Structure Optimizations of Neuromorphic Computing Architectures for Deep Neural Network

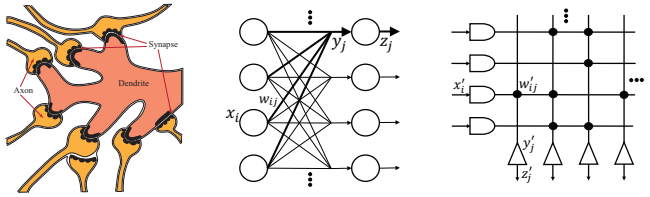
Heechun Park* and Taewhan Kim†
School of Electrical and Computer Engineering, Seoul National University, Seoul, Korea
{*phc, †tkim}@snucad.snu.ac.kr

***Abstract*—This work addresses a new structure optimization of neuromorphic computing architectures. This enables to speed up the DNN (deep neural network) computation twice as fast as, theoretically, that of the existing architectures. Precisely, we propose a new structural technique of mixing both of the dendritic and axonal based neuromorphic cores in a way to totally eliminate the inherent non-zero waiting time between cores in the DNN implementation. In addition, in conjunction with the new architecture we propose a technique of maximally utilizing computation units so that the resource overhead of total computation units can be minimized. We have provided a set of experimental data to demonstrate the effectiveness (i.e., speed and area) of our proposed architectural optimizations: ∼2x speedup with no accuracy penalty on the neuromorphic computation or improved accuracy with no additional computation time.**

I. Introduction

In the new era of neuromorphic computing, biologically inspired neural networks are realized and accelerated by various hardware platforms such as GPU, FPGA, ASIC chip, and memristor crossbar (e.g., [1]–[4]) to overcome the memory-computation gap in the traditional von Neumann architecture. TrueNorth chip, released by IBM [3], is composed of 64×64 neuromorphic cores, containing total of 1 million neurons and 256 million synapses, and each core can represent a neural network with 256 axons, 256 neurons, and a network of 256×256 synapses. TrueNorth is shown to achieve two to three orders of magnitude speedup and five orders of magnitude lower energy consumption over the traditional processors. On the other side, architectures with memristor [5] based synapse network have attracted many researchers. The feature of a memristor is similar to that of a synapse that has a tunable weight value [6], and many cognitive applications were tested with memristor crossbar of $n \times n$ memristors [7], [8].

However, most of the prior research directions focused on designing and optimizing a single neuromorphic core corresponding to a synapse network, and have no attention to the cross optimization of multiple synapse networks in DNN (deep neural network) implementation. To our best knowledges, there is no cross optimization techniques applied to any of the previous works [3] [4] [9], and they simply serially connect individual cores. Since DNN computing will be highly demanded in the future as the application complexity with high accuracy or detailed learning increases [10], structure optimization on the neuromorphic inter-core, as well as on the neuromorphic intra-core, is necessary. (In this work, *intra-core* refers to the internal structure of a neuromorphic core that performs computations using various neuron models.



(a) Neural network (b) A conceptual view (c) An implementation

Fig. 1. An example of neural network, a conceptual view and a structure of neuromorphic implementation.

Meanwhile, *inter-core* refers to the structure that integrates two neuromorphic cores. It corresponds to implementing the connections between two consecutive synapse networks in neural networks.)

This work proposes the implementation-friendly structure optimization that enables to accelerate the performance of the neuromorphic chip for DNN implementation, which is theoretically twice as fast as the conventional one, with similar use of hardware resources.

II. Preliminary: Neuromorphic Intra-core Architectures

Fig. 1(b) shows a conceptual view of a part of neural network (Fig. 1(a)), in which the output value z_j at the j^{th} output neuron is computed in two steps:

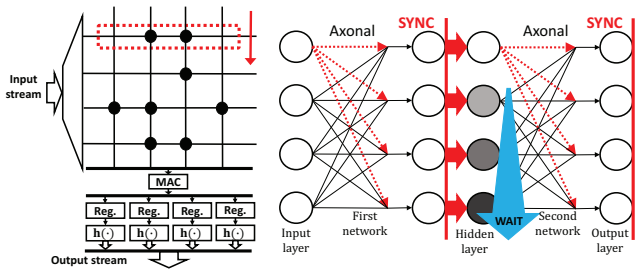
$$y_j = \sum_{i=1}^{n} w_{ij} x_i + b_j, \quad z_j = h(y_j) \tag{1}$$

where x_i is the input value from i^{th} input neuron, w_{ij} is the weight value between x_i and y_j, $h(\cdot)$ is a nonlinear activation function, and b_j is the bias value. To implement this particular network with neuromorphic core, the concept of *synapse crossbar* is generally used, as depicted in Fig. 1(c), and the weight values in the crossbar are stored in external memory [3] [9] or memristors [4].

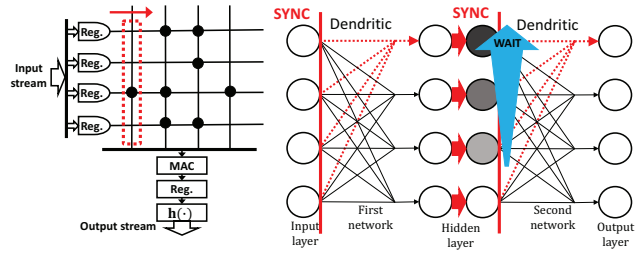
The conventional computation flows in neuromorphic core can be classified by the difference of its computation order into two models: *axonal-based* and *dendritic-based* models[1]. The left figure in Fig. 2(a) illustrates the axonal-based computation [9] which concurrently fetches the weight values from a single input (axon) to all output neurons, providing

[1]Although previous works [9] [3] are based on spiking neural network model [11], we used MAC (multiply-accumulate) unit, instead of specific module for spike inputs, as computation unit to describe both models for readability. We are only interested in the computation order of both models.

978-3-9819263-0-9/DATE18/©2018 EDAA



(a) Axonal-based model (e.g., [9])



(b) Dendritic-based model (e.g., [3])

Fig. 2. Conventional neuromorphic intra-core models and inter-core computation flows.

an iterative accumulation at every output neuron, resulting in *parallel* generation of all output values. The axonal-based model does not need all inputs ready at once, since the computation can start as long as *one* of the inputs is ready. However, to accumulate synapse weights every time when an input triggers, extra storage elements are necessary for all output neurons to retain their intermediate values until the computation for all input values is completed. Otherwise, all intermediate values of the output neurons shall be read from and restored into memory whenever an input triggers, requiring so many memory access operations which may cause a substantial increase of latency and power consumption.

On the other hand, the left figure in Fig. 2(b) illustrates the dendritic-based computation [3], which concurrently fetches the weight values from all inputs to a single output (dendrite), providing the accumulation of all weights for one output neuron at a time, resulting in *sequential* generation of output values. Therefore, unlike the axonal-based model, only one storage element for the output neuron is required and reused for the whole computation flow, to retain intermediate value and perform accumulation of the focused output neuron. However, the dendritic-based model can start computation only when *all* input values are injected, which means that the start time will be delayed until all input values are received, and additional storages to hold input values during whole computations are also needed.

III. Structure Optimization of Neuromorphic Inter-core Architecture

A. Zero Wait Inter-core Architecture

Conventionally, DNN is simply implemented by connecting multiple identical neuromorphic cores serially. The right figure in Fig. 2(a) shows the axonal-based inter-core connection

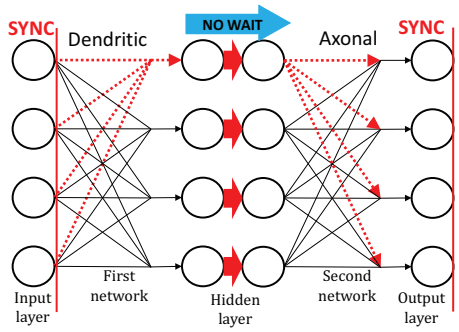


Fig. 3. Proposed denaxo-driven inter-core computation flow.

where two axonal-based cores are synchronized at the output neurons of each core due to the simultaneous output generation. We can see that even though the values of all output neurons in the first core are produced in parallel, a considerable delay waste occurs at the inputs of the second core because although all inputs are ready, MAC unit for an output neuron can only accumulate at most one input at a time, and other inputs have to stay idle. On the other hand, the right figure in Fig. 2(b) shows the dendritic-based inter-core connection where two dendritic-based cores are synchronized at the input of each core for waiting all inputs to be ready. Similarly, a large delay occurs at the inputs of the second core because the early arriving inputs for the second core have to wait for other inputs to arrive and be ready.

Based on the computation flow analysis discussed above, we devise a hybrid structure that can take advantage of the parallel computation at each of the axonal-based and dendritic-based intra-cores, as shown in Fig. 3. We call it *denaxo-driven* neuromorphic inter-core structure. The first (left) network in the denaxo-driven inter-core is implemented with a dendritic-based intra-core while the second (right) network is with an axonal-based intra-core. As a result, as shown in the computation flow denoted by red dotted arrows in Fig. 3, the dendritic-based computation in the first core and the axonal-based computation in the second core can be executed in parallel, causing no stall between the two intra-cores. In detail, both cores in a denaxo-driven structure can perform computation in parallel in a way that the i^{th} output value from the first (dendritic-based) core is received by the second core as soon as it is generated, and the accumulation for this value runs in the second (axonal-based) core in tandem with the accumulation for the $(i+1)^{th}$ output value of the first core. Therefore, unlike the conventional inter-core structures, computation process of the first and the second cores can be performed simultaneously. This parallel computation concept of denaxo-driven approach can be applied to any kind of neuromorphic computing structures with limited computation resources, and when resources are well allocated for fully utilized parallelism, denaxo-driven inter-core structure can achieve up to 2x speedup compared to the traditional approaches of connecting identical cores serially.

Consider a $m \times n \times p$ DNN consisting of $m \times n$ network

and $n \times p$ network aligned serially. For axonal-based implementation, m inputs for the first core and n inputs for the second core are treated sequentially, and both cores cannot run in parallel. Then total computation time T_{axo} becomes:

$$T_{axo} = m \cdot t_{axo\_1} + n \cdot t_{axo\_2}, \tag{2}$$

where t_{axo_1} and t_{axo_2} are the time to process computations related to one axonal input of the first ($m \times n$) and second ($n \times p$) networks, respectively. (e.g., the dotted lines in Fig. 2(a))

For dendritic-based implementation, n outputs of the first core and p outputs of the second core are generated sequentially, and both cores also cannot run concurrently. Then total computation time T_{den} is:

$$T_{den} = n \cdot t_{den\_1} + p \cdot t_{den\_2}, \tag{3}$$

where t_{den_1} and t_{den_2} are the time to generate one output value with all axonal inputs of the first and second networks, respectively. (e.g., the dotted lines in Fig. 2(b))

For denaxo-driven implementation, the total computation time T_{denaxo} is:

Lemma 1. *Let T_{denaxo} be the computation time of denaxo-driven inter-core implementation for $m \times n \times p$ neural networks. Then, T_{denaxo} can be expressed as:*

$$\begin{aligned} T_{denaxo} &= t_{den\_1} + (n-1) \cdot max\{t_{den\_1}, t_{axo\_2}\} + t_{axo\_2} \\ &= \begin{cases} n \cdot t_{den\_1} + t_{axo\_2} & \text{if } t_{den\_1} \geq t_{axo\_2}, \\ t_{den\_1} + n \cdot t_{axo\_2} & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

Lemma 1 is derived from the observation in Fig. 3 that both of the dendritic-based (first) and the axonal-based (second) intra-cores are fully parallelized, and thus the longer computation time of the intra-cores becomes the computation time of the denaxo-driven inter-core structure. For fair comparison, assume that all structures use one identical computation unit (i.e., MAC) for each of their intra-cores. Then, the computation time improvement by the denaxo-driven inter-core structure over the conventional structures can be abstracted as:

Theorem 1. *The computing speed improvement ratio ρ_{den}, of T_{denaxo} to T_{den} and ratio ρ_{axo}, of T_{denaxo} to T_{axo} for $m \times n \times p$ DNN are*:

$$\rho_{den} = \rho_{axo} = \begin{cases} \frac{mn+np}{mn+p} & \text{if } t_{den} \geq t_{axo}, \\ \frac{mn+np}{m+np} & \text{otherwise.} \end{cases} \tag{5}$$

Let us consider the improvement ratio when both synapse networks in the inter-core structure have same size, which means $m \times n = n \times p$. By Theorem 1, ρ becomes:

$$\rho_{den} = \rho_{axo} = \frac{2n}{n+1}. \tag{6}$$

Thus, for a DNN of two consecutive synapse networks with identical size, the speedup ratio of denaxo-driven inter-core structure over others theoretically approaches 2 as the network size increases.

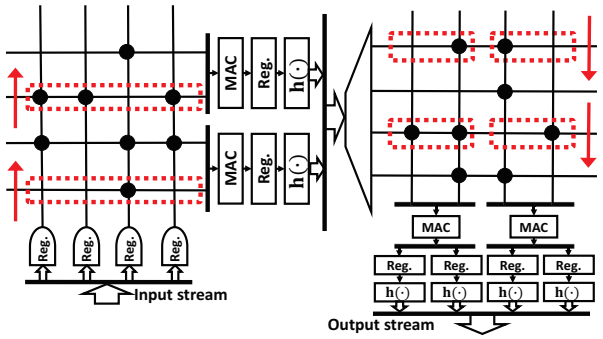


Fig. 4. Denaxo-driven inter-core implementation example of 4×4×4 DNN.

B. Resource Configuration of Inter-core Architecture

The internal configuration of the proposed inter-core structure should enable the first and second networks in the structure to be executed in parallel. Fig. 4 shows an example of denaxo-driven inter-core structure configuration with 100% computation resource utilization for a 4×4×4 DNN. The left plane implements a dendritic-based intra-core structure and produces the internal output values by using two computation units (MACs) while the right plane implements an axonal-based intra-core structure and produces the external output values by using another two MACs. We can confirm that if all MACs have identical computation time, the internal output values from the left (dendritic-based) plane can be handled by the MACs at the right (axonal-based) plane just before next output values are generated from the left plane.

Observation 1. *Let N_1 and N_2 be the numbers of MACs allocated to the first and second networks of denaxo-driven inter-core structure for the implementation of $m \times n \times p$ neural network, respectively, and let t_{mac} denote the computation delay of a MAC. Then,*

$$t_{den} = m \cdot t_{mac}, \quad t_{axo} = \frac{p \cdot N_1}{N_2} \cdot t_{mac}. \tag{7}$$

Observation 1 follows since each of N_1 MACs of the first (dendritic-based) network can multiply and accumulate the m input and weight values in t_{den} time independently, and the second (axonal-based) network receives N_1 intermediate values generated from the first network and performs multiply-accumulate with p weights for each value using N_2 MACs in parallel in t_{axo} time.

Lemma 2. *T_{denaxo} defined in Lemma 1 can be refined for the cases of $N_1 \geq 1$ and $N_2 \geq 1$:*

$$T_{denaxo} = t_{mac} \cdot [m + (\frac{n}{N_1} - 1) \cdot max\{m,\ p \cdot \frac{N_1}{N_2}\} + p \cdot \frac{N_1}{N_2}] \tag{8}$$

Theorem 2. *The computing speed improvement ratios ρ_{den} and ρ_{axo} when $N_1 (\geq 1)$ and $N_2 (\geq 1)$ MACs are allocated respectively to the first and second planes of a denaxo-driven inter-core structure for $m \times n \times p$ DNN:*

$$\rho_{den} = \rho_{axo} = \begin{cases} \frac{mn/N_1+np/N_2}{mn/N_1+N_1p/N_2} & \text{if } t_{den} \geq t_{axo}, \\ \frac{mn/N_1+np/N_2}{m+np/N_2} & \text{otherwise.} \end{cases} \tag{9}$$

For implementing DNNs with $m \times n = n \times p$ and $n \gg N_1, N_2$ (which means $N_1 \simeq N_2$), ρ becomes:

$$\rho_{den} = \rho_{axo} = \frac{2n}{n + N_1}. \quad (10)$$

Thus, the maximum theoretical speedup ratio approaches 2 as the network size increases. The variation of the speedup ratios for several configurations of N_1 and N_2 are shown in the experimental sections. Theoretically, the highest ratio with full utilization comes from the configuration that satisfies (1) $1 \leq N_1 \leq n$, (2) $1 \leq N_2 \leq p$ and (3) $\frac{N_1}{N_2} = \frac{m}{p}$. (Proof is omitted for space limitation.)

C. Using Denaxo-driven Inter-core in Large DNNs

We suggest two options to use the proposed denaxo-driven inter-core as a basic building block in implementing DNNs with many synaptic connections.

Option 1: The straightforward option is to implement DNN of k synapse networks with $k/2$ denaxo-driven inter-core structures, in which one inter-core corresponds to the implementation of two serially connected synapse networks. The computation speedup exactly follows that in Theorem 2. Note that this option can implement any form of neural networks only if $k \geq 2$, at the expense of the extra work to allocate and deploy MACs in the constituent inter-core structures.

Option 2: This option does not need redesigning denaxo-driven inter-cores depending on the size and structure of neural networks to implement. Instead, it requires a slight transformation on the structure of the input DNN, i.e., insert a hidden layer of smaller size between input-output neurons of every synapse networks and re-train weight values with transformed DNN structure. This transformation provide two distinct benefits:

1. The first one is convenience. The denaxo-driven inter-core structure can be in a compact form of size $m \times n$ that can implement transformed $m \times \lfloor \frac{mn}{m+n} \rfloor \times n$ neural network. Fig. 5 shows our proposed denaxo-driven inter-core architecture that can be used as a basic building block of the transformed DNN. In other words, it can be plugged into any transformed neural network to use it as a basic building block.
2. When we assume that an equal amount of arithmetic resources (i.e., $N = N_1 + N_2$) is used by both of the conventional intra-cores (i.e., axonal-driven and dendritic-driven) and proposed basic block (i.e., denaxo-driven) for a $n \times n$ neural network, the conventional cores will take $\frac{n \cdot n}{N} \cdot t_{mac}$ time, while our basic block takes $(\frac{n \cdot n}{N} + n) \cdot t_{mac}$ time (according to *Eq.8*). Thus, the speed ratio ρ is $\frac{n}{n+N} \simeq 1$ as $n \gg N$, and the transformed neural network with more layers (almost doubled) enables a great improvement on the learning accuracy.

IV. Experimental Results

Experiments are conducted on a Linux machine of 3.50GHz Intel i7 Processor and 16GB memory. We modeled various structures with Verilog HDL description, synthesized them with Synopsys Design Compiler using NANGATE 45nm open cell library [12], and compared the cell area reported from the tool. All structures are simulated on Cadence Incisive Enterprise Simulator to obtain DNN computation time and speed improvement ratio (ρ). The conventional inter-core structures are produced by aligning two identical intra-core structures, either dendritic-based [3] or axonal-based [9], in serial[2]. The proposed denaxo-driven inter-core structure is implemented by aligning dendritic-based core on the first and the axonal-based core on the second, followed by various MAC allocations. All values are encoded in a 16-bit fixed-point number according to [13]. A 4-cycle 16-bit multiplier and a 1-cycle 32-bit accumulator compose one MAC unit, and ReLU (rectified linear unit) module is used for activation.

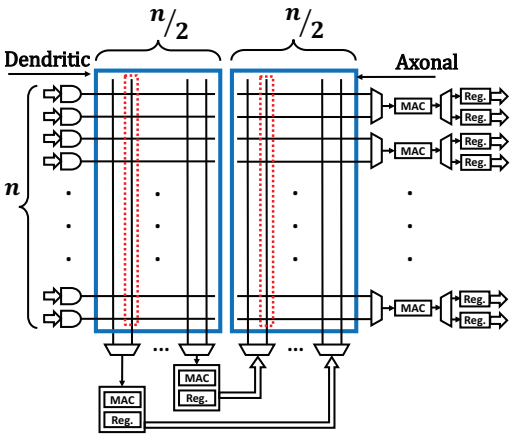


Fig. 5. The proposed architecture of a denaxo-driven inter-core (in option 2) whose crossbar size is changed from $n \times n$ to $n \times \frac{n}{2} \times n$.

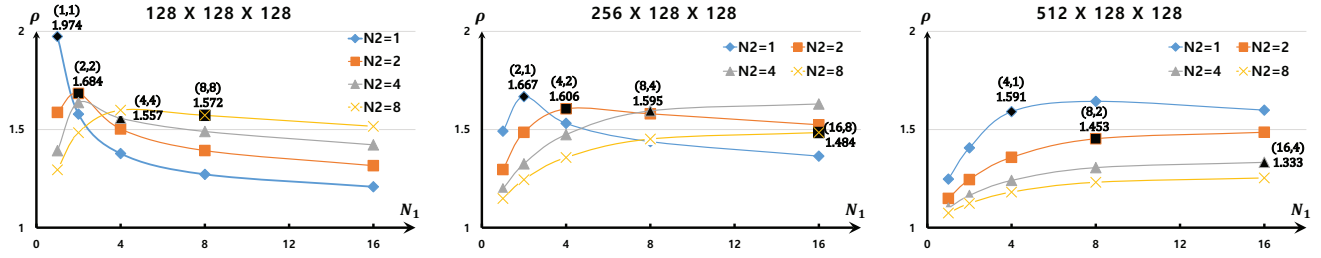
A. Evaluation of Computation Speed

Table I shows a comparison of our denaxo-driven inter-core structure with that of the conventional inter-core structures with various network sizes and MAC unit allocations. Throughout this experiment, identical weight values are used for all types of inter-core implementations, which means all DNN implementations have the same accuracy. Since the computation times of the conventional structures are similar, the speedup ratio (ρ) is calculated by comparing the computation time of denaxo-driven structure with the average of both conventional structures. Comparison of cell area will be described in the next subsection. Our denaxo-driven inter-core architecture shows better performance than the conventional ones by 38% $\sim$ 99%. Denaxo-driven structure enhances the speedup by performing calculations of both network planes in parallel, revealing from the last column of Table I that ρ becomes larger when the network size difference between two network planes is smaller.

[2]Although the intra-core structures in [3] and in [9] both were used in computing a spiking neural network in a prior work, for a fair comparison we trained and tested the conventional intra-cores that use 16-bit fixed-point precision.

TABLE I
COMPARISON OF THE PERFORMANCE OF PROPOSED DENAXO-DRIVEN INTER-CORE STRUCTURE WITH THAT OF THE CONVENTIONAL DENDRITIC-BASED (E.G., [3]) INTER-CORE AND AXONAL-BASED (E.G., [9]) INTER-CORE.

Neural networks	Size ratio	#MAC	Dendritic-based inter-core [3]		Axonal-based inter-core [9]		**Denaxo-driven inter-core**		
$(m \times n \times p)$	$(m \times n) : (n \times p)$	(N_1/N_2)	Area(μm^2)	Runtime(#cycles)	Area(μm^2)	Runtime(#cycles)	Area(μm^2)	Runtime(#cycles)	Speedup(ρ)
64× 64× 64	1 : 1	1/1	25027	**82698**	58638	**83202**	41834	**42567**	**1.949**
128× 128× 128			45864	**329226**	113623	**330242**	79741	**167047**	**1.974**
256× 256× 256			88048	**1313802**	224370	**1315842**	156157	**661767**	**1.987**
128× 128× 64	2 : 1	2/1	56400	**181133**	86884	**182018**	62827	**109062**	**1.665**
256× 128× 128			85976	**361933**	114365	**363266**	119820	**217478**	**1.667**
256× 256× 128			107052	**722701**	168520	**724482**	119876	**431110**	**1.678**
128× 128× 32	4 : 1	4/1	76951	**107091**	75859	**107906**	69651	**67507**	**1.592**
256× 128× 64			123207	**214003**	89602	**215170**	129640	**134835**	**1.591**
512× 128× 128			215466	**427827**	117083	**429698**	249359	**269491**	**1.591**
256× 128× 32	8 : 1	8/1	197985	**140047**	82178	**141122**	190701	**101747**	**1.382**
512× 128× 64			357385	**279983**	95920	**281666**	363754	**203379**	**1.381**
512× 256× 64			378510	**558591**	149248	**560770**	363809	**400067**	**1.399**



(a) 128×128×128 (b) 256×128×128 (c) 512×128×128

Fig. 6. Comparison of computation speed ratio (ρ) for various MAC allocations (N_1, N_2) to two network planes in an inter-core.

Fig. 6 shows the curves of speedup ratio ρ as MACs are distributed to the first and second planes of denaxo-driven inter-core structure. The values of m/p of the networks corresponding to Figs. 6(a), (b) and (c) are 1, 2 and 4, respectively. The curves show that the highest speedup occurs when MAC distribution ratio N_1/N_2 gets close to m/p. This relation of N_1/N_2 and m/p is obvious from the fact that the number of MACs required should follow the ratio of the size of the two network planes to achieve a maximal parallelism on the denaxo-driven structure. However, as indicated by the shape of the curves, $\frac{N_1}{N_2} = \frac{m}{p}$ does not always mean a maximal parallelism ($\rho \approx 2$) as the number of available MACs increases. This is because as the number of MACs increases, the total computation time will decrease accordingly. Thus, the relative portion of the runtime for arithmetic computation decreases since the time for memory access such as loading weights and storing intermediate values does not change. This means that, our proposed denaxo-driven structure is more effective for the computing platforms where the necessity of computation resources (i.e., MACs) is more stringent.

B. Evaluation of Cell Area

Fig. 7 shows the impact of MAC allocation on cell area. Changes of cell area for two conventional intra-core (*one* neural network) implementations as the amount of MAC resource (N) increases are shown. When N is 1 or 2, the area of axonal-based core is twice larger than that of dendritic-based one for the same network size. However, as the value of N increases, the rate of area increment in dendritic-based core is larger than that in axonal-based one. This is because if more MACs are allocated for dendritic-based core, the storage units for weight values and intermediate values are also needed. Therefore, to design a denaxo-driven inter-core structure, a careful distribution of arithmetic modules to the first (dendritic-based) plane with large input width is required since it needs more storages for loading weights from memory.

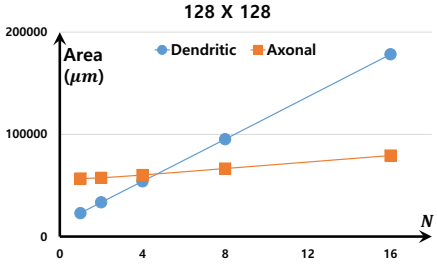


Fig. 7. Comparison of cell area used by the intra-cores under various MAC resource usages (N).

TABLE II
COMPARISON OF CELL AREA USED BY DENAXO-DRIVEN INTER-CORE STRUCTURE WITH THAT OF THE CONVENTIONAL INTER-CORES.

Neural networks	Dendritic-based [3]		Axonal-based [9]		Denaxo-driven	
$(m \times n \times p)$	Area(μm^2)		Area(μm^2)		Area(μm^2)	
256×**64**×128	56515	-	86148	-	100779	-
256×**128**×128	66943	×1.18	113636	×1.32	100818	**×1.00**
256×**256**×128	88024	×1.56	169009	×1.96	100851	**×1.00**
256×**64**×64	56493	-	58674	-	73301	-
256×**128**×64	66920	×1.18	86156	×1.47	73345	**×1.00**
256×**256**×64	88003	×1.56	141496	×2.41	73381	**×1.00**
256×**32**×32	51255	-	31197	-	59546	-
256×**64**×32	56471	×1.1	44935	×1.44	59576	**×1.00**
256×**128**×32	66899	×1.31	72432	×2.32	59616	**×1.00**

Table II summarizes the relation of network size to cell area of all inter-core structures. We assumed $N_1 = N_2 = 1$ and the same weight values are used. The area of dendritic-based structure is proportional to the input width of each network (m and n) to store all input values for accumulation. In contrast, the area of axonal-based structure is proportional to the output

TABLE III
COMPARISON OF THE PERFORMANCE OF INTER-CORE STRUCTURES WITH MNIST DATABASE [14].

N_1/N_2	784×256×10 (Accuracy = 83.90%)			
	Runtime(#cycle)			
	Dendritic-based	Axonal-based	Denaxo-driven	Speedup(ρ)
1/1	**2035548**	**2050218**	2010147	1.016
2/1	**1232095**	**1246890**	1206843	1.027
4/1	830373	845226	805419	1.040
8/1	629521	644394	605163	1.053
$N_1/N_2/N_3$	784×256×256×10 (Accuracy = 91.24%)			
1/1/1	2692192	2707627	**2039902**	1.324
2/1/1	1888739	1904299	**1239010**	1.531
4/1/1	1487017	1502635	931933	**1.604**
8/1/1	1286165	1301803	819357	1.579

width of each network (n and p) to store intermediate values and activation units. Since axonal-based structure requires more storage spaces, total cell area of axonal-based inter-core structure is larger than that of dendritic-based one in most cases. Opposite cases occur when $m \gg n, p$. Combining both features, proposed denaxo-driven structure will have the largest cell area when the hidden layer size is smaller ($n \ll m, p$). In general, the cell area of denaxo-driven structure is close to the average of the areas of the two conventional ones. The entries beginning with $\times$ in Table II indicate the amount of area increase in ratio when the hidden layer size is doubled/quadrupled/octupled. Since the total cell area of denaxo-driven structure is determined by only the values of m and p, we can flexibly reshape the proposed architecture by varying the value of n while maintaining total cell area.

C. Experiments with MNIST Dataset

We compared the inter-core structures with MNIST handwritten digit classification database [14]. This database includes 60000 handwritten digits in the form of 28×28 gray level pixels, each of which represents a digit between 0 to 9. To compare the computing performance of various inter-core structures, we used the DNN of two (784×256×10) and three (784×256×256×10) consecutive synapse networks. For the latter, denaxo-driven inter-core structure is applied to the first two networks (784×256×256) and the last network (256×10) is implemented with dendritic-based core. Weight values are obtained from TensorFlow software with random weight initialization and gradient-descent optimizer[3].

Results are summarized in Table III. For 784×256×10 DNN, denaxo-driven structure achieves only a little speed improvement (∼5.3%). Such type of DNN with unbalanced network size is not suited for our structure, since the best distribution ratio of MACs is $\frac{784}{10}$ = 78.4 and clearly it is not efficient to allocate a large number of MACs to meet this ratio (i.e., $N_1 = 78, N_2 = 1$) to achieve the theoretical speedup (53% from Eq. 9). Instead, for 784×256×256×10 DNN, in which a synapse network with comparable size of 256×256 is inserted for more accuracy, best distribution ratio for the first two network becomes $\frac{784}{256} \approx 3$ and proposed denaxo-driven implementation can achieve speedup over 60% with reasonable number of MACs.

One more interesting point from this table is that the runtime of 784×256×256×10 DNN with denaxo-driven structure is even comparable to that of much smaller DNN (784×256×10) with conventional (dendritic-based, axonal-based) structures. and it is almost equal to the cases of little usage of (i.e., 1 or 2) MACs. This implies that using our proposed denaxo-driven inter-core structure with restricted computation resources can improve accuracy by implementing 'deeper' neural network with no additional computation time.

[3]Although higher accuracy can be achieved with better weight initialization (e.g., [15]) or efficient learning algorithms (e.g., [16]), we used a simple and fast method for training since trying to improve accuracy with learning algorithm is not relevant to this work.

V. CONCLUSIONS

We proposed a new structure optimization technique to improve the computation speed in neuromorphic computing architectures for deep neural networks. Proposed denaxo-driven inter-core structure, exploiting both characteristics of dendritic-based and axonal-based neuromorphic core architectures, was able to increase the computation speed theoretically by a factor of 2 over that of the conventional structures, and practically it was able to increase the computation speed by 38%∼99% according to the network size and resource utilizations.

VI. ACKNOWLEDGEMENTS

This work was supported by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT1703-00.

REFERENCES

[1] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *ACM Multimedia*, 2014.
[2] Y. Wang *et al.*, "Deepburning: Automatic generation of fpga-based learning accelerators for the neural network family," in *ACM/EDAC/IEEE DAC*, 2016.
[3] F. Akopyan *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE TCAD*, no. 10, 2015.
[4] C. Liu *et al.*, "A spiking neuromorphic design with resistive crossbar," in *ACM/EDAC/IEEE DAC*, 2015.
[5] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on Circuit Theory*, no. 5, 1971.
[6] M. D. Ventra, Y. V. Pershin, and L. O. Chua, "Circuit elements with memory: Memristors, memcapacitors, and meminductors," *Proceedings of the IEEE*, no. 10, 2009.
[7] M. Chu *et al.*, "Neuromorphic hardware system for visual pattern recognition with memristor array and cmos neuron," *IEEE TIE*, no. 4, 2015.
[8] M. Hu *et al.*, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE TNNLS*, no. 10, 2014.
[9] J. V. Arthur *et al.*, "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," in *IJCNN*, 2012.
[10] K. Peng, S. S. Ge, and C. Wen, "An algorithm to determine neural network hidden layer size and weight coefficients," in *IEEE ISIC*, 2000.
[11] A. S. Cassidy *et al.*, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in *IJCNN*, 2013.
[12] NANGATE. (2011) Nangate 45nm Open Cell Library. http://www.nangate.com/.
[13] Y. Chen *et al.*, "Dadiannao: A machine-learning supercomputer," in *IEEE/ACM MICRO*, 2014.
[14] Y. LeCun *et al.* (1998) The MNIST database of handwritten digits.
[15] X. Glorot *et al.*, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research*, 2010.
[16] D. Kingma *et al.*, "Adam: A method for stochastic optimization," in *ICLR*, 2015.