weight value written to the first synaptic memory cell,

25 the group of synaptic memory cells collectively constitutes a combined synaptic memory cell representing a synaptic weight of a corresponding synapse of the neural network, and

the group of synaptic memory cells corresponds to a plurality of ordered bits that are ordered from highest order bit(s) to lowest order bit(s), the first synaptic memory cell corresponding to the highest order bit(s).

[0009] According to a third aspect of the present invention, there is provided a method of performing neural network inference on the neural processing core according to the above-mentioned second aspect of the present invention configured to perform neural network computations for a neural network, the method comprising:

for each memory cell column of the synaptic memory array from a first memory cell column of the synaptic memory array including the first synaptic memory cell to a last memory cell column of the synaptic memory array including a last synaptic memory cell of the group of synaptic memory cells, the first memory cell column to the last memory cell column forming a group of memory cell columns:

sensing analog current of the memory cell column to obtain a summed analog current of the memory cell column; and

digitizing the summed analog current of the memory cell column to obtain a digitized summed current value of the memory cell column; and

summing the digitized summed current values obtained for the group of memory cell columns to obtain a total digitized summed current value of the group of memory cell columns, the group of memory cell columns corresponding to a neuron of the synaptic memory array.

## BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Embodiments of the present invention will be better understood and readily apparent to one of ordinary skill in the art from the following written description, by way of example only, and in conjunction with the drawings, in which:

FIG. 1 depicts an example analog computing in neural network inference with analog current summing based on a synaptic memory array;

FIG. 2 depicts a schematic flow diagram of a method of programming a synaptic memory array of a neural processing core configured to perform neural network computations for a neural network, according to various embodiments of the present invention;

FIG. 3 depicts a schematic drawing of a neural processing core configured to perform neural network computations for a neural network, according to various embodiments of the present invention;

FIG. 4 depicts a schematic flow diagram of a method of performing neural network inference on a neural processing core configured to perform neural network computations for a neural network, according to various embodiments of the present invention;

FIG. 5A depicts a schematic drawing of a neural network processor system, according to various embodiments of the present invention;

FIG. 5B depicts a block diagram of an example configuration of the neural processing core, according to various example embodiments of the present invention;

FIG. 6 depicts a schematic block diagram of an exemplary computer system in which a neural network processor system, according to various embodiments of the present invention, may be realized or implemented;

FIGs. 7A to 7C depict schematic drawings of three example arrangements of synaptic memory cells, along with an illustration of the associated errors or noise;

FIG. 8 depicts a schematic drawing of a native 4-bit cell for illustrating an example of how a 4-bit state may be represented using only two 2-bit RRAM cells, according to various example embodiments of the present invention;

FIG. 9 depicts a table (Table 1) showing cell-stitching examples described with reference to FIG. 8;

FIG. 10 depicts a table (Table 2) showing examples of representing signed 4-bit using 2-bit cells with cell-stitching, according to various example embodiments of the present invention;

FIGs. 11A and 11B depict schematic flow diagrams illustrating processing steps for implementing the above-described cell-stitching during synapse programming and during inference, respectively, according to various example embodiments of the present invention;

FIG. 12 depicts a schematic drawing illustrating an example implementation or architecture of cell-stitching during inference, according to various example embodiments of the present invention;

FIG. 13 depicts an example analog current subtraction circuit for subtracting the reference cell column current sum from the main cell column in analog domain, according to various example embodiments of the present invention;

FIG. 14 depicts an example analog current subtraction circuit with selectable feedback resistors allowing selectable analog amplification factor of the TIA, according to various example embodiments of the present invention; and

FIGs. 15A to 15D show evaluation results of cell-stitching on AlexNet DoReFa for DoReFa 8,8,8, DoReFa 6,6,32, DoReFa 5,5,32 and DoReFa 4,4,32, respectively, according to various example embodiments of the present invention.

5

## DETAILED DESCRIPTION

[0011] Various embodiments of the present invention provide a method of programming a synaptic memory array of a neural processing core configured to perform neural network computations for a neural network, the corresponding neural processing core (i.e., a neural processing core programmed according the above-mentioned method of programming) and a method of performing neural network inference thereon.

[0012] For example, as described in the background, although conventional techniques may attempt to increase the number of bits for representing a synaptic weight by “concatenating” two or more synaptic memory cells, such conventional techniques only work in the digital computing mode, where the cell values are read out first and then digitally concatenated. In this regard, various embodiments of the present invention note that because the states in a synaptic memory cell (e.g., a memristor cell) are rarely a quantum phenomenon, these states generally are not quantum- or quantization-precise. For example, when programming a state value of “5” (which corresponds to a certain level of conductance) in a 4-bit memristor cell, the actual value after programming may be, for example, “5.2”, “4.7”, and so on, with the difference from “5” or the intended value (i.e., corresponding to the intended level of conductance) appearing as error to the intended value. In this regard, various embodiments of the present invention note that, in analog computing mode, the above-mentioned conventional techniques of “concatenating” two or more memristor cells are generally impractical or ineffective, because the error in the highest-order-bit(s) (which may also be referred to as the most-significant-bit(s) or MSB(s)) cell will be dominant and it will be essentially useless to add any low-order-bit(s) cell(s) (e.g., the last lowest-order-bit(s) cell may be referred to as the least-significant-bit(s) or LSBs cell). Therefore, the weight precision (or synapse precision) of a synaptic memory array in representing synaptic weights achieved by such conventional techniques may not be improved (or increased), or at least not improved effectively. Accordingly, various embodiments of the present invention provide a method of programming a synaptic memory array of a neural processing core, the corresponding neural processing core and a method of performing neural network inference thereon, that seek to overcome, or at least ameliorate, one or more of deficiencies associated with conventional methods of programming a synaptic

memory array of a neural processing core and the corresponding conventional neural processing cores, and more particularly, effectively improving (or increasing) synapse precision (or weight precision) of the synaptic memory array (or synaptic memory cells therein) of the neural processing core in representing synaptic weights for implementing analog neuromorphic computing for neural networks.

**[0013]** For simplicity and clarity, the method of programming a synaptic memory array according to various embodiments of the present invention may be described with respect to one neural processing core. In this regard, it will be appreciated that any one or more neural processing cores of one or more neural processing units of a neural network processor system may be programmed according to the method of programming in the same or similar manner as desired or as appropriate. Furthermore, in various embodiments, each synaptic memory cell may be a single-bit (1-bit) cell or a multi-bit cell (e.g., 2-bit cell, 3-bit cell, 4-bit cell, and so on). For example, in the case of a synaptic memory cell being a single-bit cell, such a synaptic memory cell having the highest order may thus represent a highest-order-bit or MSB, and on the other hand, such a synaptic memory cell having the lowest order may thus represent a lowest-order-bit or LSB. In the case of a synaptic memory cell being a multi-bit cell, such a synaptic memory cell having the highest order may thus represent highest-order-bits or MSBs, and on the other hand, such a synaptic memory cell having the lowest order may thus represent lowest-order-bits or LSBs. Accordingly, more generally, terms such as highest-order-bit(s) (or MSB(s)) or lowest-order-bit(s) (or LSB(s)) cover both single-bit and multi-bit cells. In addition, in the case of a multi-bit cell, although a multi-bit cell having a number of supported states being a power of 2 may be preferred according to various embodiments, the present invention is not limited to such a number of supported states, and the number of supported states may instead be other than a power of 2, such as but not limited to, a 5-state cell configured to support 5 states. Accordingly, the use of terms herein, such as MSBs and LSBs, do not limit the multi-bit cell to having a number of supported states being a power of 2, and may more generally be referred to as a multi-state cell.

**[0014]** FIG. 2 depicts a schematic flow diagram of a method 200 of programming a synaptic memory array of a neural processing core configured to perform neural network computations for a neural network, according to various embodiments of the present invention. The method 200 comprises: writing (at 202) an intended synaptic weight value for a group of synaptic memory cells of the synaptic memory array to a first synaptic memory cell of the group of synaptic memory cells of the synaptic memory array; obtaining (at 204) a first residual synaptic

weight value associated with the group of synaptic memory cells based on the intended synaptic weight value and an obtained first synaptic weight value of the group of synaptic memory cells, the obtained first synaptic weight value being obtained based on a written state value of the first synaptic memory cell from the above-mentioned writing (at 202) the intended synaptic weight value to the first synaptic memory cell; and writing (at 206) a first intended residual synaptic weight value determined based on the obtained first residual synaptic weight value to a second synaptic memory cell of the group of synaptic memory cells, the second synaptic memory cell being immediately succeeding the first synaptic memory cell in the group of synaptic memory cells. In this regard, the group of synaptic memory cells collectively constitutes a combined synaptic memory cell representing a synaptic weight of a corresponding synapse of the neural network. Furthermore, the group of synaptic memory cells corresponds to a plurality of ordered bits that are ordered from highest order bit(s) to lowest order bit(s), the first synaptic memory cell corresponding to the highest order bit(s). For example, in the case of each synaptic memory cell in the group of synaptic memory cells being a multi-bit cell, each synaptic memory cell may then represent corresponding ordered bits (e.g., the first synaptic memory cell (e.g., a 2-bit cell) corresponds to the highest order bits (e.g., highest ordered 2-bits)) of the plurality of ordered bits. In the case of each synaptic memory cell in the group of synaptic memory cells being a single-bit cell, each synaptic memory cell may then represent a corresponding ordered bit (e.g., the first synaptic memory cell (e.g., a 1-bit cell) corresponds to the highest order bit (e.g., highest ordered 1-bit)) of the plurality of ordered bits.

**[0015]** Accordingly, the method 200 of programming a synaptic memory array of a neural processing core advantageously uses a lower-order-bit(s) cell (i.e., a synaptic memory cell representing lower-order-bit(s)) to compensate for error in the synaptic weight value represented by higher-order-bit(s) cell(s) in the group of synaptic memory cells, which has been found to effectively improve (or increase) synapse precision (or weight precision) of the synaptic memory array (or synaptic memory cells therein) of the neural processing core in representing synaptic weights for implementing analog neuromorphic computing for neural networks. These advantages or technical effects, or other advantages or technical effects, will become more apparent to a person skilled in the art as the method 200 of programming a synaptic memory array, and the corresponding neural processing core and the corresponding method of performing neural network inference, are described in more details according to various embodiments and example embodiments of the present invention.

**[0016]** In various embodiments, the above-mentioned obtaining the first residual synaptic weight value comprises determining a difference between the intended synaptic weight value for the group of synaptic memory cells and the obtained first synaptic weight value of the group of synaptic memory cells.

- 5 **[0017]** In various embodiments, the method 200 further comprises obtaining a first synaptic weight value of the group of synaptic memory cells to produce the obtained first synaptic weight value of the group of synaptic memory cells, comprising: obtaining the written state value of the first synaptic memory cell, the first synaptic memory cell having associated therewith a corresponding scaling factor based on an order thereof in the group of synaptic memory cells;  
10 and multiplying the written state value of the first synaptic memory cell and the corresponding scaling factor to obtain a product thereof. In this regard, the obtained first synaptic weight value is obtained based on the above product obtained.

- [0018]** In various embodiments, the above-mentioned writing the first intended residual synaptic weight value to the second synaptic memory cell comprises: dividing the first intended  
15 residual synaptic weight value by a corresponding scaling factor associated with the second synaptic memory cell to obtain a scaled first intended residual synaptic weight value; and writing the scaled first intended residual synaptic weight value to the second synaptic memory cell.

- [0019]** In various embodiments, the intended synaptic weight value and/or the first intended  
20 residual synaptic weight value is a signed value. That is, at least one of the intended synaptic weight value and the first intended residual synaptic weight value is a signed value. In this regard, the intended synaptic weight value is written to the first synaptic memory cell with respect to a first intended reference value (e.g., in the case of the intended synaptic weight value being a signed value) and/or the first intended residual synaptic weight value is written to the  
25 second synaptic memory cell with respect to a second intended reference value (e.g., in the case of first intended residual synaptic weight being a signed value).

- [0020]** In various embodiments, the method 200 further comprises: writing the first intended reference value to a first reference memory cell communicatively coupled to the first synaptic memory cell; and/or writing the second intended reference value to a second reference  
30 memory cell communicatively coupled to the second synaptic memory cell.

**[0021]** In various embodiments, the first intended reference value is offset from a middle value of an intended state range of the first synaptic memory cell to compensate for write noise associated with the first synaptic memory cell, and/or the second intended reference value is

offset from a middle value of an intended state range of the second synaptic memory cell to compensate for write noise associated with the second synaptic memory cell.

5 [0022] In various embodiments, the group of synaptic memory cells comprises three or more synaptic memory cells, and for each synaptic memory cell from the second synaptic memory cell to a penultimate synaptic memory cell of the group of synaptic memory cells, the method further comprises: obtaining a second residual synaptic weight value associated with the group of synaptic memory cells based on the intended synaptic weight value and an obtained second synaptic weight value of the group of synaptic memory cells, the obtained second synaptic weight value being obtained based on written state values of the synaptic memory cell and each preceding synaptic memory cell of the group of synaptic memory cells with respect to the synaptic memory cell; and writing a second intended residual synaptic weight value determined based on the obtained second residual synaptic weight value to an immediately succeeding synaptic memory cell of the group of synaptic memory cells with respect to the synaptic memory cell.

15 [0023] In various embodiments, the method 200 further comprises, obtaining, for said each synaptic memory cell from the second synaptic memory cell to the penultimate synaptic memory cell of the group of synaptic memory cells, a second synaptic weight value of the group of synaptic memory cells to produce the obtained second synaptic weight value of the group of synaptic memory cells, comprising: obtaining the written state values of the synaptic memory cell and said each preceding synaptic memory cell of the group of synaptic memory cells with respect to the synaptic memory cell, each of the synaptic memory cell and said each preceding synaptic memory cell having associated therewith a corresponding scaling factor based on an order thereof in the group of synaptic memory cells; multiplying the obtained written state values of the synaptic memory cell and said each preceding synaptic memory cell with the corresponding scaling factors, respectively, to obtain a plurality of products thereof; and summing the plurality of products to obtain a summed value. In this regard, the obtained second synaptic weight value of the group of synaptic memory cells obtained for the synaptic memory cell is obtained based on the above summed value obtained.

25 [0024] FIG. 3 depicts a schematic drawing of a neural processing core 300 configured to perform neural network computations for a neural network, according to various embodiments of the present invention. The neural processing core 300 comprises: a synaptic memory array 304 configured to perform in-memory matrix computations; a plurality of input axons 308 coupled to the synaptic memory array 304 and configured to receive a plurality of axon inputs

312, respectively, to the synaptic memory array 304; and a plurality of output neurons 316 coupled to the synaptic memory array 304 and configured to output a plurality of neuron outputs 320, respectively, from the synaptic memory array 304. In various embodiments, a first synaptic memory cell (e.g., 324-1) of a group of synaptic memory cells (e.g., 324) of the synaptic memory array 304 has a written state value based on an intended synaptic weight value for the group of synaptic memory cells (e.g., 324) of the synaptic memory array 304 written to the first synaptic memory cell (e.g., 324-1). In addition, a second synaptic memory cell (e.g., 324-2) of the group of synaptic memory cells 324 has a written state value based on a first intended residual synaptic weight value written to the second synaptic memory cell (e.g., 324-2), the second synaptic memory cell (e.g., 324-2) being immediately succeeding the first synaptic memory cell (e.g., 324-1) in the group of synaptic memory cells (e.g., 324), and the first intended residual synaptic weight value being determined based on a first residual synaptic weight value associated with the group of synaptic memory cells (e.g., 324). In various embodiments, the first residual synaptic weight value is obtained based on the intended synaptic weight value and an obtained first synaptic weight value of the group of synaptic memory cells (e.g., 324), the obtained first synaptic weight value being obtained based on the written state value of the first synaptic memory cell from the intended synaptic weight value written to the first synaptic memory cell (e.g., 324-1). Accordingly, the group of synaptic memory cells (e.g., 324) collectively constitutes a combined synaptic memory cell representing a synaptic weight of a corresponding synapse of the neural network. Furthermore, the group of synaptic memory cells (e.g., 324) corresponds to a plurality of ordered bits that are ordered from highest order bit(s) to lowest order bit(s), the first synaptic memory cell corresponding to the highest order bit(s).

**[0025]** In various embodiments, the synaptic memory array 304 is programmed according to the method 200 of programming a synaptic memory array as described herein with reference to FIG. 2 according to various embodiments.

**[0026]** In various embodiments, the first residual synaptic weight value is obtained based on a difference between the intended synaptic weight value for the group of synaptic memory cells (e.g., 324) and the obtained first synaptic weight value of the group of synaptic memory cells (e.g., 324).

**[0027]** In various embodiments, the intended synaptic weight value and/or the first intended residual synaptic weight value is a signed value. In addition, the intended synaptic weight value is written to the first synaptic memory cell with respect to a first intended reference value and/or

the first intended residual synaptic weight value is written to the second synaptic memory cell with respect to a second intended reference value.

**[0028]** In various embodiments, the neural processing core 300 further comprises: a first reference memory cell communicatively coupled to the first synaptic memory cell (e.g., 324-1), the first reference memory cell having a first reference written state value based on the first intended reference value written thereto; and/or a second reference memory cell communicatively coupled to the second synaptic memory cell (e.g., 324-1), the second reference memory cell having a second reference written state value based on the second intended reference value written thereto.

**[0029]** In various embodiments, the first intended reference value is offset from a middle value of an intended state range of the first synaptic memory cell (e.g., 324-1) to compensate for write noise associated with the first synaptic memory cell (e.g., 324-1), and/or the second intended reference value is offset from a middle value of an intended state range of the second synaptic memory cell (e.g., 324-2) to compensate for write noise associated with the second synaptic memory cell (e.g., 324-2).

**[0030]** In various embodiments, the neural processing core further comprises one or more additional groups of synaptic memory cells of the synaptic memory array 304, each additional group of synaptic memory cells comprising a first synaptic memory cell and a second synaptic memory cell (e.g., configured or programmed in the same or similar manner as the first synaptic memory cell 324-1 and the second synaptic memory cell 324-2 described hereinbefore according to various embodiments). Accordingly, although FIG. 3 may only illustrate one group of synaptic memory cells 324 in the synaptic memory array 304, it will be appreciated by a person skilled in the art that additional groups of synaptic memory cells may be configured or programmed in the synaptic memory array 304 (in the same or similar manner as the first synaptic memory cell 324-1 and the second synaptic memory cell 324-2 described hereinbefore according to various embodiments) to represent synaptic weights of additional corresponding synapses, respectively, of the neural network, as desired or as appropriate.

**[0031]** In various embodiments, the first reference memory cell is further communicatively coupled to the first synaptic memory cell of the above-mentioned each additional group of synaptic memory cells, and/or the second reference memory cell is further communicatively coupled to the second synaptic memory cell of the above-mentioned each additional group of synaptic memory cells.

**[0032]** In various embodiments, the group of synaptic memory cells (e.g., 324) comprises three or more synaptic memory cells, and for each synaptic memory cell from the second synaptic memory cell to a penultimate synaptic memory cell of the group of synaptic memory cells, an immediately succeeding synaptic memory cell of the group of synaptic memory cells (e.g., 324) with respect to the synaptic memory cell has a written state value based on a second intended residual synaptic weight value written to the immediately succeeding synaptic memory cell, the second intended residual synaptic weight value being determined based on a second residual synaptic weight value associated with the group of synaptic memory cells (e.g., 324), and the second residual synaptic weight value is obtained based on the intended synaptic weight value and an obtained second synaptic weight value of the group of synaptic memory cells (e.g., 324), the obtained second synaptic weight value being obtained based on the written state values of the synaptic memory cell and each preceding synaptic memory cell of the group of synaptic memory cells (e.g., 324) with respect to the synaptic memory cell.

**[0033]** FIG. 4 depicts a schematic flow diagram of a method 400 of performing neural network inference on a neural processing core configured to perform neural network computations for a neural network, according to various embodiments of the present invention. In various embodiments, the neural processing core is the neural processing core 300 as described herein with reference to FIG. 3 according to various embodiments. The method 400 comprises: at 402, for each memory cell column of the synaptic memory array from a first memory cell column of the synaptic memory array including the first synaptic memory cell to a last memory cell column of the synaptic memory including a last synaptic memory cell of the group of synaptic memory cells (e.g., 324), the first memory cell column to the last memory cell column forming a group of memory cell columns: sensing analog current of the memory cell column to obtain a summed analog current of the memory cell column; and digitizing the summed analog current of the memory cell column to obtain a digitized summed current value of the memory cell column; and summing (at 404) the digitized summed current values obtained for the group of memory cell columns to obtain a total digitized summed current value of the group of memory cell columns, the group of memory cell columns corresponding to a neuron of the synaptic memory array 304.

**[0034]** In various embodiments, the process of configuring/programming or treating a group of synaptic memory cells collectively as a combined synaptic memory cell to represent a synaptic weight of the corresponding synapse of the neural network for various purposes (e.g., neural network inference) may be referred to as a cell-stitching method or function.

**[0035]** In various embodiments, the above-mentioned each memory cell column of the group of memory cell columns has associated therewith a corresponding scaling factor based on an order of the memory cell column in the group of memory cell columns. In addition, the above-mentioned summing the digitized summed current values obtained for the group of memory cell columns comprises: multiplying the digitized summed current values obtained for the group of memory cell columns with the corresponding scaling factors, respectively, to obtain a plurality of products thereof; and summing the plurality of products obtained for the group of memory cell columns to obtain the total digitized summed current value of the group of memory cell columns.

**[0036]** In various embodiments, the method 400 further comprises determining a difference between the total digitized summed current value of the group of memory cell columns and a reference weighted sum value associated with the group of memory cell columns to obtain a signed total digitized summed current value of the group of memory cell columns.

**[0037]** In various embodiments, the method 400 further comprises scaling the total digitized summed current value of the group of memory cell columns based on a first scaling factor configured to scale the total digitized summed current value to a weighted sum value of the group of memory cell columns. In such embodiments, the above-mentioned difference is determined between the weighted sum value of the group of memory cell columns and the reference weighted sum value associated with the group of memory cell columns to obtain a signed weighted sum value of the group of memory cell columns (that is, the above-mentioned signed total digitized summed current value becomes the signed weighted sum value of the group of memory cell columns).

**[0038]** In various embodiments, the method 400 further comprises determining (e.g., computed digitally) the reference weighted sum value associated with the group of memory cell columns, comprising: obtaining, for each synaptic memory cell of the group of synaptic memory cells (e.g., 324), a reference written state value corresponding to the synaptic memory cell, each synaptic memory cell having associated therewith a corresponding scaling factor based on an order thereof in the group of synaptic memory cells (e.g., 324); multiplying the obtained reference written state values obtained for the group of synaptic memory cells (e.g., 324) with the corresponding scaling factors, respectively, to obtain a plurality of products thereof; and summing the plurality of products obtained for the group of synaptic memory cells (e.g., 324) to obtain a summed value. In this regard, the reference weight sum value is determined based on the above summed value obtained.

[0039] In various embodiments, instead of the above-described digital manner of determining the reference weight sum value associated with the group of memory cell columns, the reference weighted sum value may be determined in an analog manner, comprising: for each reference memory cell column of the synaptic memory array from a first reference memory cell column of the synaptic memory array including a first reference memory cell and corresponding to the first memory cell column to a last reference memory cell column of the synaptic memory array including a last reference memory cell and corresponding to the last memory cell column, the first reference memory cell column to the last reference memory cell column forming a group of reference memory cell columns: sensing analog current of the reference memory cell column to obtain a reference summed analog current of the reference memory cell column; and digitizing the reference summed analog current of the reference memory cell column to obtain a digitized reference summed current value of the reference memory cell column; and summing the digitized reference summed current values obtained for the group of reference memory cell columns to obtain a total digitized reference summed current value of the group of reference memory cell columns. In this regard, the above-mentioned each reference memory cell column of the group of reference memory cell columns has associated therewith the corresponding scaling factor of the corresponding memory cell column thereof. Furthermore, the above-mentioned summing the digitized reference summed current values obtained for the group of reference memory cell columns comprises: multiplying the digitized reference summed current values obtained for the group of reference memory cell columns with the corresponding scaling factors, respectively, to obtain a plurality of products thereof; and summing the plurality of products obtained for the group of reference memory cell columns to obtain the total digitized reference summed current value of the group of reference memory cell columns. In various embodiments, the neural processing core 300 comprises: the first reference memory cell at the first memory cell column communicatively coupled to the first synaptic memory cell, the first reference memory cell having a first reference written state value based on a first intended reference value written thereto; and/or a second reference memory cell at a second memory cell column of the group of reference memory cell columns communicatively coupled to the second synaptic memory cell, the second reference memory cell having a second reference written state value based on a second intended reference value written thereto.

[0040] FIG. 5A depicts a schematic drawing of a neural network processor system 500 according to various embodiments of the present invention. The neural network processor system 500 comprises: a neural processing unit 502 comprising a plurality of neural processing

cores 300; a router network 504 comprising a plurality of routers 508 communicatively coupled to the plurality of neural processing cores 300, respectively; and a host processing unit 520 communicatively coupled to the neural processing unit 502 based on the router network 504 and configured to coordinate the neural processing unit 502 for performing neural network  
5 computations. In particular, each neural processing core of the plurality of neural processing cores 300 may be configured or programmed according to the method 200 of programming a synaptic memory array as described herein with reference to FIG. 2 according to various embodiments. In other words, each neural processing core of the plurality of neural processing  
10 cores 300 may be configured or programmed in the same, similar or corresponding manner as the neural processing core 300 as described herein with reference to FIG. 3. In various embodiments, the host processing unit 520 may be configured to send instructions to one or more neural processing cores 300 for programming the synaptic memory array 304 of the one or more neural processing cores 300 according to the method 200 of programming a synaptic memory array as described herein with reference to FIG. 2 according to various embodiments.

15 **[0041]** For simplicity and clarity, the neural network processor system 500 is illustrated in FIG. 5A with only one NPU 502. However, it will be appreciated by a person skilled in the art that the neural network processor system 500 is not limited to only one NPU, and additional one or more NPUs (configured in the same, similar or corresponding manner as the NPU 502 described herein according to various embodiments) may be included in the neural network  
20 processor system 500 as desired or as appropriate.

**[0042]** In various embodiments, the neural network processor system 500 further comprises a fabric bridge. In this regard, the host processing unit 520 may be communicatively coupled to the neural processing unit 502 (or to each neural processing unit) via the fabric bridge and the router network 504. In particular, the host processing unit 220 may be communicatively  
25 coupled to the router network 504 via the fabric bridge.

**[0043]** In various embodiments, in each neural processing unit 502, the plurality of neural processing cores 300 is arranged in a two-dimensional (2D) array (comprising rows and columns) and each neural processing core 300 has an associated unique address based on its position in the 2D array and the neural processing unit 502 it belongs to. For example, each  
30 neural processing core 300 may have an address based on the row and column at which it is located in the 2D array.

**[0044]** In various embodiments, the neural network processor system 200 may be formed as an integrated neural processing circuit. The neural network processor system 200 may also be embodied as a device or an apparatus.

**[0045]** For illustration purpose only and without limitation, FIG. 5B shows a block diagram of an example configuration of the neural processing core 300 according to various example embodiments of the present invention. The neural processing core 300 comprises a synaptic memory array 304 configured to perform in-memory matrix computations (e.g., configured or programmed as described hereinbefore with reference to FIG. 3), a row peripheral circuit or block 511 (e.g., including a row address decoder and an axon buffer), a plurality of line drivers 510, a column peripheral circuit or block 512 (e.g., including a column address decoder, a column multiplexer (MUX), a Trans-impedance amplifier (TIA), an analog-to-digital converter (ADC) and a neuron circuit), a core control unit or block 513 (e.g., including scheduling, routing and configuration information), and a network interface (NI) unit or block 514. For example, the NI unit 514 may be configured to forward packets received from its corresponding router 508 to the core control unit 513. The core control unit 513 may send row control information (e.g., axon ID and axon input value) to the row peripheral circuit 511 to control its components (e.g., the row address decoder and the axon buffer), which in turn controls the plurality of line drivers 510. The core control unit 513 may also send column control information (e.g., column time-multiplexing control signal and neuron circuit control signal) to the column peripheral circuit 512 to control its components (e.g., the column address decoder, the column MUX, the TIA, the ADC and the neuron circuit (which may be configured to perform the cell-stitching function according to various embodiment of the present invention)). Once the neuron computations are completed for the current time-step, the neuron outputs may then be conveyed back to the core control unit 513 for data packet transmission via the NI unit 514 and the corresponding router 508.

**[0046]** A computing system, a controller, a microcontroller or any other system providing a processing capability may be presented according to various embodiments in the present disclosure. Such a system may be taken to include one or more processors and one or more computer-readable storage mediums. For example, the neural network processor system 500 described hereinbefore may include a number of processing units (e.g., the host processing unit (e.g., one or more CPUs) 520 and one or more NPUs 502) and one or more computer-readable storage medium (or memory) 524 which are for example used in various processing carried out therein as described herein. A memory or computer-readable storage medium used in various

embodiments may be a volatile memory, for example a DRAM (Dynamic Random Access Memory) or a non-volatile memory, for example a PROM (Programmable Read Only Memory), an EPROM (Erasable PROM), EEPROM (Electrically Erasable PROM), or a flash memory, e.g., a floating gate memory, a charge trapping memory, an MRAM (Magnetoresistive Random Access Memory) or a PCRAM (Phase Change Random Access Memory).

**[0047]** In various embodiments, a "circuit" may be understood as any kind of a logic implementing entity, which may be special purpose circuitry or a processor executing software stored in a memory, firmware, or any combination thereof. Thus, in various embodiments, a "circuit" may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g., a microprocessor (e.g. a Complex Instruction Set Computer (CISC) processor or a Reduced Instruction Set Computer (RISC) processor). A "circuit" may also be a processor executing software, e.g. any kind of computer program, e.g., a computer program using a virtual machine code such as e.g. Java. Any other kind of implementation of the respective functions may also be understood as a "circuit" in accordance with various embodiments. Similarly, a "module" may be a portion of a system according to various embodiments in the present invention and may encompass a "circuit" as above, or may be understood to be any kind of a logic-implementing entity therefrom.

**[0048]** Some portions of the present disclosure are explicitly or implicitly presented in terms of algorithms and functional or symbolic representations of operations on data within a computer memory. These algorithmic descriptions and functional or symbolic representations are the means used by those skilled in the data processing arts to convey most effectively the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities, such as electrical, magnetic or optical signals capable of being stored, transferred, combined, compared, and otherwise manipulated.

**[0049]** Unless specifically stated otherwise, and as apparent from the following, it will be appreciated that throughout the present specification, discussions utilizing terms such as "writing", "obtaining", "determining", "multiplying", "summing", "dividing", "sensing", "digitizing", or the like, refer to the actions and processes of a computer system, or similar electronic device, that manipulates and transforms data represented as physical quantities within the computer system into other data similarly represented as physical quantities within the computer system or other information storage, transmission or display devices.

**[0050]** The present specification also discloses a system (e.g., which may also be embodied as a device or an apparatus) for performing various operations/functions described herein. Such a system or apparatus may be specially constructed for the required purposes, or may comprise a general purpose computer or other device selectively activated or reconfigured by a computer program stored in the computer. The algorithms presented herein are not inherently related to any particular computer or other apparatus. Various general purpose machines may be used with computer programs in accordance with the teachings herein. Alternatively, the construction of more specialized apparatus to perform various method steps may be appropriate.

**[0051]** In addition, the present specification also at least implicitly discloses a computer program or software/functional module, in that it would be apparent to the person skilled in the art that various steps described herein may be put into effect by computer code. The computer program is not intended to be limited to any particular programming language and implementation thereof. It will be appreciated that a variety of programming languages and coding thereof may be used to implement various methods/techniques disclosed herein.

Moreover, the computer program is not intended to be limited to any particular control flow. There are many other variants of the computer program, which can use different control flows without departing from the scope of the present invention. It will be appreciated to a person skilled in the art that various modules may be software module(s) realized by computer program(s) or set(s) of instructions executable by a computer processor to perform the required functions as appropriate, or may be hardware module(s) being functional hardware unit(s) designed to perform the required functions as appropriate. It will also be appreciated that a combination of hardware and software modules may be implemented as appropriate.

**[0052]** Furthermore, one or more of the steps of the computer program/module or method may be performed in parallel rather than sequentially. Such a computer program may be stored on any computer readable medium. The computer readable medium may include storage devices such as magnetic or optical disks, memory chips, or other storage devices suitable for interfacing with a general purpose computer. The computer program when loaded and executed on such a general-purpose computer effectively results in an apparatus that implements various steps described herein.

**[0053]** In various embodiments, there is provided a computer program product, embodied in one or more computer-readable storage mediums (non-transitory computer-readable storage medium), comprising instructions executable by one or more computer processors (e.g., the host processing unit 520) to control or instruct the neural processing core 300 to perform the

method 200 of programming the synaptic memory array (e.g., 324) of the neural processing core 300 as described herein with reference to FIG. 2 according to various embodiments of the present invention. Accordingly, various computer programs or modules described herein may be stored in a computer program product receivable by a system for execution by at least one processor of the system to perform various functions.

**[0054]** Various software or functional modules described herein may also be implemented as hardware modules. More particularly, in the hardware sense, a module is a functional hardware unit designed for use with other components or modules. For example, a module may be implemented using discrete electronic components, or it can form a portion of an entire electronic circuit such as an Application Specific Integrated Circuit (ASIC). Numerous other possibilities exist. Those skilled in the art will appreciate that the software or functional module(s) described herein can also be implemented as a combination of hardware and software modules.

**[0055]** In various embodiments, the neural network processor system 500 may be realized by or embodied as any computer system (e.g., portable or desktop computer system, such as tablet computers, laptop computers, mobile communications devices (e.g., smart phones), and so on) including the host processing unit 520 and the NPU 502 configured as described herein according to various embodiments, such as a computer system 600 as schematically shown in FIG. 6 as an example only and without limitation. Various methods/steps or functional modules may be implemented as software, such as a computer program (e.g., one or more neural network applications) being executed within the computer system 600, and instructing the computer system 600 (in particular, one or more processors therein) to conduct various methods/functions of various embodiments described herein. The computer system 600 may comprise a computer module 602, input modules, such as a keyboard 604 and a mouse 606, and a plurality of output devices such as a display 608. The computer module 602 may be connected to a computer network 612 via a suitable transceiver device 614, to enable access to, e.g., the Internet or other network systems such as Local Area Network (LAN) or Wide Area Network (WAN). The computer module 602 in the example may include a processor 618 (e.g., corresponding to the host processing unit 520 of the neural network processor system 500 as described herein according to various embodiments) for executing various instructions (e.g., neural network application(s)), a neural network processor 619 (e.g., corresponding to the neural processing unit 510 of the neural network processor system 500 as described herein according to various embodiments), a Random Access Memory (RAM) 620 and a Read Only Memory (ROM) 622.

The neural network processor 619 may be coupled to the interconnected bus (system bus) 628 via one or more fabric bridges. The computer module 602 may also include a number of Input/Output (I/O) interfaces, for example I/O interface 624 to the display 608, and I/O interface 626 to the keyboard 604. The components of the computer module 602 typically communicate  
5 via an interconnected bus 628 and in a manner known to the person skilled in the relevant art.

**[0056]** It will be appreciated by a person skilled in the art that the terminology used herein is for the purpose of describing various embodiments only and is not intended to be limiting of the present invention. As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further  
10 understood that the terms "comprises" and/or "comprising" when used in this specification, specify the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof. Furthermore, it will be appreciated by a person skilled in the art that the term "and/or" refers to any one or more of the stated items,  
15 including any combination thereof.

**[0057]** Any reference to an element or a feature herein using a designation such as "first", "second" and so forth does not necessarily limit the quantity or order of such elements or features, unless stated or the context requires otherwise. For example, such designations may be used herein as a convenient way of distinguishing between two or more elements or instances  
20 of an element. Thus, a reference to first and second elements does not necessarily mean that only two elements can be employed, or that the first element must precede the second element. In addition, a phrase referring to "at least one of" a list of items refers to any single item therein or any combination of two or more items therein.

**[0058]** In order that the present invention may be readily understood and put into practical effect, various example embodiments of the present invention will be described hereinafter by way of examples only and not limitations. It will be appreciated by a person skilled in the art that the present invention may, however, be embodied in various different forms or configurations and should not be construed as limited to the example embodiments set forth  
25 hereinafter. Rather, these example embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the present invention to those skilled in the art. In particular, for better understanding of the present invention and without limitation or loss of generality, and unless stated or context required otherwise, various example embodiments of the present invention will now be described with respect to synaptic memory  
30

cells each being a multi-bit cell having a number of supported states being a power of 2. However, as explained hereinbefore, the present invention is not limited to such multi-bit cells and that, for example, single-bit cells may be provided or multi-bit cells having a number of supported states other than a power of 2 may be provided.

5 [0059] Various example embodiments relate to an effective implementation of analog neuromorphic computing for neural networks, and more particularly, to cell-stitching for analog neuromorphic computing.

[0060] As described in the background, a highly efficient approach towards achieving the goal of a non-von-Neumann architecture is via analog computing, with one of the most common  
10 forms being analog current summing of synaptic memory cells for a given neuron, where each synaptic memory cell is typically represented by a memristor cell (e.g., a programmable resistor). The change of membrane potential in a neuron (e.g., denoted as  $\Delta P$ ), whether in a spiking neuron (which is more biologically inspired), or an artificial neuron (used in deep learning), typically follow the above-mentioned Equation (1), as a sum of axon inputs weighted  
15 by its synapses (which may simply be referred to as a weighted sum).

[0061] In both spiking neural networks (SNNs) and artificial neural networks (ANNs), the weighted sum in the above-mentioned Equation (1) may be the most compute-intensive part. In ANNs, each such multiplication and its accompanying addition together may be referred to a MAC (Multiply-and-Accumulate) operation. In SNNs, since  $x_i$  is generally 0 or 1, the MAC  
20 degrades to an addition operation when  $x_i$  is 1, and degrades to no-operation when  $x_i$  is 0. Therefore, each addition operation when  $x_i$  is 1 is referred to as a Synaptic Operation (SOP).

[0062] Memristors refer to a broad category of memory devices which behave as programmable resistors, with examples such as RRAM (Resistive RAM), PCRAM (Phase Change RAM), MRAM (Magnetic RAM), where RAM means Random Access Memory. If  
25 each synapse corresponds to a memristor device (which may be referred to as a synaptic memory cell, or simply as a cell), and the memristor cell's conductance  $c_i$  (which is  $1/r_i$ , inverse of its resistance  $r_i$ ) is programmed to be proportional to the synaptic weight  $w_i$ , and a voltage  $v_i$  proportional to  $x_i$  is applied to the cell, then if all the memristor cells of a neuron are connected in parallel, the total current being drained by all  $n$  cells may be derived as follows:

30

$$c_i = 1/r_i \propto w_i, \quad v_i \propto x_i$$

$$I_{sum} = \sum_{i=0}^{n-1} c_i * v_i \propto \sum_{i=0}^{n-1} w_i * x_i = \Delta P$$

(Equation 2)

**[0063]** In various example embodiments, when two memristor devices are used to store positive and negative weights respectively and together represent a signed weight (as will be described later below), the two devices together are considered as one cell (e.g., at an abstract level).

5 **[0064]** Therefore, by sensing and sampling the total current drained by all  $n$  cells in a neuron, a value proportional to the weighted sum can be obtained based on the above-mentioned Equation (1). Such sensing and sampling may be performed by an analog-to-digital converter (ADC). In addition, because ADCs generally take voltage instead of current as input, in various example embodiments, the above-mentioned total current is translated into voltage. For  
10 example and without limitation, a preferred translation circuit is a Trans-impedance Amplifier (TIA). The output voltage of a TIA is proportional to the input current, thus introducing another linear scaling factor. Furthermore, an ADC may translate an input voltage into a multi-bit value, thus introducing a fourth linear scaling factor. So overall, there may be four linear scaling factors involved in deriving the weighted sum  $\Delta P$ , namely, (1)  $c_i/w_i$ ; (2)  $v_i/x_i$ ; (3)  $V_{\text{output,TIA}} / I_{\text{sum}}$  in TIA; and (4) ADC output /  $V_{\text{output,TIA}}$ . In various example embodiments, an additional scaling factor  $F$  may be utilized to convert ADC output into the actual weighted sum, or any range that is suitable for neural network computation in hardware. For example, for SNN, since the output of each layer is binary, there is no need to apply the above-mentioned scaling factor  $F$  (or in other words,  $F$  can be 1), the SNN firing threshold can instead be scaled properly to be  
20 compatible with ADC output. For ANN, it may also be possible to operate in the ADC output range (i.e.,  $F=1$ ) or any range that is suitable for hardware implementation (e.g., some other  $F$ ), for example, by modifying activation and/or quantization functions. According to various example embodiments, the above-mentioned scaling factor  $F$  may be derived during system calibration, by presenting to the hardware with some known axon inputs  $x_i$  and obtain the  
25 scaling factor as  $F = \frac{\sum_{i=0}^{n-1} w_i * x_i}{\text{ADC\_output}}$ . For example, a number of (e.g., a few) sets of axon inputs may be used to obtain several estimations of  $F$ , and then averaged or applied with linear regression to obtain a scaling factor  $F$  that is more robust or accurate.

**[0065]** In various example embodiments, other sensing and sampling mechanisms may be provided, such as voltage clamping circuits in conjunction with a sense amplifier (SA), and the  
30 latter may be viewed as a crude equivalent of ADC and may use current sensing and current comparators against multiple reference currents to determine the output. Such voltage clamping circuit plus sense amplifier behaves functionally like a TIA plus ADC, and the above-mentioned linear factors (3) and (4) may be treated as one combined linear factor (3) SA output /  $I_{\text{sum}}$ .

Similarly, a scaling factor may also be utilized which may be estimated as  $F = \frac{\sum_{i=0}^{n-1} w_i * x_i}{SA\_output}$ . In various example embodiments, the *SA\_output* may be treated as a form of *ADC\_output*, since both are digitized values of current, and thus, whether the current is transformed via TIA or some other circuit may be irrelevant.

5   **[0066]**   For a neural network that has only non-negative weights and non-negative axon inputs, the weight-to-device mapping as described with respect to the above-mentioned Equation (2) and FIG. 1 may be sufficient. However, since conductance may only be non-negative, a problem may arise if the weight can be negative (which may be quite common in neural networks). Furthermore, various example embodiments of the present invention note that  
10   the dynamic range of conductance in a given memristor technology may be fairly limited as opposed to an infinite range of  $[0, +\infty)$ .

**[0067]**   According to various example embodiments, two approaches are provided for implementing negative weights on memristor cells. A first approach uses two memristor devices (a pair of memristor devices), where a first memristor device (e.g., referred to as device  
15   0) always stores the positive component, and a second memristor device (e.g., referred to as device 1) always stores the negative component. If the weight is positive, device 1 may generally be programmed into the least conductive state which can be referred to as a 0 state (although it is not necessarily a zero conductance state). Conversely, if the weight is negative, device 0 may generally be programmed into the least conductive state. If the weight is zero,  
20   both devices 0 and 1 may be programmed into the least conductive state. Accordingly, the difference in conductance between these two devices may be used to represent the synaptic weight. This first approach only increases weight precision by 1-bit (the sign bit) by using two devices. For example, if each device is 4-bit, then the device pair (which can be considered as one cell (e.g., at abstract level)) achieves signed 5-bit.

25   **[0068]**   A second approach uses a reference level, which may be near (e.g., with an offset as will be described later below) or at the middle of all raw states, and the main cell's written/stored state/level is subtracted by this reference level to support a signed weight. A key advantage of the second approach over the first approach is more efficient use of bits. For example, if the reference level is either stored digitally, or stored in a shared memristor cell,  
30   then the overhead across the main cells can be very small. As will be described later, when the reference level approach is combined with the cell-stitching techniques according to various embodiments of the present invention, two 4-bit cells can be combined to form a virtual 8-bit

synapse when there is no bit-overlap, or to form a virtual 7-bit synapse when there is 1-bit overlap, in contrast to a signed 5-bit synapse obtained by the first approach.

[0069] With the introduction of reference level in the second approach to handle negative weights, the derivation of the above-mentioned scaling factor F may be modified from  $F = \frac{\sum_{i=0}^{n-1} w_i * x_i}{ADC\_output}$  to  $F = \frac{\sum_{i=0}^{n-1} (w_i + ref_i) * x_i}{ADC\_output_{main}}$ , where  $ref_i$  denotes the reference level used for  $w_i$  (though the same reference level may be used for all  $w_i$ ), where  $ADC\_output_{main}$  denotes the ADC output (or more generally any output from *DIGITIZE()* function) of the main cell column for the current neuron being evaluated.

[0070] In various example embodiments, it is also possible to apply cell-stitching based on the first approach, whereby the first pair of devices are used to represent the higher order bits, and the second pair of devices are used to represent the residual synaptic-level weight arising from the first pair of devices. For example, if each device is 3-bit, then four of such devices together may represent a virtual signed 8-bit synapse when there is no bit-overlap. The difference in conductance between two devices in the same device pair (i.e., a cell) may be used to represent a signed number, and during inference the difference in these two devices' current may be treated as that cell's cell current.

[0071] When axon inputs (i.e., input activations) may be negative, if it is difficult for the electrical circuit to present a negative activation voltage, then similar to the above-described two approaches for signed synapse, two passes (with the first pass applying only the positive activations and the second pass applying only the negative activations) may be used, but for both passes, the activation voltages presented in the circuit are always positive, and then subtract the two ADC outputs from the two passes in digital domain and treat it as the final ADC output. In another technique, a reference activation voltage may be provided, and its ADC output is subtracted from the ADC output of the main activation (which are shifted by using the reference activation voltage) and treated as the final ADC output. This ADC conversion based on the reference activation voltages may only need to be done once per unique values of axon inputs to save energy.

[0072] As described in the background, one of the limitations in analog computing is the achievable precision (i.e., weight precision or synapse precision). For example, if a memristor cell can only express 4-bit (i.e., 16 linear states), then the synaptic weight can only have a 4-bit precision, and this may limit the accuracy achievable in the neural network. Although there have been disclosed conventional techniques for "concatenating" multiple memristor cells to form a higher precision synapse, such conventional techniques only work in the digital

computing mode, where the cell values are read out first and then digitally concatenated. In this regard, various example embodiments of the present invention note that because the states in a memristor cell are rarely a quantum phenomenon, these states generally are not quantum- or quantization-precise. For example, when programming a state value of “5” (which corresponds to a certain level of conductance) in a 4-bit memristor cell, the actual value after programming may be, for example, “5.2”, “4.7”, and so on, with the difference from “5” or the intended value (i.e., corresponding to the intended level of conductance) appearing as error to the intended value. Such errors are generally static, and may also be referred to herein as write variations, because the errors are generally fixed after writing/programming. This is different from dynamic errors, such as Random Telegraph Noise (RTN), where a cell can change its state randomly at run-time after writing/programming.

**[0073]** In this regard, various embodiments of the present invention note that, in analog computing mode, the above-mentioned conventional techniques of “concatenating” two or more memristor cells are generally impractical or ineffective, because the error in the highest-order-bit(s) (which may also be referred to as the most-significant-bit(s) or MSB(s)) cell will be dominant and it will be essentially useless to add any low-order-bit(s) cell(s) (e.g., the last lowest-order-bit(s) cell may be referred to as the least-significant-bit(s) or LSB(s) cell). In other words, various embodiments of the present invention note that although the above-mentioned conventional techniques may attempt to increase the number of bits for representing a synaptic weight by “concatenating” two or more memristor cells, the weight precision (or synapse precision) achieved by such conventional techniques may not be improved (or increased), or at least not improved effectively.

**[0074]** To address, or at least ameliorate, the above-mentioned problems associated with the above-mentioned conventional techniques, various example embodiments of the present invention provide a different technical approach to using multiple synaptic memory cells (e.g., memristor cells) to form a higher precision synapse. In particular, in contrast to the above-mentioned conventional techniques of “concatenating” two or more memristor cells, a method of programming a synaptic memory array (e.g., corresponding to the method 200 of programming a synaptic memory array as described herein with reference to FIG. 2 according to various embodiments) is provided according to various example embodiments, which may herein be referred to as “cell-stitching”. In the cell-stitching method, according to various example embodiments, the intended synaptic weight  $w$  is represented in an MSBs cell, and any short-fall from  $w$ , is then represented in the next-significant-bits (NSBs) cell, and so on, for a

total of B ( $B \geq 2$ ) memristor cells. In various example embodiments, the cell-stitching method or cell-stitching programming method (which may be referred to herein as the first or general cell-stitching method) may be performed or implemented in the following manner or steps:

1. Each k-th significant-bits cell (with  $k=0$  meaning MSBs) has a corresponding scaling factor  $f_k$ , typically a power of 2. For example, for an 8-bit weight and four 2-bit memristor cells without any bit-overlap (which will be discussed in detail later below),  $f_0 = 2^6, f_1 = 2^4, f_2 = 2^2, f_3 = 2^0$ ; Initialize with  $j = 0$ , where  $j$  denotes the number of cells stitched so far;
2. Let  $\tilde{w}(C_k)$  denote the achieved written value (or written state value) of k-th significant-bits cell  $C_k$  (obtained by measurement) where  $k = 0$  to  $j-1$ , the corresponding synaptic-level weight (or obtained synaptic weight value) achieved so far may be derived or obtained by stitching  $C_0$  to  $C_{j-1}$ , as  $w' = \sum_{k=0}^{j-1} f_k * \tilde{w}(C_k)$ . Note that  $w' = 0$  when  $j=0$ , and  $\tilde{w}(C_k)$  may be a signed value and the means to achieve a signed value from raw states which are usually positive will be discussed in detail later below;
3. Let  $E_j = w - w'$ , which represents the residual synaptic weight (or residual synaptic weight value) after stitching  $C_0$  to  $C_{j-1}$ ;
4. Then, the most significant component of  $E_j$  may be represented in  $C_j$  (i.e., immediately succeeding cell), by writing a nominal/intended value of  $E_j * f_j^{-1}$  to  $C_j$ . In various example embodiments, the memristor cell write operation is configured to limit the writing error to be within what  $C_{j+1}$  is able to compensate during cell-stitching, which by default or when  $j+1 \geq B$  may be within  $(-0.5, 0.5)$  as an example; and
5. Next, let  $j = j + 1$  (i.e., increment  $j$ ), then repeat step (2) to step (5) in a loop while  $j < B$ .

[0075] In various example embodiments, there is also provided a method of performing neural network inference (e.g., corresponding to the method of performing neural network inference on a neural processing core as described herein with reference to FIG. 4 according to various embodiments). After two or more memristor cells have been programmed ("stitched") according to the above-described cell-stitching method, the method of performing inference may include:

1. During inference, measure the summed cell current  $I_{sum\_main,j}$  for a given neuron for its j-th significant-bits cells in the j-th cycle, by sensing the digitized summed current via the digitization output  $Y_{sum\_main,j} = DIGITIZE(I_{sum\_main,j})$  and derive the neuron's total digitized summed current as  $Y_{sum\_main} = \sum_{j=0}^{B-1} f_j * Y_{sum\_main,j}$ . In this regard, the function

*DIGITIZE()* may be an electrical circuit that converts an analog input current into a digitized form, and may be implemented by cascading a TIA and an ADC; and

2. Compute  $Z_{sum\_main} = Y_{sum\_main} * F$ , where F is a predetermined scaling factor (e.g., corresponding to the first scaling factor as described hereinbefore according to various embodiments) for converting the digitized summed current into weighted sum or a range suitable for neural network computation in hardware;

**[0076]** By way of an example, in an example case of “stitching” two cells, the MSBs cell is used to represent the intended synaptic weight using the available bit-precision and hence the dynamic range of the MSBs cell, and the LSBs cell is used to compensate for the error (which may also be referred to as noise (e.g., write noise)) in the MSBs cell, by representing the residual synaptic weight. In various example embodiments, the residual synaptic weight means the short-fall, i.e., the value remaining (which may be negative) that has not been captured in the cells stitched so far, which corresponds to  $w-w'$  in the above-mentioned step (3) of the cell-stitching programming method. Accordingly, it is the opposite of the synaptic weight error which may be expressed as  $w'-w$ .

**[0077]** In various example embodiments, the LSBs cell’s value (ordered bits represented by the LSBs cell) may be logically shifted left by 1-bit if the potential error range in the MSBs cell is more than the LSBs cell is able to handle. For example, with 1-bit left shift in LSBs cell where two 4-bit cells are “stitched” together, the “stitched” synapse can achieve  $4+4-1 = 7$ -bit precision. Accordingly, as mentioned in relation to the cell-stitching method above, when there is no bit-overlap, two 4-bit cells have their corresponding scaling factors  $f_0 = 2^4, f_1 = 2^0$ , respectively. On the other hand, if there is a 1-bit overlap, their corresponding scaling factors become  $f_0 = 2^4, f_1 = 2^1$ , respectively, which is still able to represent an 8-bit weight, but only the top-7 bits may be reliable, and thus, the effective precision is 7-bit. In various example embodiments, the cell-stitching method may be configured or designed to compensate for static errors, but not for dynamic errors, such as RTN. Furthermore, various example embodiments note that the synapse formed after cell-stitching is not quantum-like: for example, two 4-bit cells without bit overlap can achieve 8-bit weight precision, but it is not an exact discrete 8-bit value, since there may be some noise after its least-significant-bit, typically with a range limited by  $(-0.5, 0.5)$ , because any noise larger than this range could flip the least-significant-bit, contradicting with the definition of its acclaimed precision. For example, as a result, there may be no need to have a bit-overlap of more than 1, because a bit-overlap of 2 or more would imply that the higher-order-bits cell’s precision is really tangibly lower than what it can claim and not

just caused by borderline errors. In addition, it is also possible for a k-th-significant-bits cell and the corresponding scaling factor to be non-power-of-2. For example, the MSBs and LSBs cells may be 5-state cells (which is slightly more than 2-bit but less than 3-bit), and thus,  $f_0 = 5, f_1 = 1$  may be provided to support a total of up to  $5 \times 5 = 25$  states (more than 4-bit but less than 5-bit) after cell-stitching and without (including a fraction of) bit-overlap, which may be useful in pushing the performance limit of individual memristor cells and of cell-stitching, but may incur an expense when implementing multiplication with the scaling factors, since multiplying a power of 2 can be easily implemented as bit-shift, whereas multiplying a non-power-of-2 requires a general multiplier circuit.

10 **[0078]** FIGs. 7A to 7C depict schematic drawings of three example arrangements of synaptic memory cells, along with an illustration of the associated errors or noise. FIG. 7A shows an example native 8-bit cell, and illustrates that if a native 8-bit memristor cell was available (which is not yet shown to be feasible commercially), it is not a “perfect” 8-bit cell where its stored/written state value corresponds to an exact 8-bit value. Instead, it would  
15 represent an 8-bit integer weight with an error  $\varepsilon \in (-0.5, 0.5)$ . If the weight is a signed 8-bit integer, the maximum signal-to-noise ratio (SNR, where noise corresponds to error) is  $2^{8-1}/0.5 = 256:1$ . In practice, various example embodiments note that if only 4-bit cells are available, then the naïve concatenation as shown in FIG. 7B would not achieve 8-bit synapse, because the MSBs cell, used to represent higher order 4-bits, would have an error/noise  $\in (-0.5, 0.5)$  at  
20 individual cell level but an error/noise of  $\varepsilon_{m4} \in (-0.5, 0.5) \times 2^4$  at synapse level because its sensed value would be shifted left by 4 bits, which would render the LSBs cell’s content useless (or irrelevant). The above-mentioned subscript m4 denotes the most-significant 4 bits, and  $\varepsilon_{m4}$  denotes synaptic level error introduced by the 4-bit MSBs cell, and later below, similarly, the subscript l4 denotes the least-significant 4 bits, and  $\varepsilon_{l4}$  denotes synaptic level error introduced  
25 by the 4-bit LSBs cell.

**[0079]** In FIG. 7C, according to various example embodiments of the present invention, the native 4-bit MSBs cell is used to represent a  $2^4$  times scaled-down version of the entire 8-bit weight. In this regard, there exists an error because the cell is only 4-bit, and the LSBs cell is used to represent the remaining/residual component of the weight (i.e., the opposite of the error)  
30 times a corresponding scaling factor of  $2^1$  because 1-bit overlap is used in the example shown in FIG. 7C to make it essentially improbable for the error to exceed the dynamic range of the LSBs cell. In FIG. 7C, the final achieved synapse precision is 7-bit due to the 1-bit overlap. In various example embodiments, the error from MSB cell is controlled such that it does not

exceed the dynamic range of the LSBs cell (or the immediately succeeding cell), for example, by multiple write-verify operations if necessary where write operations are repeated until the error is within the dynamic range of the LSBs cell, or by supporting precise control of the nominal center state to be programmed. In this manner, there may be no bit overlap in cell-stitching, and the two 4-bit cells can achieve 8-bit synapse precision, and so forth.

**[0080]** In various example embodiments, for the case of “stitching” more than two cells (i.e., a group of more than two cells), the above-described cell-stitching method may be similarly applied, whereby the residual synaptic weight (i.e., opposite of remaining synaptic error) associated with the cells stitched so far is represented by (or written to) the immediately next cell, and so forth, in an iterative manner until the residual synaptic weight associated with the cells stitched so far is represented by the last cell in the group.

### *Synaptic Weight as Unsigned Integer*

**[0081]** An example case of the synaptic weight being an unsigned integer of m-bits will now be described according to various example embodiments of the present invention. FIG. 8 depicts a schematic drawing of a native 4-bit cell for illustrating an example of how a 4-bit state may be represented using only two 2-bit RRAM cells. In particular, FIG. 8 illustrates cell-stitching with precise control of nominal center state in MSBs cell. In FIG. 8, the top row shows 16 states of a native 4-bit RRAM cell, and each ellipse on the bottom row shows an example of a 4-bit state may be encoded/represented in the MSBs cell of the two 2-bit RRAM cells. In FIG. 8, a small triangle 804 denotes an example illustration of the actual written value (written state value) in the MSBs cell, where the intended state value for the MSBs cell is 0, and the actual written state value of this MSBs cell is 0.4 (denoted by triangle 804), and due to the corresponding scaling factor for the MSBs cell is  $2^2$ , this introduces an error ( $\epsilon_{m2}$ ) of  $(0.4-0) \times 2^2 = 1.6$  at the synaptic weight level, and shall be compensated for in the LSBs cell accordingly.

**[0082]** For example, to represent state 0 (of 4-bit) (the leftmost state in the top row of FIG. 8), the 2-bit MSBs cell may be programmed to state  $0 \times 2^{-2} = 0$  (of 2-bit), that is, scaling factor (associated with the 2-bit MSBs cell)  $f_0 = 2^2$ . The MSBs cell’s actual written state (denoted by a small triangle 804 in FIG. 8) may then have a nominal center value of 0 and fall in a range (e.g., -0.5, 0.5), and after 2-bit left-shift due to being MSBs cell, the MSBs cell may represent, at synaptic level, a range of  $(-0.5, 0.5) \times 2^2$ , or  $(-2, 2)$ , and the synaptic level error introduced by the MSBs cell may also be in the range of  $(-2, 2)$ . Next, the 2-bit LSBs cell allows unsigned raw states of 0, 1, 2, 3, or more generally  $[0, 2^l - 1]$  for an l-bit memristor cell. Since each state

may be assumed to have error range of  $(-0.5, 0.5)$ , the state range of the LSBs cell with error may be  $(-0.5, 3.5)$  or more generally  $(-0.5, 2^l - 0.5)$ . By defining the nominal middle reference level of LSB cell to be  $2^{l-1} - 0.5 = 1.5$  (or more generally  $2^{l-1} - 0.5$ ), then the signed state range including error may be  $(-2, 2)$  or more generally  $(-2^{l-1}, 2^{l-1})$  as will also be explained later below under “Synaptic Weight as Signed Integer”, being just enough to cover the error introduced by the MSB cell. Furthermore, assuming that the 2-bit cells (whether MSBs or LSBs) can be programmed with precise control of its nominal center state value, the nominal center state value of LSB cell (after subtracting middle reference to become signed) is the residual synaptic level weight, which is opposite of the synaptic level error introduced by the MSBs cell. As described hereinbefore, another technique is to use multiple write-verify operations to keep writing the MSBs cell until the error from the MSBs cell can be compensated by the LSBs cell. In various example embodiments, a nominal middle reference level other than  $2^{l-1} - 0.5$  may also be used, though it may result in less efficient writing. In various example embodiments, even the LSBs cell may be used in its unsigned or raw mode (e.g., with a reference level of 0), as long as the cell write operation can ensure that the residual synaptic-level weight (i.e., opposite of remaining error) is also non-negative and within the range supportable by the remaining cells.

**[0083]** As another example, to represent state 6 (of 4-bit) in FIG. 8, the MSBs cell may be programmed to the nominal center state  $6 \times 2^{-2} = 1.5$  (of 2-bit) (associated scaling factor  $f_0 = 2^2$ ). Similarly, in FIG. 8, to represent state 9 or 15 (of 4-bit), the MSB cell's nominal center state should be  $9 \times 2^{-2} = 2.25$ , or  $15 \times 2^{-2} = 3.75$ , respectively (associated scaling factor  $f_0 = 2^2$ ). Therefore, various example embodiments assume that the nominal center state can be precisely controlled, but still assume that the error range is  $(-0.5, 0.5)$  at cell level.

**[0084]** By way of an example, FIG. 9 depicts a table (Table 1) showing the cell-stitching examples described above with reference to FIG. 8. In Table 1,  $\varepsilon$  denotes the error at the cell level (defined as actual state subtracted by nominal center state), and  $\varepsilon_{m2}$  denotes the synaptic level error introduced by the 2-bit MSBs cell, which in this case is  $\varepsilon \times 2^2$  because it is the MSBs cell and is 2-bit higher than the LSBs cell.  $\varepsilon_{l2}$  denotes the synaptic level error introduced by the 2-bit LSBs cell, and because in this case there is no bit overlap,  $\varepsilon = \varepsilon_{l2}$  and reflects the final synaptic level error. The nominal (center) state of MSBs cell in this case may be determined as  $w \times 2^{-2}$  (associated scaling factor  $f_0 = 2^2$ ), where  $w$  is the desired 4-bit state. Accordingly, the actual 4-bit state equivalent achieved by cell-stitching in this example may be obtained by  $w_{m2} \times 2^2 + w_{l2}$ , (associated scaling factors  $f_0 = 2^2$  and  $f_1 = 2^0$ ) where  $w_{m2}$  and  $w_{l2}$  denote actual

MSBs and LSBs cell states, respectively, and  $w_{12}$  denotes a signed state obtained by raw state subtracted by  $2^1 - 0.5$ .

[0085] Various example embodiments note that, for example, when the desired state is 15 (of 4-bit), MSBs cell's nominal center of 3.75 may appear to exceed the range of a 2-bit cell (-0.5, 3.5). To address this, according to various example embodiments, the programming function may attempt to program as large a state as possible, so that the state does not exhibit a range of (2.5, 3.5) but more shifted toward right, to (3.25, 4.25). In practice, since the raw cell state originally should have a range of (-0.5, 3.5) including error, it would likely achieve an actual state range such as (3.0, 3.5), and with sufficient programming, it may achieve a range such as (3.25, 3.5).

[0086] If the desired state is unsigned, then the MSBs cell can also be used in its original, unsigned mode. This means, the error introduced by the MSB cell is bounded by (-0.5, 0.5), as long as the MSBs cell is capable of a precise nominal central state. Actual error range may be smaller, if the desired nominal central state is close to the range boundary of the MSB cell.

15

#### ***Synaptic Weight as Signed Integer***

[0087] The case of the synaptic weight being a signed integer of  $m$ -bits will now be described according to various example embodiments of the present invention. Because a memristor cell usually represents a weight by conductance which is the inverse of resistance, and cell resistance generally has a finite range and is always positive, it implies that the cell conductance is also positive and has a finite range. However, this does not directly map proportionally to a signed integer which can be negative. To address this issue, according to various example embodiments, a middle reference conductance level may be defined and subtracted from the conductance of the cell. Such a middle reference conductance level can be mapped to a state value in the middle of the range of allowable states in the cell. For example, for a native 4-bit memristor cell, which can nominally take on integer state values from 0 to  $2^4 - 1$  inclusive, the nominal (i.e., intended) middle reference state value may be  $2^3$  (or more generally  $2^{l-1}$  for an  $l$ -bit memristor cell) and after subtraction, the cell may represent a nominal integer range of  $-2^3$  to  $2^3 - 1$  inclusive, which is also the same range as in 2's complement notation. The term "nominal" or the like, such as "nominally", used herein in the context of state value refer to the intended state value. However, the actual written value may deviate slightly, with the deviation being an error. Such an error may generally be within (-0.5, 0.5), because if the error has a bigger range, the achievable bit-precision of the cell would also be

less correspondingly. It will be appreciated by a person skilled in the art that  $2^3$  is not the only one choice of nominal middle reference state, for example, there are other choices that may be more suitable depending on the usage scenario, such as the required number range for the synaptic weight. In particular, when the cell's native precision is low, the use of  $2^{l-1}$  as reference may introduce more and more asymmetry in range length between the positive states and negative states. The raw states of an  $l$ -bit memristor cell has a range of  $(-0.5, 2^l - 0.5)$  after including a noise range of  $(-0.5, 0.5)$ . Accordingly, to produce a fully symmetric range length for positive and negative states, various example embodiments provide a reference value that is at the middle of the noise-included raw state range, that is,  $ref = \frac{2^l - 0.5 + -0.5}{2} = 2^{l-1} - 0.5$ , which can better utilize the dynamic range of the lower-order-bits cells (after reference subtraction) by making them symmetric to zero. However, the achieved synaptic range, which may be  $(-2^{m-1}, 2^{m-1})$  with noise or  $(-2^{m-1} + 0.5, 2^{m-1} - 0.5)$  without noise, may not be fully compatible with 2's complement range of  $[-2^{m-1}, 2^{m-1} - 1]$  without noise, but it may be sufficient for training framework, such as DoReFa described in Shuchang Zhou et al, *DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients*, arXiv:1606.06160, Feb 2018, where the magnitude of weight is clipped to  $2^{m-1} - 1$ .

**[0088]** FIG. 10 depicts a table (Table 2) showing examples of representing signed 4-bit using 2-bit cells with cell-stitching, according to various example embodiments of the present invention. In Table 2, 0 (-8) means the unsigned state equivalent is 0, with the signed value being -8, by using 8 (of 4-bit) or equivalently  $2^1$  (of 2-bit) as a middle reference level, which is what is used in 2's complement notation.  $\epsilon$  takes into account both the deviation of raw state as well as of middle reference. This is because, according to various example embodiments, the middle reference may be implemented as a normal cell (in a shared column) and to leverage on analog computing, so the signed state value is linked with the actual middle reference.  $\epsilon$  or  $\epsilon_{12}$  denote the error introduced by the LSBs cell, which may also be the final error.

**[0089]** In particular, Table 2 illustrates the results if the desired state is signed by comparing against Table 1. The MSBs cell may be configured to become signed by adopting a nominal middle reference state, which in this example is  $2^1$ , so that it is compatible with 2's complement notation and range. For example, to represent a signed 4-bit value of 7 (equivalent to 15 in unsigned 4-bit with 2's complement), the MSBs cell's nominal center raw state should be  $15 \times 2^{-2} = 3.75$ , but in this example, the actual written state is 3.4 and the actual middle reference after writing to a shared memristor cell is 2.1 instead of 2. This results in the MSBs cell actually representing  $(3.4 - 2.1) \times 2^2 = 5.2$ , and hence  $5.2 - 7 = -1.8$  is the synaptic level error from the MSBs

cell. Therefore, according to various example embodiments, the LSBs cell may be programmed to encode a signed value of 1.8 to cancel out this error, which corresponds to a raw nominal center state of  $1.8 + 1.5 = 3.3$ . In this example, the actual written center raw state is 2.9, and the actual written middle reference is 1.6 instead of 1.5, resulting in an actual signed value of  $2.9 - 1.6 = 1.3$  instead of 1.8, and the final synaptic error is  $-1.8 + 1.3 = -0.5$ .

[0090] It will be appreciated by a person skilled in the art that the choice of  $2^1$  as nominal middle reference for the MSBs cell is due to considerations for simplified compatibility with 2's complement notation and range, where a signed 4-bit has a nominal range of  $[-8, 7]$ , and after adding noise may be  $(-8.5, 7.5)$ . In such a case, a nominal raw state of 0 in the MSBs cell may have a state range of  $(-0.5, 0.5)$  after including error, which corresponds to synaptic level range of  $[(-0.5, 0.5) \times 2^1] \times 2^2 = (-10, -6)$ . If it is desired to support 2's complement notation and range, but instead make each raw nominal state of the MSBs cell after noise map exactly to the same power of 2 number of synaptic-level states, for example, for  $m=4$  and  $l_0 = 2$ -bit for the MSBs cell, the MSB cell has 4 raw nominal states: 0, 1, 2, 3, and 0 should cover the synaptic states  $\{-8, -7, -6, -5\}$ , and 1 should cover  $\{-4, -3, -2, -1\}$ , and 2 should cover  $\{0, 1, 2, 3\}$ , and 3 should cover  $\{4, 5, 6, 7\}$ . With noise, 0 should cover synaptic range  $(-8.5, -4.5)$ , 1 cover  $(-4.5, -0.5)$ , 2 cover  $(-0.5, 3.5)$  and 3 cover  $(3.5, 7.5)$ . Based on these constraints, various example embodiments derive that a preferred reference level may be  $ref = 2^{l_0-1} - 0.5 \times (1 - 2^{e_0})$ , where  $e_0 = l_0 - m$ . Therefore, according to various example embodiments, the nominal middle reference may be  $2^{l_0-1} - 0.5 \times (1 - 2^{e_0}) = 2^{2-1} - 0.5 \times (1 - 2^{-2}) = 1.625$ , which relaxes some requirements on lower order cells when compensating for errors in MSB cells. In various example embodiments, all lower order (including LSBs) cells may use a nominal middle reference of  $2^{l_j-1} - 0.5$ , so that it can represent a range that is symmetric to zero to maximize its chance to compensate either positive or negative error in higher order cells. If instead it is desired to support a number range that is symmetric with respect to zero, which is not 2's complement's range, then nominal middle reference may be  $2^{l_j-1} - 0.5$  from the MSBs cell to the LSBs cell. Therefore, according to various example embodiments, for the MSBs cell and/or the LSBs cell, the intended reference value is offset from a middle value of an intended state range of the cell to compensate for write noise associated with cell.

[0091] The precise control nominal center state may be feasible depending on the underlying memristor technology. For example, in Peng Yao et al., *Fully hardware-implemented memristor convolutional neural network*, Nature, Volume 577, 30 Jan 2020, up to 500 programming pulses were used to program to one of 32 states in a type of RRAM cells,

where 500 is much more fine-grained than 32, and demonstrates that the nominal center state can be more fine-grained than 32 states.

**[0092]** Various example embodiments note that the precise control of nominal center state is coupled with the fact that the noise is generally limited to  $(-0.5, 0.5)$  around it. Therefore, even if such control is not so precise or does not even exist at the programming circuit level, as long as the noise can be limited to  $(-0.5, 0.5)$  of the nominal center state, the cell-stitching method according to various example embodiments can operate properly. This noise limit requirement can for example be supported by the cell's write() feature intrinsically if the cell is intrinsically capable of a relatively precise nominal center state, such as in the above-mentioned Peng Yao reference, or if a write-verify scheme is contained within its write() feature to keep writing until the noise limit requirement is met.

**[0093]** By way of an example only for illustration purpose and without limitation, an example pseudo code for implementing the cell-stitching method to represent signed integer is presented below, according to various example embodiments of the present invention.

```

15 // global variables
const int B, S[B]; // B >= 2, S[j] denote  $S_j$ , bit overlap of cell  $C_j$  with  $C_{j-1}$ , = 0 or 1, and  $S[0] = 0$ 
int l[B]; // l[j] denote  $l_j$ , the available bit-precision for cell  $C_j$  which can support  $2^{l_j}$  states;
20 int e[B]; // e[j] denote  $e_j$ , the scaling-down exponent for  $C_j$ 
void cell_stitching_init() { // called once globally at beginning
    for (int j = 0; j < B; j++)
         $e_j = \sum_{k=0}^j (l_k - S_k) - m$ ; //  $S_0 = 0$ , so  $e_0 = l_0 - m$ , and  $e_{j+1} = l_{j+1} - S_{j+1} + e_j$ 
    }
25 void cell_stitching_program(C[B], int w : m) { // stitch B memristor cells to represent w
    // C[j] denotes  $C_j$ , the j-th significant-bits memristor cell with j=0 being most-significant-bits
    // and j < B; weight w is signed m-bit
    for (int j=0; j < B; j++) {
30          $E_j = w - \text{cell\_stitching\_read}(C[j], j)$ ; //  $E_j$  denotes the remaining synaptic-level error
        // after writing  $C_0$  through  $C_{j-1}$ ; so  $E_0 = w$ 
        write( $C_j, 2^{e_j} * E_j + \text{ref}(j)$ ); // try to represent the remaining error with  $C_j$ 
    }
35 }

// reads the equivalent synaptic level weight represented by C[B] from  $C_0$  to  $C_{j-1}$ 
float cell_stitching_read(C[B], int j /* = B */) { // j is optional parameter defaulting to B
    return  $\sum_{k=0}^{j-1} 2^{-e_k} * (\hat{w}(C_k) - \text{ref}(k))$ ; //  $\hat{w}(C_k)$  denotes the actual value written into  $C_k$ ,
40     read by the sensing circuit such as ADC, ref(k) denotes the middle reference state value for
     $C_k$ ; so when j = 0, it returns 0
}

```

```

float ref(int j) {
    #if defined(STANDARD_TWOS_COMPLEMENT)
        // this is more compatible with 2's complement coding, but not so balanced when m and l are
    5    very small, therefore resulting in larger error range
        if (j == 0)
            return  $2^{l_j-1}$ ;
        else // j > 0
            return  $2^{l_j-1} - 0.5$ ;
    10    #elif defined(IDEAL_CENTERED_TWOS_COMPLEMENT)
        // this is the ideal and preferred mode; now, if the actual written reference is slightly off, the
        error will be bigger than (-0.5, 0.5), and some bit-overlap may be needed, OR we need to control
        error in the main cell so that actual error including error of reference cell is still correctable by
        lower order cells.
    15    if (j == 0)
        return  $2^{l_j-1} - 0.5 * (1 - 2^{e_j})$ ; // this is for the MSBs cell only, and it is for compatibility
        with 2's complement where its number range  $[-2^m, 2^m-1]$  is not symmetric, so when
        mapping MSB cells to cover higher precision weight, the reference needs to be shifted
        slightly so that each MSB cell original discrete state can cover exactly  $2^{-e_0}$  states of the
    20    higher precision weight.
        else // j > 0
            return  $2^{l_j-1} - 0.5$ ;
        #elif defined(IDEAL_CENTERED_SYMMETRIC)
        // this is where the intended weight range is symmetric with respect to zero, this means the
    25    nominal raw state range is  $[0, 2^{l_j-1}]$  and only includes integer values, or after including the
        general error limit of (-0.5, 0.5), the state range becomes  $(-0.5, 2^{l_j-0.5})$ , and after subtracting
        the reference below, the signed state range becomes  $(-2^{l_j}, 2^{l_j})$ , which is a totally symmetric
        range.
        return  $2^{l_j-1} - 0.5$ ;
    30    #else
        return some reference value; // for example it can be the actual achieved/written raw state
        value of nominal/intended reference value in
        IDEAL_CENTERED_TWOS_COMPLEMENT or IDEAL_CENTERED_SYMMETRIC
        or STANDARD_TWOS_COMPLEMENT, as sensed by the ADC, when the reference value
    35    is written into a memristor cell. Note the reference cell itself will be treated as an unsigned
        cell, but the writing of the reference cell should be treated carefully, so that its
        achieved/written raw state value is sufficiently close to the intended/nominal value. For
        example, if intended/nominal middle reference is  $2^{l_j-1} - 0.5$ , and the achieved value is
    40     $2^{l_j-1} - 0.4$ , the error is 0.1, which is probably acceptable, whereas if achieved value is
         $2^{l_j-1} + 0.5$ , the error is 1.0, which is likely too large for cell-stitching to compensate. This
        implies multiple write-verify operations may be needed during writing of the middle
        reference cell, until the written value is sufficiently close to its nominal value.
        // One criterion for judging whether it is sufficiently close is, the gap (if any) in the intended
        range that cannot be covered by the resulting signed range with the achieved middle
    45    reference cell should not be longer than what the lower order cells are capable of
        compensating (after considering bit-overlap if any).
        // For example, for  $m=4$ ,  $l_0 = l_1 = 2$ , no bit-overlap, representing 2's complement where
        intended range is  $[-8, 7]$ , for IDEAL_CENTERED_TWOS_COMPLEMENT setting,
        nominal middle reference is  $2^{l_0-1} - 0.5 * (1 - 2^{e_0}) = 1.625$  where  $e_0 = l_0 - m = -2$ ,

```

and given a 2-bit cell raw state range with noise is  $(-0.5, 3.5)$ , the MSBs cell on its own can represent a synaptic range of  $[(-0.5, 3.5) - 1.625] * 2^{-e_0} = (-8.5, 7.5)$ . If the achieved state in the (shared column's) middle reference cell is 1.5 instead of 1.625, the MSBs cell now represents a synaptic range of  $[(-0.5, 3.5) - 1.5] * 2^{-e_0} = (-8, 8)$ , which is still ok to cover the intended range  $[-8, 7]$ . But if the achieved middle reference is 0.9 instead, the MSBs cell will represent a synaptic range of  $[(-0.5, 3.5) - 0.9] * 2^{-e_0} = (-5.6, 10.4)$ , and for the intended range  $[-8, 7]$ , there is a gap of  $[-8, -5.6]$  of length 2.4, which is bigger than the  $(-2, 2)$  signed range of lower order cells, so there is a danger that the error cannot be sufficiently compensated.

10 // For lower order cells, since middle reference cells are also used and there may be some error between nominal and achieved middle reference values, multiple verify operations may be needed in the write() feature to in effect keep the final synaptic error under the requirement target.

15 #endif  
}

void write( $C_j$ , float w1)  
{  
#if defined(QUANTIZED\_CELL)  
20 // this is the kind of cell whose nominal state values are quantized, but actual written state will have some error with respect to the nominal state with the error following some distribution of a finite range where typical range is  $(-0.5, 0.5)$ . w1 should be a floating-point number in the range  $(-2^{l_j-1}, 2^{l_j-1})$ , and w1 should be first converted to unsigned and then be truncated to an integer, i.e. quantized to its nearest nominal state.  
25 return write\_quantized\_cell( $C_j$ ,  $\lfloor w1 + 2^{l_j-1} \rfloor$ );  
#elif defined(ANALOG\_CELL)  
// this is the kind of cell where we can program any analog value w1 (with a certain range) to the cell, and the actual state value will have some error with respect to w1 with the error following some distribution of a finite range where typical range is  $(-0.5, 0.5)$   
30 return write\_analog\_cell( $C_j$ , w1);  
#endif  
// write( $C_j$ , float w1) is the basic primitive function, it writes a signed floating-point value into memristor cell  $C_j$ ; it can be a single physical write operation, or can be multiple physical write operations using techniques like write-verify to repeat until the error is within desired range.  
35

**[0094]** In various example embodiments, based on the above-described example pseudo code and comparing with the cell-stitching method described hereinbefore, when the intended synaptic weight  $w$  is an  $m$ -bit integer and  $j$ -th significant-bits memristor cell is  $l_j$ -bit, the following refinements may be provided (which may be referred to herein as the second or refined cell-stitching method):

1.  $f_j = 2^{-e_j}$ , where  $e_0 = l_0 - m$ , and  $e_{j+1} = l_{j+1} - S_{j+1} + e_j$ , and  $l_j$  denotes the available bit-precision for  $C_j$  which can support  $2^{l_j}$  states, and  $S_j = 0$  or 1 is a predetermined bit-overlap for each  $j$ , and  $S_0 = 0$ ;

2. A reference level is used to shift the supported range of each  $C_j$ , where  $\tilde{w}(C_j) = \hat{w}(C_j) - \text{ref}(j)$ , and  $\hat{w}(C_j)$  denotes the achieved written raw state of  $C_j$ , and writing  $E_j * f_j^{-1}$  in shifted range to  $C_j$  means writing a raw intended/nominal state of  $E_j * f_j^{-1} + \text{ref}(j)$  to  $C_j$ ;
3. During inference,  $I_{\text{sum\_main},j} = \sum_{i=0}^{n-1} c_{i,j} * v_{i,j}$ , where  $c_{i,j}$  denotes conductance of  $j$ -th most-significant-bits memristor cell  $C_j$  for  $i$ -th synapse;
4. Obtain the reference weighted sum for the neuron, which is defined as  $Z_{\text{sum\_ref}} = \sum_{i=0}^{n-1} x_i * (\sum_{j=0}^{B-1} f_j * \text{ref}(j))$ ;
5. Compute final weighted sum  $\sum_{i=0}^{n-1} w_i * x_i$  as  $Z_{\text{sum}} = Z_{\text{sum\_main}} - Z_{\text{sum\_ref}}$ .

## 10 Hardware Implementations

**[0095]** In various example embodiments, the nominal middle reference state value may be stored in a digital form (e.g., in SRAM or CMOS registers or hardwired in CMOS), which can avoid errors in retrieval, but it will require digital computation (e.g., either MAC or SOP) over all input axons to calculate the “middle reference weighted sum”, such as defined as  $Z_{\text{sum\_ref}} = \sum_{i=0}^{n-1} x_i * (\sum_{j=0}^{B-1} f_j * \text{ref}(j))$ , where  $x_i$  are the axon inputs (i.e., input activations) to the neuron, and  $\text{ref}(j)$  is the nominal reference state value (this formula works for any desired reference level, including 0) or level for the  $j$ -th significant-bits memristor cell, and  $f_j$  denotes the scaling for the  $j$ -th cell. This computation requires both time and energy overhead, but with the advantage that this “middle reference weighted sum” can be reused for all neurons in a memristor array, such as a crossbar matrix for the same axon input pattern.

**[0096]** In various example embodiments, in contrast to the above-mentioned digital form and computation approach, as illustrated in FIGs. 11A and 11B, to make the computation of the “middle reference weighted sum”  $Z_{\text{sum\_ref}}$  more efficient, the nominal middle reference state value may be stored in a shared column of the main memristor array (such as a crossbar array), such that the “middle reference weighted sum” can be calculated based on analog computing in the same manner as  $Y_{\text{sum\_main},j}$ ,  $Y_{\text{sum\_main}}$  and  $Z_{\text{sum\_main}}$ , where  $Y_{\text{sum\_ref}} = \sum_{j=0}^{B-1} f_j * Y_{\text{sum\_ref},j}$  and  $Z_{\text{sum\_ref}} = Y_{\text{sum\_ref}} * F$  by sensing the summed current on the shared reference column for each corresponding  $j$ , and then stored in digital form once converted by ADC or other sense amplifier circuit, for reuse with all neurons. In this regard, the time and energy spent is the same as with a normal neuron which is very fast and low energy with analog computing, and similar to the digital form and computation approach, such overhead is shared over all neurons, making the amortized overhead very small, in terms of both energy and timing.

[0097] In various example embodiments, the subtraction of the “middle reference weighted sum” from the weighted sum of the main cells are done digitally, e.g., via digital adder/subtractor circuit. Various example embodiments note that such an approach may result in a less efficient utilization of the dynamic range of the ADC, for example when negative weighted sum is always truncated/clipped to 0 as in the case of DoReFa low precision training algorithm as disclosed in the above-mentioned Shuchang Zhou reference. Therefore, as illustrated in FIG. 13, in various example embodiments, an analog current subtraction circuit 1300 may be used to subtract the summed current of reference memristor cells from the summed current of the main memristor cells for the  $j$ -th cell in a virtual, stitched synapse at  $j$ -th cycle prior to ADC conversion, so as to subtract the “middle reference weighted sum” in analog. A tradeoff with such an approach is that the reference column (which can be either shared as described hereinbefore according to various embodiments, or non-shared) may have to keep running and passing current for each neuron’s computation, as opposed to computing only once and reuse for all neurons for the same axon inputs, so energy efficiency may decrease as a tradeoff.

[0098] FIGs. 11A and 11B depict schematic flow diagrams illustrating processing steps for implementing the above-described cell-stitching during synapse programming and during inference, respectively, according to various example embodiments of the present invention.

[0099] FIG. 12 depicts a schematic drawing illustrating an example implementation or architecture of cell-stitching during inference, according to various example embodiments of the present invention, in view of the general analog computing architecture shown in FIG. 1, the cell-stitching method as illustrated in FIG. 7C, and the above-described example pseudo code for implementing the cell-stitching method, the steps described in relation to the above-described first cell-stitching method and the steps described in relation to the above-described second cell-stitching method. In FIG. 12, the  $B$  ( $=2$  in the example) memristor cell columns boxed by the rounded dashed rectangle 1210 may be used as reference cells to store the various  $ref(j)$  ( $0 \leq j < B$ ). In the example architecture, each reference cell is shared with the other corresponding memristor cells (i.e., corresponding main cells) on its corresponding row of the crossbar array, and the  $B$  columns of reference cells, thus form  $B$  shared reference columns (e.g., corresponding to the group of reference memory cell columns as described hereinbefore according to various embodiments). For example, as seen in FIG. 12, each group of  $B$  main cells (e.g.,  $B$  stitched cells per synapse) on a row are linked (or coupled) to the  $B$  columns of reference cells on the row, respectively. For example, on a row, the MSBs main cell of each

group of B main cells on the row is coupled to the same corresponding MSBs reference cell on the row, and the LSBs main cell of each group of B main cells on the row is coupled to the same corresponding LSBs reference cell on the row. As an illustrative example, if B=2, and the main cell array is  $256 \times 256$ , this means that 256 axon inputs and  $256/B = 128$  neurons can be supported for such a neural core. For each row (out of 256), 128 MSBs cells share one MSBs reference cell, and the other 128 LSBs cells share one LSBs reference cell. Hence, after including the reference cells, the cell array becomes  $256 \times (256 + 2)$  in size, with the B=2 reference columns being the shared reference columns.

**[00100]** As mentioned hereinbefore, when a reference level is introduced to handle negative

weights, the derivation of scaling factor F can be modified to  $F = \frac{\sum_{i=0}^{n-1} (w_i + ref_i) * x_i}{ADC\_output\_main}$ . When cell-

stitching uses reference level, the derivation of scaling factor F may be further refined to  $F =$

$$\frac{\sum_{i=0}^{n-1} (\tilde{w}_i(c_j) + ref_i(j)) * x_i}{ADC\_output\_main,j} = \frac{\sum_{i=0}^{n-1} \hat{w}_i(c_j) * x_i}{ADC\_output\_main,j}, \text{ where } \tilde{w}_i(c_j) = \hat{w}_i(c_j) - ref_i(j) \text{ denotes the actual signed}$$

value of cell  $c_j$  with the reference, and  $\hat{w}_i(c_j)$  denotes the actual raw value written (written state

value) to cell  $c_j$ , and  $ref_i(j)$  is the more generalized version of  $ref(j)$  and denotes the reference

level for  $c_j$  (which can be stored in a reference cell or as a digital value), all three variables

subscripted to denote for the current neuron i, and  $ADC\_output\_main,j$  denotes the ADC output

for j-th significant-bits column of neuron i, and is equivalent to  $Y_{sum\_main,j}$  in FIGs. 11A, 11B

and 12. In various example embodiments, instead,  $F = \frac{\sum_{i=0}^{n-1} ref_i(j) * x_i}{ADC\_output\_ref,j}$  may be used, where

$ADC\_output\_ref,j$  denotes the ADC output for j-th significant-bits reference column of neuron

i (which may be shared multiple j-th significant-bits main cell columns), and  $ref_i(j)$  refers to the

actual reference level (written state value) for  $c_j$  that is stored in a reference cell as opposed to

a digital value. In addition, all B stitched cells may be enumerated and use  $F =$

$$\frac{\sum_{j=0}^{B-1} f_j \sum_{i=0}^{n-1} (\tilde{w}_i(c_j) + ref_i(j)) * x_i}{ADC\_output\_main} = \frac{\sum_{j=0}^{B-1} f_j \sum_{i=0}^{n-1} \hat{w}_i(c_j) * x_i}{ADC\_output\_main}, \text{ where } ADC\_output\_main = \sum_{j=0}^{B-1} f_j * ADC\_output\_main,j,$$

which is equivalent to  $Y_{sum\_main}$  in FIGs. 11A, 11B and 12. The reference

weighted sum may also be subtracted and use  $F = \frac{\sum_{j=0}^{B-1} f_j \sum_{i=0}^{n-1} \tilde{w}_i(c_j) * x_i}{ADC\_output\_main - ADC\_output\_ref} =$

$$\frac{\sum_{j=0}^{B-1} f_j \sum_{i=0}^{n-1} (\hat{w}_i(c_j) - ref_i(j)) * x_i}{ADC\_output\_main - ADC\_output\_ref}, \text{ where } ADC\_output\_ref = \sum_{j=0}^{B-1} f_j * ADC\_output\_ref,j \text{ which is}$$

proportional to  $Z_{sum\_ref} = \sum_{j=0}^{B-1} f_j \sum_{i=0}^{n-1} ref_i(j) * x_i$  in FIGs. 11A, 11B and 12, and

$ADC\_output\_main - ADC\_output\_ref$  is proportional to  $Z_{sum}$  in FIGs. 11A, 11B and 12.

Various example embodiments found that the above last formula is particularly useful when

there are certain fixed offsets in the system, such as ADC offset. As described hereinbefore, while F can be derived based on one of the above-mentioned formulas, such formulas may be used in conjunction with averaging or linear regression to improve the robustness in deriving F. More generally, the term *ADC\_output* can refer to any output from the *DIGITIZE()* function in the cell-stitching method according to various example embodiments.

**[00101]** As described above, FIG. 13 depicts an example analog current subtraction circuit 1300 for subtracting the reference cell column current sum from the main cell column in analog domain, prior to digitization via TIA and ADC conversion, according to various example embodiments of the present invention. In particular, FIG. 13 shows an example of how to perform current subtraction in analog domain which can better utilize ADC dynamic range, though at the expense of higher power consumption due to the reference column having to drain current for every main cell column. In FIG. 13, a feedback resistor  $R_b$  may be used to control the (analog) gain factor of TIA in converting input current to the output voltage which can be sensed by the ADC. Typically, the value of  $R_b$  is proportional to this gain.

**[00102]** FIG. 14 depicts an example analog current subtraction circuit 1400 with selectable feedback resistors (e.g., 1x, 8x, 64x) allowing selectable analog amplification factor (e.g., 1x, 8x, 64x) of TIA, according to various example embodiments of the present invention. In particular, FIG. 14 illustrates how a few feedback resistors can be used with select transistors to choose from a number of desired gain factors. For example, using 1x, 8x, 64x of a baseline resistance, a gain of 1x, 8x, and 64x, respectively, can be obtained. The choice of 1x, 8x, 64x allows simple bit-shift operation in digital domain after ADC to convert the ADC output back to a unified numbering range, though other choices of gain factors are also allowed, depending on the application requirement.

**[00103]** Therefore, according to various example embodiments, the use of a number of (e.g., a few) well-chosen feedback resistors improves ADC's operational dynamic range, and this allows one ADC to be reused for a crossbar array, instead of having to implement a separate ADC for each unique dynamic range. For example, verification after cell programming, which has relatively low amount of current, can use a large feedback resistor to provide a large gain, as long as the circuit noise does not become a dominant bottleneck in the measurement accuracy of TIA and ADC. To maximize the useful ADC resolution, the largest gain factor that does not cause dynamic range overflow in the ADC (or generally any circuit implementing the *DIGITIZE()* function) may be chosen. Furthermore, the choice of multiple gain factors can enable support of Batch Normalization (BN) scaling partially in analog, by using the largest

possible gain factor that does not lead to ADC dynamic range overflow, and then the residual BN scaling factor (i.e., the remaining factor not included by analog BN scaling) can be multiplied with the ADC output in digital domain to complete BN scaling, followed by BN offset subtraction in digital domain, and so on. For example, if BN in effect (i.e., equivalently) requires scaling the output by 10x compared to the baseline 1x gain configuration, then the 8x gain factor (i.e., with 8x feedback resistor) may be chosen, and then multiply the ADC output with residual BN scaling factor of  $10/8 = 1.25$  in digital domain. Since the residual BN scaling in digital domain is done per neuron instead of per synapse, the overhead may be quite limited, while it can increase ADC operational dynamic range by 8 times or 3-bit. Without this approach, the ADC may require an additional 3-4 bit resolution to support BN scaling completely in digital domain without noticeable loss in precision of weighted sum output, resulting in a substantial increase in ADC's overhead in both area and power/energy. In various example embodiments, for simplicity of implementation, the residual BN scaling factor may be subsumed (i.e. absorbed) into the scaling factor F, though this means F may be different for each neuron because BN coefficients may be different for each neuron. This however, does not necessarily increase storage overhead, because even if a common scaling factor F is used first followed by residual BN scaling, both in digital domain, each residual BN scaling factor per neuron may still need to be stored.

## 20 ***Experimental Results***

**[00104]** An evaluation of the cell-stitching method according to various example embodiments was performed on the golden standard of ILSVRC-2012 (i.e., ImageNet 1000-class problem), based upon the DoReFa low precision neural network training framework. For AlexNet, when training and evaluated with 32-bit floating-point (FP32) precision, Top-1 error obtained was 40.3%, whereas a DoReFa model of 8-bit weight and 8-bit in activation trained with 8-bit gradient (though not important during inference), achieved 42%. If the same model were ported to 4-bit memristor cells, Top-1 error increases significantly to 52%. If naïvely concatenating two 4-bit cells, Top-1 error remains at 51.97%, confirming that the error/noise in MSBs cell dominates, rendering the LSBs cell irrelevant. In comparison, stitching two 4-bit cells with 1-bit overlap according to various example embodiments achieves equivalently 7-bit precision, and achieves Top-1 error of 42.19%, only 0.19% higher than the baseline.

**[00105]** FIGs. 15A to 15D show evaluation results of cell-stitching on AlexNet DoReFa for DoReFa 8,8,8, DoReFa 6,6,32, DoReFa 5,5,32 and DoReFa 4,4,32, respectively. For example,

in the table (Table 3B) shown in FIG. 15B, it can be observed that the baseline error is slightly better (lower) than the stitching of two 3-bit cells with no bit overlap, because the stitched synapse has a statistical error range of  $(-0.5, 0.5)$ , whereas the baseline weight is an integer and has no error. In Table 3D, baseline is 4-bit weight, and stitching two 2-bit cells together is much better than single 2-bit cell, but error is still slightly higher than baseline. If DoReFa is trained with a statistical noise as opposed to quantization noise in the weights, then it is possible the difference in performance between baseline and stitched cases to be much smaller.

**[00106]** Accordingly, a method of achieving higher synaptic precision in hardware implementation of neural networks according to various example embodiments is provided. In various example embodiments, the method of programming a synaptic memory array of a neural processing core is based on stitching  $B$  ( $B \geq 2$ ) memristor cells to form a virtual synapse, and may include the following steps:

1.  $w$  denotes the intended synaptic weight, and initialize  $j = 0$ , where  $C_k$  denotes the  $k$ -th significant-bits memristor cell and has a corresponding scaling factor  $f_k$ , and  $C_0$  means the most-significant-bits (MSBs) cell;

2. For  $k = 0$  to  $j-1$ , measure the achieved written value (written state value) of  $C_k$ , denoted as  $\tilde{w}(C_k)$ , and multiply each  $\tilde{w}(C_k)$  with their respective corresponding scaling factor  $f_k$  and sum together to derive the corresponding synaptic-level value (obtained synaptic weight value of the group of  $B$  memristor cells)  $w' = \sum_{k=0}^{j-1} f_k * \tilde{w}(C_k)$ ; Note that  $w' = 0$  when  $j=0$ ;

3. Let  $E_j = w - w'$ , which represents the residual synaptic weight (residual synaptic weight value) after stitching  $C_0$  to  $C_{j-1}$ ;

4. Represent the most significant component of  $E_j$  in  $C_j$  (i.e., immediately succeeding cell), by writing a nominal/intended value of  $E_j * f_j^{-1}$  to  $C_j$ , where the memristor cell write operation limits the writing error to be within what  $C_{j+1}$  is able to compensate during cell-stitching, which by default or when  $j+1 \geq B$  may be within  $(-0.5, 0.5)$  as an example; and

5. let  $j = j + 1$  (i.e., increment  $j$ ), then repeat step (2) to step (5) in a loop while  $j < B$ ;

**[00107]** There is also provided a corresponding method of performing neural network inference, including the following steps:

1. During inference, measure the summed cell current  $I_{sum\_main,j}$  for a given neuron for its  $j$ -th significant-bits cells in the  $j$ -th cycle, by sensing the digitized summed current via the digitization output  $Y_{sum\_main,j} = DIGITIZE(I_{sum\_main,j})$  and derive the neuron's total

digitized summed current as  $Y_{sum\_main} = \sum_{j=0}^{B-1} f_j * Y_{sum\_main,j}$ , where the function  $DIGITIZE()$  is an electrical circuit that converts an analog input current into a digitized form; and

2. Compute  $Z_{sum\_main} = Y_{sum\_main} * F$ , where  $F$  is a predetermined scaling factor for converting the digitized summed current into weighted sum or a range suitable for neural network computation in hardware;

**[00108]** In various example embodiments, the following properties may be provided or implemented:

1.  $w$  is  $m$ -bit integer,  $f_j = 2^{-e_j}$ , where  $e_0 = l_0 - m$ , and  $e_{j+1} = l_{j+1} - S_{j+1} + e_j$ , and  $l_j$  denotes the available bit-precision for  $C_j$  which can support  $2^{l_j}$  states, and  $S_j = 0$  or  $1$  is a predetermined bit-overlap for each  $j$ , and  $S_0 = 0$ ;

2. A reference level is used to shift the supported range of each  $C_j$ , where  $\tilde{w}(C_j) = \hat{w}(C_j) - ref(j)$ , and  $\hat{w}(C_j)$  denotes the achieved written raw state of  $C_j$ , and writing  $E_j * f_j^{-1}$  in shifted range to  $C_j$  means writing a raw intended/nominal state of  $E_j * f_j^{-1} + ref(j)$  to  $C_j$ ;

3. During inference,  $I_{sum\_main,j} = \sum_{i=0}^{n-1} c_{i,j} * v_{i,j}$ , where  $c_{i,j}$  denotes conductance of  $j$ -th most-significant-bits memristor cell  $C_j$  for  $i$ -th synapse;

4. Obtain the reference weighted sum for the said neuron, which is defined as  $Z_{sum\_ref} = \sum_{i=0}^{n-1} x_i * (\sum_{j=0}^{B-1} f_j * ref(j))$ ;

5. Compute final weighted sum  $\sum_{i=0}^{n-1} w_i * x_i$  as  $Z_{sum} = Z_{sum\_main} - Z_{sum\_ref}$ .

**[00109]** In various example embodiments,  $w$  is unsigned, and where nominally  $ref(j) = 2^{l_j-1} - 0.5$  for  $j > 0$ , and  $ref(0) = 0$ .

**[00110]** In various example embodiments,  $w$  is signed, and where nominally  $ref(j) = 2^{l_j-1} - 0.5$  for  $j > 0$ , and  $ref(0) = 2^{l_0-1} - 0.5 * (1 - 2^{e_0})$ .

**[00111]** In various example embodiments,  $w$  is signed, and where nominally  $ref(j) = 2^{l_j-1} - 0.5$  for  $0 \leq j < B$ .

**[00112]** In various example embodiments,  $ref(j)$  is the actual achieved/written value of a reference memristor cell whose nominal/intended state is any one of the above-mentioned nominal  $ref(j)$ .

**[00113]** In various example embodiments, the above-mentioned reference memristor cell is shared with multiple corresponding memristor cells (e.g., MSBs reference cell is shared with multiple MSBs main cells (i.e., corresponding main cells) of multiple groups of main cells) in the same row of a memristor array as described hereinbefore.

[00114] In various example embodiments, the reference weighted sum  $Z_{sum\_ref}$  is computed in the same manner as  $Y_{sum\_main,j}$ ,  $Y_{sum\_main}$  and  $Z_{sum\_main}$ , where  $Y_{sum\_ref} = \sum_{j=0}^{B-1} f_j * Y_{sum\_ref,j}$  and  $Z_{sum\_ref} = Y_{sum\_ref} * F$  by sensing the summed current on the shared reference column for each corresponding  $j$ .

5 [00115] In various example embodiments, an analog current subtraction circuit is used to subtract the summed current of reference memristor cells from the summed current of the main memristor cells prior to ADC conversion;

[00116] In various example embodiments, each  $ref(j)$  is stored digitally and shared with multiple corresponding memristor cells of a memristor array, and the reference weighted sum  
10  $Z_{sum\_ref}$  is computed digitally as described above, and reused for all neurons in the memristor for the same given axon input pattern;

[00117] In various example embodiments, each memristor cell comprises two memristor devices, with a first device (e.g., device 1) representing the positive component, and a second device (e.g., device 2) representing the negative component, and the difference in the above-  
15 mentioned two devices' conductance is designated as the conductance of the cell and is used to represent a signed number, and during inference the difference in the above-mentioned two devices' current are treated as the above-mentioned memristor cell's cell current.

[00118] In various example embodiments, the circuit implementing the *DIGITIZE()* function supports a set of selectable, predetermined analog gain factors.

20 [00119] In various example embodiments, the largest gain factor among the above-mentioned set of gain factors that does not cause dynamic range overflow in the circuit implementing the *DIGITIZE()* function is chosen.

[00120] In various example embodiments, the circuit implementing the *DIGITIZE()* function is implemented by cascading a TIA and an ADC, and the gain factors are implemented  
25 by feedback resistors.

[00121] In various example embodiments, the largest gain factor is chosen to implement BN scaling partially in analog and the residual BN scaling implemented with multiplication in digital domain.

[00122] In various example embodiments, the above-mentioned scaling factor  $F$  is derived  
30 based on one of the following formulas:

1.  $F = \frac{\sum_{i=0}^{n-1} (\tilde{w}_i(c_j) + ref_i(j)) * x_i}{ADC\_output_{main,j}} = \frac{\sum_{i=0}^{n-1} \hat{w}_i(c_j) * x_i}{ADC\_output_{main,j}}$ , where  $ADC\_output_{main,j}$  denotes the  $DIGITIZE()$  function output for j-th significant-bits column of neuron i, and is equivalent to  $Y_{sum\_main,j}$ ;

2.  $F = \frac{\sum_{i=0}^{n-1} ref_i(j) * x_i}{ADC\_output_{ref,j}}$ , where  $ADC\_output_{ref,j}$  denotes the  $DIGITIZE()$  function output for j-th significant-bits reference column of neuron i;

3.  $F = \frac{\sum_{j=0}^{B-1} f_j \sum_{i=0}^{n-1} (\tilde{w}_i(c_j) + ref_i(j)) * x_i}{ADC\_output_{main}} = \frac{\sum_{j=0}^{B-1} f_j \sum_{i=0}^{n-1} \hat{w}_i(c_j) * x_i}{ADC\_output_{main}}$ , where  $ADC\_output_{main} = \sum_{j=0}^{B-1} f_j * ADC\_output_{main,j}$ , which is equivalent to  $Y_{sum\_main}$ ;

4.  $F = \frac{\sum_{j=0}^{B-1} f_j \sum_{i=0}^{n-1} \tilde{w}_i(c_j) * x_i}{ADC\_output_{main} - ADC\_output_{ref}} = \frac{\sum_{j=0}^{B-1} f_j \sum_{i=0}^{n-1} (\hat{w}_i(c_j) - ref_i(j)) * x_i}{ADC\_output_{main} - ADC\_output_{ref}}$ , where  $ADC\_output_{ref} = \sum_{j=0}^{B-1} f_j * ADC\_output_{ref,j}$ ;

where  $\tilde{w}_i(c_j) = \hat{w}_i(c_j) - ref_i(j)$  denotes the actual signed value of cell  $c_j$  with the reference, and  $\hat{w}_i(c_j)$  denotes the actual raw value written to cell  $c_j$ , and  $ref_i(j)$  is the more generalized version of  $ref(j)$  and denotes the reference level for  $c_j$ , all three variables subscripted to denote for the current neuron i.

**[00123]** Accordingly, various example embodiments of the present invention provide a cell-stitching method for analog neuromorphic computing, which advantageously achieve higher synaptic weight precision in analog neuromorphic computing, although the underlying memristor cell's precision is not enough. In particular, various example embodiments utilize a lower-order-bits cell to compensate for error in higher-order-bits cell(s), whereby such error in higher-order-bits cell is mostly fixed after writing thereto. In this regard, the error in the higher-order-bits cell may be sensed, and then compensate for the error sensed in the lower-order-bits cell, and so on (iteratively until the last memristor cell in the group of memristor cells representing a synaptic weight of a corresponding synapse of a neural network. For example, the cell-stitching method advantageously uses a lower-order-bits cell to compensate for error in higher-order-bits cell(s) which overcomes pit-falls of the naïve cell concatenation techniques in the art as discussed in the background, thereby enabling analog computing at a truly higher weight precision. In addition, a reference level is provided which enables signed range and weight, and derived good guidance values for the reference level that works more effectively. There is also provided a reference cell (as a shared column) as a reference level, which enables reference weighted sum computation in analog and the reference level can be reused across all neurons in the same row of the crossbar memory array. Furthermore, precise control of nominal

center state is introduced, which improves error compensation efficiency and helps to avoid bit-overlap (e.g., leverage on fine-grained programming control or perform multiple write-verify operations to ensure that the error from the higher-order-bits cell(s) does not exceed the dynamic range of the lower-order-bits cell. Optionally, a bit-overlap between the higher-order-bits cell(s) and the lower-order-bits cell (e.g., as illustrated in FIG. 7C) may be applied to enable stronger error compensation by the lower-order-bits cell, but at loss of bit-width (e.g., 7-bit precision instead of 8-bit precision). Furthermore, analog computing circuitry may also be reused and minimizing change to circuitry. For example, TIA and ADC would have been needed anyway, and just a few feedback resistors in TIA (e.g., 1x, 8x, 64x) add 8-bit dynamic range to the ADC, thereby avoiding separate ADCs for programming and inference, and enables BN (batch normalization) scaling partially in analog (reducing needed ADC resolution). For example, shared reference cell column reuses cell array architecture, and its reference weighted sum can be calculated in analog like the main cells, as described hereinbefore.

**[00124]** In various example embodiments, instead of computing reference weighted sum digitally or in analog and reusing it, an analog current subtraction circuit is used to subtract the reference current sum in analog domain, prior to TIA and ADC conversion, which advantageously enables better utilization of ADC dynamic range (e.g., 1-2 bit may be saved on ADC resolution requirement). However, higher power/energy consumption may be required, since the reference current and subtraction circuit needs to run all the time for all neurons, instead of doing it once and reuse for all neurons. In various example embodiments, the cell-stitching method is compatible with (+/-) device-pair based memristor cell design. For example, for two 3-bit devices (which otherwise would have been called two 2-bit cells in the reference-level based approach), a first device (e.g., device 0) always stores positive component (or 0-state if value is negative), and a second device (e.g., device 1) always stores negative component (or 0-state if value is positive), then the difference of conductance between these two devices can represent a signed 4-bit value and form a signed 4-bit cell. Accordingly, cell-stitching can be applied to four 3-bit devices (as two signed 4-bit cells) to represent signed 8-bit weight (w/o bit-overlap), or signed 7-bit weight (with bit-overlap).

**[00125]** While embodiments of the invention have been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the scope of the invention as defined by the appended claims. The scope of the invention is thus indicated

by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.

**CLAIMS**

What is claimed is:

- 5 1. A method of programming a synaptic memory array of a neural processing core configured to perform neural network computations for a neural network, the method comprising:
- 10 writing an intended synaptic weight value for a group of synaptic memory cells of the synaptic memory array to a first synaptic memory cell of the group of synaptic memory cells of the synaptic memory array;
- obtaining a first residual synaptic weight value associated with the group of synaptic memory cells based on the intended synaptic weight value and an obtained first synaptic weight value of the group of synaptic memory cells, the obtained first synaptic weight value being obtained based on a written state value of the first synaptic memory cell from said writing the
- 15 intended synaptic weight value to the first synaptic memory cell; and
- writing a first intended residual synaptic weight value determined based on the obtained first residual synaptic weight value to a second synaptic memory cell of the group of synaptic memory cells, the second synaptic memory cell being immediately succeeding the first synaptic memory cell in the group of synaptic memory cells, wherein
- 20 the group of synaptic memory cells collectively constitutes a combined synaptic memory cell representing a synaptic weight of a corresponding synapse of the neural network, and
- the group of synaptic memory cells corresponds to a plurality of ordered bits that are ordered from highest order bit(s) to lowest order bit(s), the first synaptic memory cell
- 25 corresponding to the highest order bit(s).
2. The method according to claim 1, wherein said obtaining the first residual synaptic weight value comprises determining a difference between the intended synaptic weight value for the group of synaptic memory cells and the obtained first synaptic weight value of the group
- 30 of synaptic memory cells.

3. The method according to claim 1, further comprises obtaining a first synaptic weight value of the group of synaptic memory cells to produce the obtained first synaptic weight value of the group of synaptic memory cells, comprising:

5 obtaining the written state value of the first synaptic memory cell, the first synaptic memory cell having associated therewith a corresponding scaling factor based on an order thereof in the group of synaptic memory cells; and

multiplying the written state value of the first synaptic memory cell and the corresponding scaling factor to obtain a product thereof,

10 wherein the obtained first synaptic weight value is obtained based on the product.

4. The method according to claim 3, wherein said writing the first intended residual synaptic weight value to the second synaptic memory cell comprises:

15 dividing the first intended residual synaptic weight value by a corresponding scaling factor associated with the second synaptic memory cell to obtain a scaled first intended residual synaptic weight value; and

writing the scaled first intended residual synaptic weight value to the second synaptic memory cell.

5. The method according to claim 1, wherein  
20 the intended synaptic weight value and/or the first intended residual synaptic weight value is a signed value, and

the intended synaptic weight value is written to the first synaptic memory cell with respect to a first intended reference value and/or the first intended residual synaptic weight value is written to the second synaptic memory cell with respect to a second intended reference  
25 value.

6. The method according to claim 5, further comprising:

writing the first intended reference value to a first reference memory cell communicatively coupled to the first synaptic memory cell; and/or

30 writing the second intended reference value to a second reference memory cell communicatively coupled to the second synaptic memory cell.

7. The method according to claim 6, wherein

the first intended reference value is offset from a middle value of an intended state range of the first synaptic memory cell to compensate for write noise associated with the first synaptic memory cell, and/or

5 the second intended reference value is offset from a middle value of an intended state range of the second synaptic memory cell to compensate for write noise associated with the second synaptic memory cell.

8. The method according to claim 1, wherein  
the group of synaptic memory cells comprises three or more synaptic memory cells, and  
10 for each synaptic memory cell from the second synaptic memory cell to a penultimate synaptic memory cell of the group of synaptic memory cells, the method further comprises:

obtaining a second residual synaptic weight value associated with the group of synaptic memory cells based on the intended synaptic weight value and an obtained second synaptic weight value of the group of synaptic memory cells, the obtained second synaptic  
15 weight value being obtained based on written state values of the synaptic memory cell and each preceding synaptic memory cell of the group of synaptic memory cells with respect to the synaptic memory cell; and

writing a second intended residual synaptic weight value determined based on the obtained second residual synaptic weight value to an immediately succeeding synaptic  
20 memory cell of the group of synaptic memory cells with respect to the synaptic memory cell.

9. The method according to claim 8, further comprises, obtaining, for said each synaptic memory cell from the second synaptic memory cell to the penultimate synaptic memory cell of the group of synaptic memory cells, a second synaptic weight value of the group of synaptic  
25 memory cells to produce the obtained second synaptic weight value of the group of synaptic memory cells, comprising:

obtaining the written state values of the synaptic memory cell and said each preceding synaptic memory cell of the group of synaptic memory cells with respect to the synaptic memory cell, each of the synaptic memory cell and said each preceding synaptic memory cell  
30 having associated therewith a corresponding scaling factor based on an order thereof in the group of synaptic memory cells;

multiplying the obtained written state values of the synaptic memory cell and said each preceding synaptic memory cell with the corresponding scaling factors, respectively, to obtain a plurality of products thereof; and

summing the plurality of products to obtain a summed value,

5 wherein the obtained second synaptic weight value of the group of synaptic memory cells obtained for the synaptic memory cell is obtained based on the summed value.

10. A neural processing core configured to perform neural network computations for a neural network, the neural processing core comprising:

10 a synaptic memory array configured to perform in-memory matrix computations;

a plurality of input axons coupled to the synaptic memory array and configured to receive a plurality of axon inputs, respectively, to the synaptic memory array; and

a plurality of output neurons coupled to the synaptic memory array and configured to output a plurality of neuron outputs, respectively, from the synaptic memory array, wherein

15 a first synaptic memory cell of a group of synaptic memory cells of the synaptic memory array has a written state value based on an intended synaptic weight value for the group of synaptic memory cells of the synaptic memory array written to the first synaptic memory cell,

a second synaptic memory cell of the group of synaptic memory cells has a written state value based on a first intended residual synaptic weight value written to the second synaptic memory cell, the second synaptic memory cell being immediately succeeding the first synaptic memory cell in the group of synaptic memory cells, and the first intended residual synaptic weight value being determined based on a first residual synaptic weight value associated with the group of synaptic memory cells,

20 the first residual synaptic weight value is obtained based on the intended synaptic weight value and an obtained first synaptic weight value of the group of synaptic memory cells, the obtained first synaptic weight value being obtained based on the written state value of the first synaptic memory cell from the intended synaptic weight value written to the first synaptic memory cell,

25 the group of synaptic memory cells collectively constitutes a combined synaptic memory cell representing a synaptic weight of a corresponding synapse of the neural network, and

the group of synaptic memory cells corresponds to a plurality of ordered bits that are ordered from highest order bit(s) to lowest order bit(s), the first synaptic memory cell corresponding to the highest order bit(s).

5 11. The neural processing core according to claim 10, wherein the first residual synaptic weight value is obtained based on a difference between the intended synaptic weight value for the group of synaptic memory cells and the obtained first synaptic weight value of the group of synaptic memory cells.

10 12. The neural processing core according to claim 10, wherein  
the intended synaptic weight value and/or the first intended residual synaptic weight value is a signed value, and  
the intended synaptic weight value is written to the first synaptic memory cell with respect to a first intended reference value and/or the first intended residual synaptic weight  
15 value is written to the second synaptic memory cell with respect to a second intended reference value.

13. The neural processing core according to claim 12, further comprising:  
a first reference memory cell communicatively coupled to the first synaptic memory  
20 cell, the first reference memory cell having a first reference written state value based on the first intended reference value written thereto; and/or  
a second reference memory cell communicatively coupled to the second synaptic memory cell, the second reference memory cell having a second reference written state value based on the second intended reference value written thereto.

25 14. The neural processing core according to claim 13, wherein  
the first intended reference value is offset from a middle value of an intended state range of the first synaptic memory cell to compensate for write noise associated with the first synaptic memory cell, and/or  
30 the second intended reference value is offset from a middle value of an intended state range of the second synaptic memory cell to compensate for write noise associated with the second synaptic memory cell.

15. The neural processing core according to claim 10, wherein the group of synaptic memory cells comprises three or more synaptic memory cells, and

for each synaptic memory cell from the second synaptic memory cell to a penultimate synaptic memory cell of the group of synaptic memory cells,

5                   an immediately succeeding synaptic memory cell of the group of synaptic memory cells with respect to the synaptic memory cell has a written state value based on a second intended residual synaptic weight value written to the immediately succeeding synaptic memory cell, the second intended residual synaptic weight value being determined based on a second residual synaptic weight value associated with the group of synaptic memory cells, and  
10                   the second residual synaptic weight value is obtained based on the intended synaptic weight value and an obtained second synaptic weight value of the group of synaptic memory cells, the obtained second synaptic weight value being obtained based on the written state values of the synaptic memory cell and each preceding synaptic memory cell of the group of synaptic memory cells with respect to the synaptic memory cell.

15

16. A method of performing neural network inference on a neural processing core configured to perform neural network computations for a neural network,

the neural processing core comprising:

20                   a synaptic memory array configured to perform in-memory matrix computations;

                  a plurality of input axons coupled to the synaptic memory array and configured to receive a plurality of axon inputs, respectively, to the synaptic memory array; and

                  a plurality of output neurons coupled to the synaptic memory array and configured to output a plurality of neuron outputs, respectively, from the synaptic memory  
25                   array, wherein

                  a first synaptic memory cell of a group of synaptic memory cells of the synaptic memory array has a written state value based on an intended synaptic weight value for the group of synaptic memory cells of the synaptic memory array written to the first synaptic memory cell,

30                   a second synaptic memory cell of the group of synaptic memory cells has a written state value based on a first intended residual synaptic weight value written to the second synaptic memory cell, the second synaptic memory cell being immediately succeeding the first synaptic memory cell in the group of synaptic memory cells, and the first intended residual

synaptic weight value being determined based on a first residual synaptic weight value associated with the group of synaptic memory cells,

the first residual synaptic weight value is obtained based on the intended synaptic weight value and an obtained first synaptic weight value of the group of synaptic memory cells, the obtained first synaptic weight value being obtained based on the written state value of the first synaptic memory cell from the intended synaptic weight value written to the first synaptic memory cell,

the group of synaptic memory cells collectively constitutes a combined synaptic memory cell representing a synaptic weight of a corresponding synapse of the neural network, and

the group of synaptic memory cells corresponds to a plurality of ordered bits that are ordered from highest order bit(s) to lowest order bit(s), the first synaptic memory cell corresponding to the highest order bit(s); and

the method comprising:

for each memory cell column of the synaptic memory array from a first memory cell column of the synaptic memory array including the first synaptic memory cell to a last memory cell column of the synaptic memory including a last synaptic memory cell of the group of synaptic memory cells, the first memory cell column to the last memory cell column forming a group of memory cell columns:

sensing analog current of the memory cell column to obtain a summed analog current of the memory cell column; and

digitizing the summed analog current of the memory cell column to obtain a digitized summed current value of the memory cell column; and

summing the digitized summed current values obtained for the group of memory cell columns to obtain a total digitized summed current value of the group of memory cell columns, the group of memory cell columns corresponding to a neuron of the synaptic memory array.

17. The method according to claim 16, wherein

said each memory cell column of the group of memory cell columns has associated therewith a corresponding scaling factor based on an order of the memory cell column in the group of memory cell columns; and

said summing the digitized summed current values obtained for the group of memory cell columns comprises:

5 multiplying the digitized summed current values obtained for the group of memory cell columns with the corresponding scaling factors, respectively, to obtain a plurality of products thereof; and

summing the plurality of products obtained for the group of memory cell columns to obtain the total digitized summed current value of the group of memory cell columns.

10 18. The method according to claim 17, further comprising:

determining a difference between the total digitized summed current value of the group of memory cell columns and a reference weighted sum value associated with the group of memory cell columns to obtain a signed total digitized summed current value of the group of memory cell columns.

15

19. The method according to claim 17, further comprising:

scaling the total digitized summed current value of the group of memory cell columns based on a first scaling factor configured to scale the total digitized summed current value to a weighted sum value of the group of memory cell columns; and

20 determining a difference between the weighted sum value of the group of memory cell columns and a reference weighted sum value associated with the group of memory cell columns to obtain a signed weighted sum value of the group of memory cell columns.

20. The method according to claim 18, further comprising determining the reference weighted sum value associated with the group of memory cell columns, comprising:

25 obtaining, for each synaptic memory cell of the group of synaptic memory cells, a reference written state value corresponding to the synaptic memory cell, each synaptic memory cell having associated therewith a corresponding scaling factor based on an order thereof in the group of synaptic memory cells;

30 multiplying the obtained reference written state values obtained for the group of synaptic memory cells with the corresponding scaling factors, respectively, to obtain a plurality of products thereof; and

summing the plurality of products obtained for the group of synaptic memory cells to obtain a summed value,

wherein the reference weight sum value is determined based on the summed value.

- 5 21. The method according to claim 18, further comprising determining the reference weighted sum value associated with the group of memory cell columns, comprising:

for each reference memory cell column of the synaptic memory array from a first reference memory cell column of the synaptic memory array including a first reference memory cell and corresponding to the first memory cell column to a last reference memory cell column of the synaptic memory including a last reference memory cell and corresponding to the last  
10 memory cell column, the first reference memory cell column to the last reference memory cell column forming a group of reference memory cell columns:

sensing analog current of the reference memory cell column to obtain a reference summed analog current of the reference memory cell column; and

15 digitizing the reference summed analog current of the reference memory cell column to obtain a digitized reference summed current value of the reference memory cell column; and

summing the digitized reference summed current values obtained for the group of reference memory cell columns to obtain a total digitized reference summed current value of  
20 the group of reference memory cell columns, wherein

said each reference memory cell column of the group of reference memory cell columns has associated therewith the corresponding scaling factor of the corresponding memory cell column thereof,

said summing the digitized reference summed current values obtained for the group of  
25 reference memory cell columns comprises:

multiplying the digitized reference summed current values obtained for the group of reference memory cell columns with the corresponding scaling factors, respectively, to obtain a plurality of products thereof; and

summing the plurality of products obtained for the group of reference memory  
30 cell columns to obtain the total digitized reference summed current value of the group of reference memory cell columns, and

the neural processing core comprises:

the first reference memory cell at the first reference memory cell column communicatively coupled to the first synaptic memory cell, the first reference memory cell having a first reference written state value based on a first intended reference value written thereto; and/or

- 5                   a second reference memory cell at a second reference memory cell column of the group of reference memory cell columns communicatively coupled to the second synaptic memory cell, the second reference memory cell having a second reference written state value based on a second intended reference value written thereto.