Technology-Design Co-optimization of Resistive Cross-point Array for Accelerating Learning Algorithms on Chip

Pai-Yu Chen, Deepak Kadetotad, Zihan Xu, Abinash Mohanty, Binbin Lin, Jieping Ye, Sarma Vrudhula, Jae-sun Seo, Yu Cao, Shimeng Yu*

Arizona State University, Tempe, AZ 85281, USA

Email: shimengy@asu.edu

Abstract—Technology-design co-optimization methodologies of the resistive cross-point array are proposed for implementing the machine learning algorithms on a chip. A novel read and write scheme is designed to accelerate the training process, which realizes fully parallel operations of the weighted sum and the weight update. Furthermore, technology and design parameters of the resistive cross-point array are co-optimized to enhance the learning accuracy, latency and energy consumption, etc. In contrast to the conventional memory design, a set of reverse scaling rules is proposed on the resistive cross-point array to achieve high learning accuracy. These include 1) larger wire width to reduce the IR drop on interconnects thereby increasing the learning accuracy; 2) use of multiple cells for each weight element to alleviate the impact of the device variations, at an affordable expense of area, energy and latency. The optimized resistive cross-point array with parallel circuitry is implemented at the 65 nm node. Its performance is benchmarked for handwritten digit recognition on the MNIST database using gradient-based sparse coding. Compared to state-of-the-art software approach running on CPU, it achieves >10^5 speed-up and >10^6 energy efficiency improvement, enabling real-time image feature extraction and learning.

Keywords—machine learning; neuromorphic computing; cross-point array; resistive memory; synaptic device

I. INTRODUCTION

Recent advances in neuro-inspired machine learning algorithms have achieved tremendous success in speech/image recognition, with the support of supercomputers [1]. However, conventional CPUs/GPUs, which are based on the sequential von Neumann architecture, is inadequate for fast training with large-scale data set, especially with limited power constraints. Although the custom designed ASIC with partial parallelism (e.g., SyNAPSE chip by IBM [2]) has shown advantages over CPUs/GPUs, its computing speed still lags behind the requirement of real-time online learning (e.g., SyNAPSE chip trains weights offline [2]). To achieve further speed-up, it is attractive to explore beyond-traditional CMOS designs with emerging non-volatile memory devices such as phase change memory, and resistive memory, etc. The memory device that can tune its conductance into multilevel-states is referred to as the synaptic device [3]. Fig. 1 shows an example of Ag/a-Si based synaptic device [4]; the device conductance can continuously increase or decrease depending on the input voltage pulse polarities.

A resistive cross-point array architecture with synaptic devices has been proposed to perform the weighted sum and weight update in a neural network [5-6]. Although simple learning rules have been demonstrated at a single device level or small array level [7-10], an extension to a large array may result in significant loss of the learning accuracy due to the IR drop along the interconnects or the intrinsic device variations. In this work, we aim to address the challenges and potential solutions for a large-scale implementation of machine learning algorithms on the resistive cross-point array architecture. The contributions of this work include:

1) Developed co-optimization methodologies of learning algorithms, device characteristics and array architecture for on-chip implementation. Sparse coding algorithm were truncated into finite precisions for on-chip implementation.

2) Proposed a fully parallel read and write scheme of the weighted sum and the weight update in the resistive cross-point array for speed-up in the weight training.

3) Proposed a set of reverse scaling rules for sizing the array geometrical dimensions such as the wire width and the cell spacing to achieve high learning accuracy.

4) Analyzed the effect of realistic non-ideal synaptic device’s properties on the learning accuracy, and proposed using multiple cells to represent one weight element to alleviate the impact of the device variations.

5) Validated the proposed design methodologies through circuit-level HSPICE simulations, and the metrics of area, energy and latency were benchmarked with the traditional software approach running on CPU.

This work is partially supported by Samsung GRO program and Raytheon through NSF I/UCRC Center for Embedded Systems.
II. IMPLEMENTATION OF SPARSE CODING ALGORITHM FOR ON-CHIP LEARNING

A. Sparse Coding Algorithm

In this work, we target at implementing resistive cross-point array for accelerating the sparse coding algorithm, which is a class of machine learning algorithms that can extract the features from the training data in an unsupervised learning way [11]. Similar design methodologies in this work can also be applied to other supervised learning or multi-layer deep learning algorithms. Sparse coding has demonstrated its power in numerous domains such as audio processing, text mining and image recognition. Sparse coding can also model the inhibition between the learned feature vectors. Similar inhibition properties have been observed in biological neurons, making sparse coding a plausible bio-physiological model of the primary visual cortex in mammals [12].

The sparse coding algorithm aims to learn a dictionary matrix (D) such that each input vector (X) can be represented by a sparse linear combination of the feature vectors (Z) by the dictionary. For sparse coding, it is assumed that each input can be represented only by a small group of elements in Z, that is, the learned feature Z is a sparse vector. Mathematically, sparse coding solves the following optimization problem. Both the extracted feature Z and the dictionary D are learned by iteratively minimizing the objective error function in Eq. (1):

\[
E = \min \sum (\|DZ - X\|^2 + \lambda \|Z\|_1) \tag{1}
\]

The first term of Eq. (1) is the reconstruction error, which measures how well the dictionary reconstructs the input data. The second term of Eq. (1) ensures the sparsity of the learned feature. When both D and Z are unknown, the above optimization problem is a non-convex problem. However it reduces to a convex optimization problem when either D or Z is known or fixed. Thus, the convergence can be achieved by iteratively update Z (with fixed D) with coordinate descent method and update D (with fixed Z) with stochastic gradient descent method, as shown in Eq. (2) and Eq. (3).

Update Z:

\[
Z_j \leftarrow h_j [(D_j)^T(X - DZ) + Z_j] \tag{2}
\]

Update D:

\[
D \leftarrow D - \eta RZ^T (R = DZ - X) \tag{3}
\]

The handwriting MNIST data set [13] is used as the training data set in this work, where the raw images are densely sampled into several small image patches with 10×10 pixels as X input vector with a dimension of 100, as shown in Fig. 2. The dictionary (D) before training is randomly sampled with the image patches, and after training, we can see that the Gabor-filter-like features have been learned, as shown in Fig. 3. Then after the pooling of the learned features, the classification techniques, e.g. the support vector machine (SVM), can be used to classify the 10 digits.

B. Truncation of the Algorithm for Limited Precision

In the sparse coding algorithm, the values of D, Z and X are normalized between 0 and 1. To implement the algorithm directly on chip, the precision of the real values in the code needs to be limited as compared to the typical double precision format with 52 bits of mantissa (2^{52} levels) in the software approach. Therefore, we truncate the values (D, Z, X) in the sparse coding algorithm into finite precisions. Fig. 4 shows the objective function error as a function of the number of D/Z levels and Z vector dimension. The objective function error is an indicator of the learning accuracy (smaller error, better accuracy). A balanced design point between the hardware resources required on-chip and the learning accuracy is chosen: 4 bits for X and Z (16 levels), 6 bits for D values (64 levels), and 500 for Z vector dimension. D values=64 levels suggests that 64 levels of conductance states are needed, which is shown to be feasible in many synaptic device candidates [4, 8-10]. In addition, the Z vector dimension=500 requires a 100×500 dimension of the D matrix. Therefore, a 100×500 resistive cross-point array with 64-level synaptic devices is used to represent the D matrix in this work.
Fig. 4 Objective function error in Eq. (1) with (a) different precision levels of D and Z and (b) different Z vector dimensions. Z=16 levels (4 bits), D=64 levels (6 bits) and Z vector dimension=500 are chosen as the design point.

III. PROPOSED ARCHITECTURE OF RESISTIVE CROSS-POINT ARRAY FOR ON-CHIP LEARNING

A. Parallel Read and Write Scheme for Speed-up

Fig. 5 shows the schematic of the proposed architecture of the resistive cross-point array. To compute the weighted sum (DZ) in the read operation, a small read voltage is applied in parallel to each row for every non-zero element of Z. There are two ways to encode Z: voltage-amplitude analog scheme and the number-of-pulse digital scheme. We will compare these two encoding schemes in the next section. Then the read voltage multiplied by the conductance of the synaptic device at each cross-point, and the weighted sum results in the output current at each X node at the end of the columns. The read peripheral circuitry for each column then converts the analog current output to the digital numbers. Note that the sneak path problem of the unselected cells in the array for conventional memory application does not exist here. This is because the conventional memory requires reading out data by bit or by row, while the weighted sum operation here reads the entire array in parallel, thus all the cells in the array are participating in the computation according to the Kirchhoff Law.

Fig. 5 Proposed resistive cross-point array architecture. Read scheme is shown for performing the weighted sum (matrix-vector product DZ) in parallel. The read voltage (Z) is multiplied by the conductance (D) of the synaptic device at each cross-point, and the weighted sum results in the output current at each X node at the end of the columns. There are two ways to encode Z: voltage-amplitude analog scheme and number-of-pulse digital scheme. We will compare these two encoding schemes in the next section. Then the read voltage multiplied by the conductance of the synaptic device at each cross-point, and the weighted sum results in the output current at each X node at the end of the columns. The read peripheral circuitry for each column then converts the analog current output to the digital numbers. Note that the sneak path problem of the unselected cells in the array for conventional memory application does not exist here. This is because the conventional memory requires reading out data by bit or by row, while the weighted sum operation here reads the entire array in parallel, thus all the cells in the array are participating in the computation according to the Kirchhoff Law.

To update the weight matrix D, the synaptic device conductance in the array needs to be adapted according to the product of Z and R (R is the residual error DZ-X in Eq. (3)). A fully parallel write scheme is developed to update the entire array for speed-up. Programming pulses that represent Z and R are simultaneously applied on the rows and on the columns of the array, respectively. We overlap the Z pulses and r pulses over the write enable period to effectively realize the multiplication function (ZR). Specifically, we encode R value into spikes of short pulses in a fixed time period, and encode Z value into the duty cycle of the write period when R pulses are applied to each synaptic device. Thus, in such a synchronous design, the accumulated overlap time of these two pulses in each write cycle indicates the product of ZR. Fig. 6 shows an example of the write scheme for updating D using the product of ZR (simulated in HSPICE with peripheral circuitry in 65 nm CMOS). The details of the read and write peripheral circuitry design can be referred to [14].

B. Synaptic Device Model in Cross-Point Array

To simulate the weighted sum operation in HSPICE, synaptic device in each cross-point is modeled as a cell resistor and capacitor with the wire resistances and parasitic capacitances. The interconnect parameters are obtained from the ITRS table [15]. Fig. 7 (a) shows a sub-circuit module of a cross-point, and such module is to be duplicated for the entire array. The HSPICE simulation is then performed on the entire array. Typically the synaptic device resistance is nonlinear under different applied voltages due to tunneling or hopping nature of the electrons in the cross-point junction. Fig. 7 (b) shows an example of the nonlinear resistance with experimental data from synaptic device in [4].

Fig. 6 Write scheme for updating D in parallel using the product of ZR (simulated in SPICE with peripheral circuitry in 65 nm CMOS). The write time for updating D is 84 ns, the first 42 ns for R>0 (left), and the second 42 ns for R<0 (right). The Z value is encoded by length of time and R is encoded by number of active-high pulses (R>0) or active-low pulses (R<0). The D value changes when there is an overlapped pulse window of Z and R.

Fig. 7 (a) Sub-circuit module of a synaptic device cell (S: cell spacing; W: wire width). The cell capacitor (C_s) is in parallel with the cell resistor (R_s). The wire resistors (R_w) and capacitors (C_w) for top and bottom interconnect are considered. Sub-circuit is duplicated in HSPICE for simulating the entire array. (b) Nonlinear cell resistance versus applied voltage, see the experimental I-V from [4] (inset).
IV. DESIGN METHODOLOGIES OF RESISTIVE CROSS-POINT ARRAY FOR ON-CHIP LEARNING

In this section, the design methodologies are evaluated by HSPICE simulation. First, we sample the real values of D, Z, X over the entire learning period from the sparse coding software simulation, and map them to the applied voltages or device conductance. Thus the DZ matrix-vector product is mapped to a current value called \( I(\text{ideal}) \). Then we run the HSPICE simulation and measure the current output at end of the columns as \( I(\text{actual}) \). Here the read accuracy deviation is defined as \( |I(\text{actual})-I(\text{ideal})|/I(\text{ideal}) \). The read energy is referred to as the total energy consumption of the entire array defined as \( \sum_{i=1}^{n} E_i \).

A. Encoding the Signal: Analog Voltage vs. Digital Spike

There are two ways to encode Z: voltage-amplitude analog scheme and the number-of-pulse digital scheme. To represent the 16 levels of Z, the analog scheme uses voltage pulses from 0 V to 0.75 V with a step 50 mV, while the digital scheme uses a fixed 0.3 V pulse and the number of pulses vary from 0 to 15. Both schemes fix the width for each pulse to be 5 ns. Fig. 8 shows the read accuracy deviation and read energy for the two encoding schemes of Z. It shows that the digital scheme is more accurate because the analog scheme has a distortion problem due to the nonlinearity of cell resistance at different applied voltage (see Fig. 7 (b)). On the other hand, the analog scheme is more energy-efficient because the digital scheme generally needs more pulses. As our emphasis is on achieving high learning accuracy, the digital scheme is preferred. In addition, the analog scheme is inherently more susceptible to noise, and it adds more complexity in the peripheral circuitry to generate fine-grain voltages on-chip. Thus the digital scheme is used in the following simulations.

B. Impact of Cell Resistance on Read Accuracy

We investigate the effect of synaptic device’s on-state resistance \( R_{\text{ON}} \) that stands for the maximum D value. The off resistance is fixed to be 1000× \( R_{\text{ON}} \) which gives a sufficiently small D value of 0.001 for an approximation of minimum D=0. Fig. 9 shows the read accuracy deviation and read energy for different \( R_{\text{ON}} \). It shows that increasing \( R_{\text{ON}} \) improves the accuracy and also reduces the energy. This is because a high synaptic device resistance helps to minimize the IR drop on the interconnect wires. It is seen that \( R_{\text{ON}} \) above \( 1 \) M\( \Omega \) significantly reduces the read inaccuracy below 1%, and the read energy below 0.1 pJ. Such high \( R_{\text{ON}} \) has been demonstrated in some synaptic device candidates [4]. Therefore, \( R_{\text{ON}} = 1 \) M\( \Omega \) is used in the following simulations.

C. Sizing Wire Width for Minimizing Interconnect IR drop

We study the impact of geometrical scaling on the wire width (W) and the cell spacing (S). The definitions of W and S are referred to the schematic of Fig. 7 (a). Fig. 10 shows the read accuracy deviation as a function of W and S, respectively. It is seen in Fig. 10 (a) that for a small wire width (W=20 nm), the read accuracy becomes unacceptable due to the severe IR drop on interconnects as the copper resistivity increases dramatically at the nanoscale [15]. Thus it is necessary to scale up W, for example, W=200 nm reduces the read inaccuracy below 1%. It is seen in Fig. 10 (b) that the effect of the cell spacing (S) on the read accuracy is less prominent.

To estimate the read latency due to the wire interconnects and parasitic capacitances, transient simulations for various W and S are performed. Fig. 11 shows the current waveform measured at end of the column as a function of W and S, respectively. It is seen in Fig. 11 (a) and its inset figure that a small W=50 nm (red) leads to a remarkable loss of the DC value of the current as compared to a large W=200 nm (blue), which agrees with the results in Fig. 10 (a) on the read accuracy. In addition, a small W increases the read latency due to the increased wire resistance. Therefore, scaling up W is also necessary from the viewpoint of improving the read latency. It is seen in Fig. 11 (b) and its inset figure that smaller S reduces the read latency. However, the current overshoot in the waveform due to the synaptic device’s cell capacitance (that couples input to output, see Fig. 7 (a)) is severe when S is small. Therefore, S=1 \( \mu \)m is chosen, limiting the current overshoot <100%. In summary, the W=200 nm and S=1 \( \mu \)m are identified as the optimal design parameters considering the learning accuracy and latency. This conclusion essentially suggests that the resistive cross-point array should be integrated on the intermediate interconnect (e.g. M4 or M5) for on-chip learning. This reverse scaling rule proposed here is in contrast to the aggressive downscaling applied on conventional high-density memory.
Fig. 10 Distribution of (a) read accuracy deviation and (b) read energy for different wire width (W) and cell spacing (S). Scaling up W=200 nm is necessary to minimize the wire resistance effect on the read accuracy, while the impact of cell spacing on the read accuracy is insignificant.

Fig. 11 Transient simulation on current waveforms for (a) different wire width (W) and (b) different cell spacing (S). As W or S shrinks, the read latency becomes larger due to larger wire resistance (inset). Scaling up W=200 nm is helpful to reduce the latency. The current overshoot in the waveform is due to the synaptic device’s cell capacitance that couples input to output.

D. Redundant Cells for Minimizing Device Variations

It is known that the synaptic devices relying on the ion/defects motion typically show noticeable device variations from cell-to-cell and from cycle-to-cycle. The device variations may significantly hamper the read accuracy. Fig. 12 shows the read accuracy deviation as a function of the standard deviation (σ) of the device resistance variations. It is seen that the read accuracy degrades quickly with increasing σ. To alleviate the impact of the variations, we propose to use multiple cells as one D weight element, which statistically averages out the resistance variations. Fig. 13 (a) shows the averaging effect of resistance variations by using multiple cells. Fig. 13 (b) shows the read accuracy deviation as a function of the number of the cells. It is seen that using 3×3 cells as one element results in a great reduction of the read inaccuracy down to ~10%. With the help of the read peripheral circuitry that functions as an analog-to-digital converter, the accuracy is expected to be further improved because some analog distortions can be eliminated when digitalized. However, further increasing the number of the cells has drawbacks: first, the latency increases due to longer wire distance, as shown in Fig. 14 (a), approaching 1.4 ns for 3×3 cells; second, the energy consumption increases, approaching 1 nJ for 3×3 cells for write operation, as shown in Fig. 14 (b); lastly, the area of the cross-point array inevitably increases. The upper limit on the number of the redundant cells should be constrained by the peripheral circuitry area. Our rule of thumb is that the cross-point array area should be less than the peripheral circuitry area as the cross-point array is built on top of the CMOS circuits at back-end-of-line interconnect levels.

Fig. 12 Distribution of read accuracy deviation for different sigma of the device variations on cell resistance. Device variations significantly degrade the read accuracy of the entire array.

Fig. 13 (a) Resistance distribution for different number of cells as one D element. Using multiple cells as one D element can average out the device variations on the resistance. (b) Distribution of read accuracy versus number of cells. Using 3×3 cells as one D element significantly improves the read accuracy.

Fig. 14 (a) Transient simulation on current waveforms when using different number of cells. (b) Read and write energy consumption when using different number of cells. Using more cells adds more wire resistances/capacitances and thus increases the read latency (inset), and energy is proportional to the number of cells used.

V. BENCHMARK WITH SOFTWARE APPROACH

With the design methodologies proposed in Section IV, a set of reverse scaling rules of resistive cross-point array for on-chip learning is applied, as shown in Table 1: the wire width and cell spacing need to be scaled up, and additional redundancy with multiple cells is needed as compared with the traditional memory application. The read/write peripheral circuitry is designed in 65nm CMOS, which realizes a fully parallel write/read scheme proposed in Section III. The performance of the optimized resistive cross-point array architecture is benchmarked with the software approach running on Intel i7 3.4GHz 8-core processor for training the handwriting MNIST data set, as shown in Table 2. It is seen that the cross-point array area with 3×3 cells is still smaller than that of the peripheral circuitry, obeying the design
methodology on the limit of the reverse scaling rule. The latency and the energy for completing one iteration step (200 read operations and 1 write operation) in the sparse coding algorithm are compared. The resistive cross-point array architecture achieves >3,000 speed-up and >10^6 energy efficiency improvement in these most time-consuming steps (weighted sum and weight update) in the algorithm. Further study to benchmark all the steps in the algorithm and to include the I/O latency and energy into the calculation will give a thorough comparison between the software approach and the proposed ASIC approach with the resistive cross-point array architecture.

Table 1. Comparison of Resistive Cross-point Array for Different Applications

<table>
<thead>
<tr>
<th></th>
<th>Memory</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wire Width</td>
<td>20 nm</td>
<td>200 nm</td>
</tr>
<tr>
<td>Cell Spacing</td>
<td>20 nm</td>
<td>1 μm</td>
</tr>
<tr>
<td>Redundancy</td>
<td>~10% (ECC)</td>
<td>800% (3×3 Cells)</td>
</tr>
</tbody>
</table>

Table 2. Evaluation of Area, Energy and Speed-up in Computing

<table>
<thead>
<tr>
<th>Task</th>
<th>Software on CPU</th>
<th>Proposed ASIC (65 nm)</th>
<th>Improve ment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update Z</td>
<td>17.2 ms</td>
<td>5 μs (200 Read)</td>
<td>3440 ×</td>
</tr>
<tr>
<td>Update D</td>
<td>26.4 μs</td>
<td>84 ns (1 Write)</td>
<td>314 ×</td>
</tr>
<tr>
<td>Time for 1 Iteration</td>
<td>17.2 ms</td>
<td>5.01 μs</td>
<td>3430 ×</td>
</tr>
<tr>
<td>Energy for 1 Iteration</td>
<td>208 mJ</td>
<td>Cross-point Array: 1.03 nJ Peripher cil: 0.2 μJ</td>
<td>10^8 ×</td>
</tr>
<tr>
<td>Total Area</td>
<td>Intel i7 3.4 GHz (8 Cores)</td>
<td>Cross-point Array: 0.54 mm² Peripheral Circuitry: 0.57 mm²</td>
<td>□</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

Design methodologies have been developed for co-optimizing a resistive cross-point array architecture with machine learning algorithms. The read and write scheme that implements fully parallel operations of the weighted sum and the weight update is proposed. The digital spike encoding scheme is found to outperform the analog voltage encoding scheme in terms of the learning accuracy. In contrast to the conventional memory design, a set of reverse scaling rules is applied on the resistive cross-point array to achieve high learning accuracy. These include 1) larger wire width to reduce the IR drop on interconnects thereby increasing the learning accuracy; 2) use of multiple cells for each weight element to alleviate the impact of resistive device variation, at an affordable expense of area, energy and latency. The optimized resistive cross-point array with peripheral circuitry is designed in 65 nm node. Compared to state-of-the-art software approach running on CPU, it can potentially achieve >10^3 speed-up and >10^6 energy efficiency improvement, enabling real-time image feature extraction and learning.

REFERENCES