Benchmark Non-volatile and Volatile Memory Based Hybrid Precision Synapses for In-situ Deep Neural Network Training

Yandong Luo and Shimeng Yu
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
Email: shimeng.yu@ece.gatech.edu

Abstract — Compute-in-memory (CIM) with emerging non-volatile memories (eNVMs) is time and energy efficient for deep neural network (DNN) inference. However, challenges still remain for *in-situ* DNN training with eNVMs due to the asymmetric weight update behavior, high programming latency and energy consumption. To overcome these challenges, a hybrid precision synapse combining eNVMs with capacitor has been proposed. It leverages the symmetric and fast weight update in the volatile capacitor, as well as the non-volatility and large dynamic range of the eNVMs. In this paper, *in-situ* DNN training architecture with hybrid precision synapses is proposed and benchmarked with the modified NeuroSim simulator. First, all the circuit modules required for *in-situ* training with hybrid precision synapses are designed. Then, the impact of weight transfer interval and limited capacitor retention time on training accuracy is investigated by incorporating hardware properties into Tensorflow simulation. Finally, a system-level benchmark is conducted for hybrid precision synapse compared with baseline design that is solely based on eNVMs.

I. Introduction

Compute-in-memory (CIM) with emerging non-volatile memories (eNVMs) is a promising paradigm to accelerate the data-intensive computation of deep neural network (DNN) [1]-[4]. Here eNVMs are referred to as resistance-switching two terminal devices such as resistive random access memory (RRAM) or phase change memory (PCM) [5]. Although it has been demonstrated that CIM scheme is both time and energy efficient for DNN inference [1][2], challenges still remain for CIM based *in-situ* DNN training due to the following reasons: 1) significant training accuracy degradation due to the nonlinear and asymmetric weight update behavior of eNVM devices [6]-[8]. 2) the high write latency and energy consumption compared to SRAM technologies.

To enable reliable and efficient *in-situ* training, a hybrid precision synapse is proposed recently [9] [10], where eNVMs and capacitor are combined to leverage the non-volatility and large dynamic range of the eNVMs, together with the symmetric and fast charging/discharging behavior of the capacitor. During the training phase, the gradients are accumulated on the capacitor. After certain number of training batches, the weights stored in the capacitor are read-out and transferred to the eNVMs to avoid data loss. Only the weights on the eNVMs are used for inference after the training. Software comparable *in-situ* training accuracy and promising energy efficiency have been reported for multi-layer perceptron (MLP) network on simple MNIST dataset with hybrid precision synapse [9] [11].

However, whether this design is applicable to large-scale DNN training is questionable due to the limited data retention of capacitor when processing time is significantly increased. In this paper, *in-situ* DNN training architecture with hybrid precision synapse is designed for a larger convolutional neural network (CNN) on CIFAR-10 dataset. The hardware-software co-simulation on Tensorflow platform is performed to evaluate the training accuracy, and then the system-level benchmark is conducted with modified NeuroSim simulator [12]. The results show that about 90% *in-situ* training accuracy can be obtained for CIFAR10 dataset, although the impact of capacitor retention becomes more severe when processing DNN. The energy efficiency of training system based on hybrid precision synapse is improved by at least 3.11× compared with those using eNVMs based synapses.

II. Hybrid Precision Synapse for In-situ Training

As mentioned earlier, the nonlinear and asymmetric weight update behavior of eNVMs degrades the *in-situ* training accuracy. The expensive write operations to eNVMs leads to high training latency and poor energy efficiency. To alleviate these problems, capacitor based analog synapses such as 3-transistor-1-capacitor (3T1C) cell design [13] is proposed. Although capacitor based analog synapse offers symmetric and fast weight update, it could not support the subsequent inference after the training due to its volatile nature.

A hybrid precision synapse combines eNVMs and capacitor, as shown in Fig.1, which leverages the good linearity, fast programming of capacitor and the non-volatility, large dynamic range of eNVMs. The synaptic weight is divided into 2 parts: the first few bits of the weight has higher numerical significance and it is stored in the eNVMs, which is termed as high significance weight (HSW, W_{HSW}). The last few bits of the weight have lower numerical significance and it is stored in the capacitor, which is termed as low significance weight (LSW, W_{LSW}). Fig. 1 shows how 8-bit software weight is represented by 2-bit W_{HSW} and 6-bit W_{LSW}. 1 bit overlapping between W_{HSW} and W_{LSW} may exist if W_{LSW} precision is increased to 7bit. During training, only the LSW is frequently updated with the fast charging/discharging of the capacitor. A significance factor F is defined to represent the numerical significance of the HSW. Therefore, the weight stored can be represented as $W = F \times W_{HSW} + W_{LSW}$. In this paper, the significant factor is an integer power of 2, i.e. 2, 4, 8…, so that the multiply operation can be conducted by shifting the W_{HSW}. Besides, it is assumed that W_{HSW} and W_{LSW} are stored into separate arrays and the corresponding partial sums are added up digitally in the periphery.

In the hybrid precision synapse, the HSW is stored in a regular 1-transistor 1-resistor (1T1R) memory cell. The capacitor synapse is based on the 3T1C cell design as proposed in [13]. Two more transistors called power gate

978-1-7281-4123-7/20/$31.00 ©2020 IEEE

(PG) are added (compared with the original design [13]) to enable array level operation [11]. The LSW is programmed by charging or discharging the capacitor, which modulates the channel conductance of the transistor by tuning its gate voltage. During the charging phase, for example, a low voltage is applied to the gate of the PMOS transistor PG1 and a set of low voltage pulses are applied to the gate of the AG1 through the gate control lines PG1 and WL1, respectively. Zero gate voltage is applied to the two NMOS to turn them off. The discharging step is conducted in a similar way by turning on the NMOS transistors.

Weight transfer from LSW to HSW is conducted periodically to prevent capacitor leakage. The W_{LSW} is first read out and then converted to W_{HSW} by shift F bit. Then, the eNVM synapses are programmed to the desired levels. The LSB weight will be reset to zero after weight transfer. Fig. 1 illustrate the operations of a hybrid precision synapse.

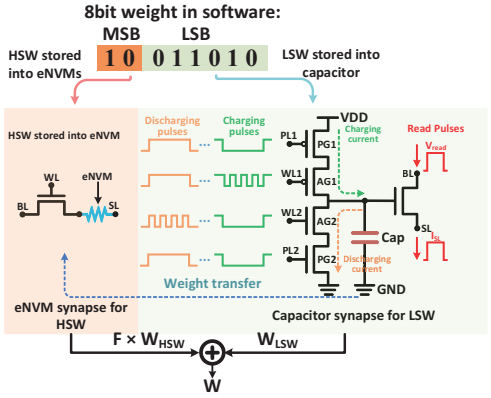


Fig. 1 Illustration of a hybrid precision synapse and its operation.

Fig. 2 shows the weight update curve of a capacitor synapse, where 128 conductance states (7bit) are stored in a 20fF capacitor. A foundry 28nm PDK is used here for SPICE simulation. The period of the charging/discharging pulse is 0.4ns. By choosing the appropriate amplitude of the pulse, the capacitor voltage increment (decrement) is about 2mV between two adjacent states. A relative symmetric weight update curve is obtained.

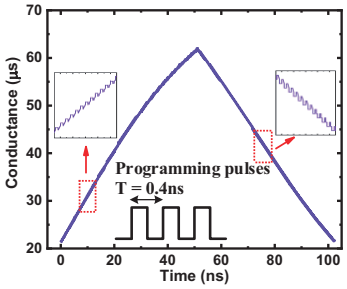


Fig. 2 Weight update curve of the capacitor for LSW.

III. In-situ Training Architecture Based on Hybrid Precision Synapse

A. Overall training architecture design

Fig. 3 shows the overall training architecture design for hybrid precision synapse. A 3-layer CNN is used as an example. 4 phases are involved during the training: the forward pass (FP) for activation calculation, the backward pass (BP) for error calculation (EC) and the gradient calculation (GC), and the weight update (WU). For the GC phase, more specifically, the gradients are calculated for layer i with its input feature maps (IFMs) I_i and the error δ_{i+1} from the deeper layer i+1. The error needs to be backpropagated to calculate the gradients of the shallower layers. In this architecture, FP, EC and GC for convolutional layers are all implemented with CIM. For the fully connected (FC) layers, the GC is conducted with digital multiplier while the other FP and EC steps are implemented with CIM.

To support the training architecture, two types of arrays, transposable arrays (T-array) are used for the FP and the EC step while non-transposable array (NT-array) are used for the GC step. During FP, the output feature maps (OFM) of CNN layers are calculated by the T-array with row input. The activations are stored into the NT-arrays for the GC step. During EC, the error of the *i* th layer δ_i is computed by the T-arrays with the error δ_{i+1} as column input. The gradients of each training image will be stored in a global buffer, which is accumulated over the entire batch with 200 training images. The WU step occurs after each training batch. The accumulated gradients are read-out from the global buffer and applied to the capacitor synaptic arrays for weight update. After certain number of training images (termed as transfer interval), weight transfer occurs by reading the LSW weight stored in the capacitor and transferring it to the HSW by programming the eNVM synapse.

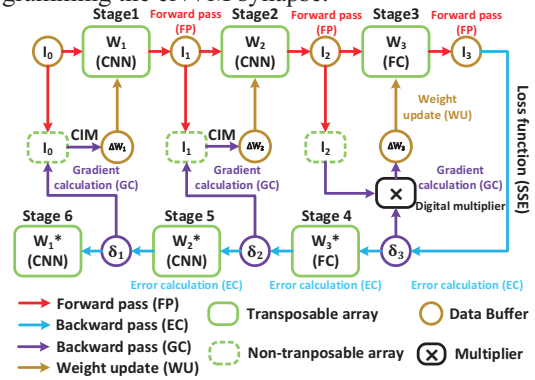


Fig. 3 Illustration of the training flow in a 3-layer CNN example.

B. Array level design

The general principles for array design follow that was proposed in [14]. The N-bit IFM is split into single bit input for N cycles. Shift and add is conducted for multi-bit input and multi-bit output. The partial sum current (I_{psum}) is sensed by the analog-to-digital converter (ADC) implemented by multi-level sense amplifiers (MLSA, a group of current mode sense amplifier). To represent the negative weight, there is a dummy column that is tuned to intermediate conductance $(G_{max} + G_{min})/2$ in each array. The actual partial sum value is obtained by subtracting the partial sum digits from regular columns and that from dummy column.

The schematic of the eNVM T-array is shown in Fig. 4(a), where one more transistor is added to each 1T1R memory cell to enable transposable read-out. The gates of the two access transistors are controlled by a horizontal word line (WL-H) and vertical word line (WL-V), respectively. During the FP

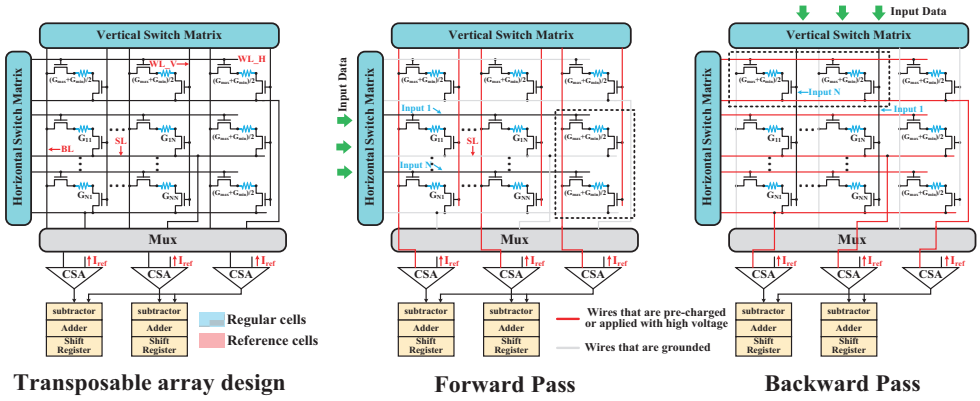


Fig. 4 (a) T-array design for eNVM synapse. (b) Array level operation in forward pass. The red lines are wires with a high voltage applied. The grey lines are the wires that are grounded. (c) Array level operation in the error calculation in backward pass.

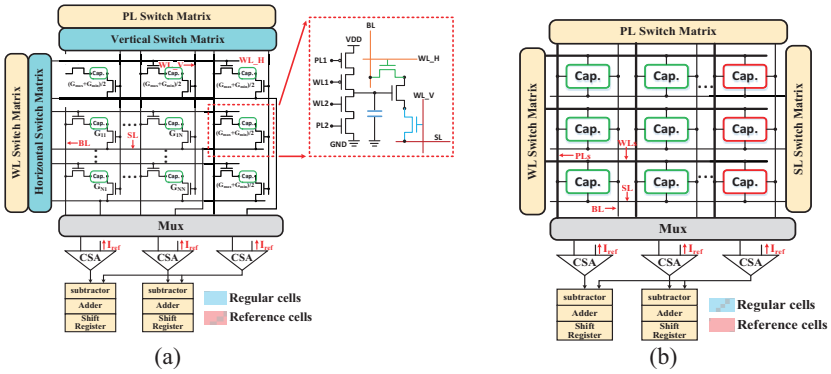


Fig. 5 (a) T-array design for capacitor synapse used in FP and EC step. (b) NT-array design for capacitor synapse used in GC step.

(Fig. 4(b)), the IFMs are applied to the WL-H by the horizontal switch matrix (H-switch matrix) to turn on the corresponding rows while the WL-V are fully turned on by the vertical switch matrix (V-switch matrix) to connect the memory cells to the grounded SL. The BL is connected to the MLSA by mux and is precharged to sense the I_{psum} along the column. During the EC step, the error δ_{i+1} from the deeper layer i+1 is input at the WL-V through the vertical switch matrix and the WL-H is applied with a high voltage to fully turn on the access transistor. In this step, the SL is connected to the CSAs by mux to sense the I_{psum} along the row for a transposed read-out. The BL is grounded in this step, as illustrated in Fig. 4 (c). For the NT-array design of eNVM synapses, regular 1T1R memory cell is used. In this paper, the array size is 128 × 128.

During programming, the eNVM cells are programmed row-by-row. the H-switch matrix turned on the WL-H of the row to be programmed while a high voltage is applied to the WL-V by V-switch matrix. The programming pulse is applied to either the SL or BL while the other is grounded, depending on the polarity of the programming voltage.

T-array for capacitor synapse follows a similar design, where two more access transistors are added, as shown in Fig. 5(a). Similarly, the H-switch matrix and V-switch matrix are for the FP and EC step in BP. Compared with the T-array design for eNVM synapse, two more switch matrices: the PL switch matrix and WL switch matrix are added to program the capacitor synapse. The control wires for capacitor programming are not shown in Fig. 5(a). The NT-array design for capacitor synapse is shown in Fig. 5(b). Only WL switch matrix and PL switch matrix are needed. In the NT-array, two capacitors are grouped to store an 8-bit activation where 4bits are stored in each capacitor. The reason to use two capacitors is that it is difficult to store 8-bit (256 states) into one capacitor, which is limited by the voltage step between adjacent states.

C. Chip level design

The chip level architecture design is shown in Fig. 6. Since eNVM synapse and capacitor synapse are separated, there are two types of processing element (PE) in the system: the PE that contains T-arrays for FP and EC (T-PE), and the PE that contains the NT-arrays for the GC step of CNN layers (NT-PE). Then the T-PE groups the T-arrays of eNVM synapse and capacitor synapse to form a "super" array. For example, if the HSW is 2 bit and LSW is 6 bit, 2 eNVM arrays (with 1 bit per cell) and 1 capacitor (with 7 bit per cell with 1 bit overlapping) array are grouped. Each T-PE contains 9 "super" arrays corresponding to the 3×3 filter [15]. During the FP (or EC), the IFMs (or the error) are delivered to each array from the input buffer. The output partial sums from the eNVM arrays and the capacitor array are first added up according to the significance factor F. Then, the partial sum output from the different "super" arrays are added up (if they are of the same output neuron) and stored into the output buffer to be transferred outside the PE. The NT-PE contains NT-arrays based on capacitor synapse to store the activations of each

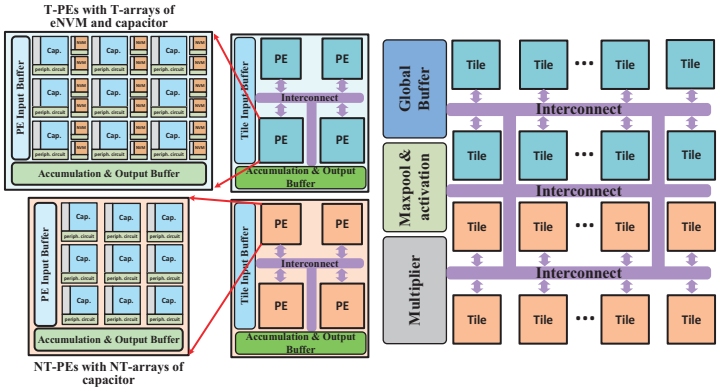


Fig. 6 The hierarchy of the training architecture designs: (from left to right) the PE level design, the tile level design and the chip level.

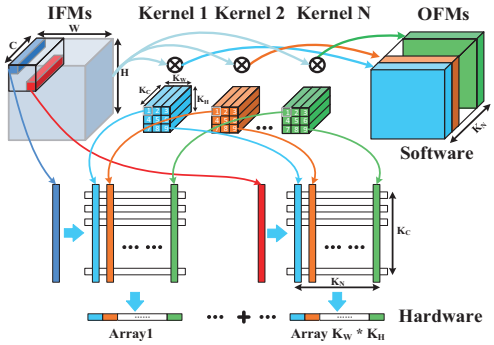


Fig. 7. The novel mapping scheme for convolutional layers for FP. Similar mapping scheme is used for the EC step.

layer for the later GC step.

In the tile level, four PEs are grouped in this design. The tile level input buffer stores the IFMs and deliver them to the PEs within the tile. The output partial sum from PEs can be added up in the tile level accumulation unit if they are of the same output neuron.

The partial sum output from tiles will first be sent to the chip level accumulation unit to be added up (if needed) and the output will then be sent to the ReLU unit and Maxpooling unit to get the activations. The output activation is stored into the global buffer temporarily and then delivered to the tiles of the next layer for processing. Digital multipliers are also used on the chip level for the GC step for FC layers.

D. Mapping scheme for training

For FP, the convolutional kernels can be mapped into the memory arrays with the novel mapping method proposed in [15] (as shown in Fig. 7) while the FC layers are mapped with the conventional mapping method mentioned in the same reference. The IFMs will be sent to the corresponding array as row input to get partial sum. The output partial sums from these k×k arrays are added up to get the final output. Array partitioning is needed if the kernel size is large. The same mapping scheme can be used for EC step except that the error is delivered to each array as column input for transposed computation.

For the GC step of a convolutional layer, the IFMs of this convolutional layer is regarded as the convolutional kernel and stored into the capacitor-based NT-arrays. Using the conventional mapping in [15], the IFM pixels of the same channel are unrolled into a long bar and mapped along a column of the memory array, as shown in Fig. 8. Similarly, the error pixels in channel k is unrolled into a long bar and delivered to the NT-array as input to compute the gradients of the k^{th} CNN kernel. For example, in Fig. 8, the gradients of the blue kernel is obtained by convolution of the blue channel of the error with the IFM of layer i.

IV. Benchmark Methods

A. Training accuracy

The *in-situ* training accuracy is evaluated by incorporating the hardware properties into the software simulation in Tensorflow. The WAGE code [16] is modified to incorporate the nonlinear and asymmetric weight update behavior of the eNVM device as illustrated in [6]. The capacitor leakage is considered by multiplying a retention ratio to the LSW after each batch. The weight, activation, error and gradient are all quantized to 8-bit during the training. With the VGG-8 network in WAGE, software baseline training accuracy ~91% is obtained for CIFAR-10 dataset after 150 training epoch.

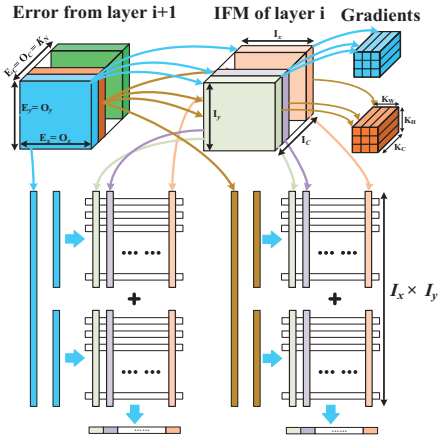


Fig. 8. The conventional mapping scheme for gradient calculation (GC) for a convolutional layer.

B. System level performance estimation

The system level performance benchmark is conducted with modified NeuroSim simulator at 32nm node [14]. RRAM with 1-bit per cell (1-bit RRAM) and 2-bit per cell (2-bit RRAM) are considered for HSW in the hybrid precision synapse. The device parameters used in the simulation are listed in TABLE I, which is obtained from the experiment data in [17]. For 1-bit RRAM, N RRAMs are grouped to represent an N-bit W_{HSW}. 2-bit RRAM can reduce the area cost due to the higher storage capacity in each cell but it suffers from more write-verify cycles and therefore higher latency and energy consumption as suggested by [17].

The training architecture based on solely digital eNVM synapses is used as the baseline, which means for 8-bit weight, 8 1-bit RRAM cells (or 4 2-bit RRAM cells) are used in T-array. The IFM of each layer will also be stored into the RRAM based NT-array for the GC. Analog RRAM with high cell precision (e.g. 8bit) is not considered here due to its

asymmetry weight update behavior and poor training accuracy (see Section V.A).

TABLE I. The Device Parameters Used in the Simulation

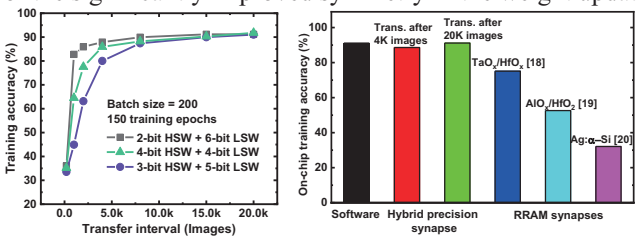
	Capacitor synapse	1-bit RRAM [17]	2-bit RRAM [17]
R_{ON}	16.1kΩ	21kΩ	7.35kΩ
R_{OFF}	46.5kΩ	500kΩ	500kΩ
Cell size	~2000F²	31F²	31F²
Reset pulse width	0.4ns	200ns	200ns
Reset pulse amplitude	High: 0.41V Low: 0V	3.8V	3.8V
Reset pulse number	128 (7-bit) /16 (4-bit)	1	1
Set pulse width	0.4ns	100ns	100ns
Set pulse amplitude	High 1V Low: 0.6V	2.1V	2.1V
Set pulse number	128 (7-bit) /16 (4-bit)	1	10

V. Results and Discussions

A. Training accuracy

First, the impact of weight transfer interval on training accuracy is studied without considering the capacitor leakage. The results are plotted in Fig. 9. When the weight transfer interval is small, the amount of gradients accumulated at LSW is small. Therefore, a large portion of the LSW is eliminated when dividing the significance factor before adding them to the HSW. With different HSW bit precision, the training accuracy is slightly different when the weight transfer interval is smaller than 15K images. Since 2-bit HSW + 6-bit LSW shows the highest training accuracy, in the rest of the paper, only this configuration is considered. It can also be noted that all of the three HSW configurations can achieve ~91% training accuracy when the weight transfer interval is large, i.e. 20K images.

Fig. 9(b) compares the training accuracy achieved by hybrid precision synapse with state-of-the-art analog RRAM devices (TaO_x/HfO_x [18], AlO_x/HfO_2 [19], Ag: α-Si [20]). Here 8-bit weight is stored in single analog RRAM. It can be concluded that hybrid precision synapse achieves better *in-situ* training accuracy than analog RRAM synapses because of the significantly improved symmetry in the weight update.

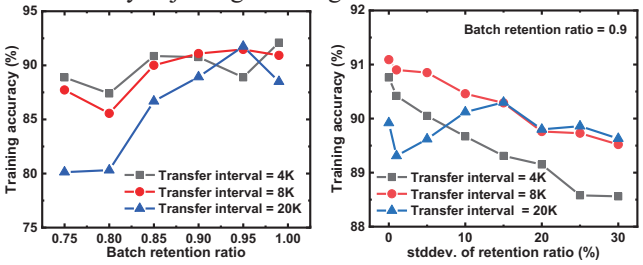


(a) (b)

Fig. 9. (a) The training accuracy vs. weight transfer intervals with different HSW bit precision (b) Comparison between hybrid precision synapse and analog RRAM synapses.

Fig. 10 (a) shows the *in-situ* training accuracy with capacitor leakage. The effect of capacitor leakage is modeled as the batch retention ratio, which is defined as the ratio between the weight before and after a training batch. Batch processing time, leakage current and the capacitance determines the batch retention ratio. With about 1.92ms batch processing time, average leakage current of 0.267pA and 20fF capacitance, the batch retention ratio is about 0.9 by $\Delta V_{cap} = I_{leakage\_avg} \Delta t / C_{cap}$, assuming the W_{LSW} is proportional to the voltage of the capacitor V_{cap}. Only the leakage for the W_{LSW} is considered in the simulation as the weight transfer interval is a few thousands images in this paper. The leakage in the NT-arrays that store activations are not considered because the longest life time of the activations is the same as the time that one image stays in the pipeline. With a weight transfer interval of 20K images, the training accuracy decays as retention ratio is increased. It can be explained by the fact that the gradients of older batches suffer from more severe decaying with a larger transfer interval. When the transfer interval is reduced to 4K or 8K, it is observed that the training accuracy first gets improved when the retention ratio is reduced then it starts to drop when retention ratio is further reduced. It can be attributed to the fact that the gradients of each batch suffer from less decaying with such small transfer interval. Besides, the retention ratio has similar effect as the decay rate in Momentum, which is a hyper-parameter. Therefore, training accuracy fluctuates as the retention ratio (hyper-parameter) changes.

Due to the process variation, the retention ratio of each capacitor is different. The impact of retention ratio variation on training accuracy is studied by generating individual batch retention ratio for each capacitor from Gaussian distribution and then clipped to values between (0,1]. The standard deviation σ_{RET} is normalized to the mean value of the retention ratio. From Fig. 10 (b), when the transfer interval is 4K and 8K images, the training accuracy drops from about 91% to less than 89.5% and 88.5 %, respectively, as σ_{RET} is increased to 30%. It can be attributed to more capacitors at the distribution "tail" with poor retention when σ_{RET} is large. However, for larger transfer interval 20K image, a slight rise of the training accuracy is first observed before it drops to about 89.6%. The slight accuracy rise can be explained by the fact that the portion of capacitors with high retention ratio (>0.9) is increased as σ_{RET} rises slightly, which helps improve the training accuracy according to the results in Fig. 10(a). However, when σ_{RET} is large, the capacitors with poor retention ratio degrades the training accuracy. From the results in Fig. 10, it can be concluded that the impact of capacitor leakage on training accuracy is small and it can be alleviated by adjusting the weight transfer interval.



(a) (b)

Fig. 10 (a) The training accuracy with different batch retention ratios. (b) The training accuracy vs. retention ratio variation.

B. System level performance benchmark

With the device parameters in TABLE I, system level benchmark results are summarized in TABLE II. The energy consumption and latency here are for a training batch (200 images). From the results, first, training systems based on hybrid precision synapse show lower energy consumption than those are solely based on RRAM synapses. It can be explained by the following reasons: 1) Less energy consumption in FP as less arrays need to be read out. For example, for hybrid precision synapse with 1-bit RRAM as 2-bit HSW, only two RRAM arrays and one capacitor arrays are needed to be read. But for 8-bit weight with 1-bit RRAM, the weight needs to be stored in 8 RRAM arrays. Therefore, more energy is consumed for the digital RRAMs. 2) Less write energy consumption to store the activations of each layer into the NT-arrays for in-memory gradient calculation. For hybrid precision synapse, the activations are stored into the capacitor arrays and the programming energy for capacitor is low. For RRAM synapse, the programming energy consumption is high, especially for 2-bit RRAM because write and verify pulses are needed. Therefore, it is noted that energy efficiency is improved by 3.11× and 4.44× for hybrid precision synapse with 1-bit RRAM and 2-bit RRAM as the HSW, compared with the two baselines, respectively.

TABLE II. The System Level Benchmark Results (32nm Node)

	Hybrid cell (1bit RRAM)	Hybrid cell (2bit RRAM)	Sole RRAM (1bit RRAM)	Sole RRAM (2bit RRAM)
Chip Area	**300.7 mm²**	**289.0 mm²**	**294.6 mm²**	**215.7 mm²**
Total Energy consumption	97.72 mJ	89.76 mJ	305.75 mJ	397.38 mJ
Energy FP	28.21 mJ	23.10 mJ	62.28 mJ	41.81 mJ
Energy BP-EC	27.46 mJ	22.48 mJ	60.62 mJ	40.70 mJ
Energy BP-GC *	**39.72 mJ**	**39.72 mJ**	**177.41 mJ**	**300.92 mJ**
Write energy BP-GC	**0.28 mJ**	**0.28 mJ**	77.45 mJ	**234.55 mJ**
Read energy BP-GC	39.44 mJ	39.44 mJ	99.96 mJ	66.37 mJ
Energy WU**	2.33 mJ	4.46 mJ	5.44 mJ	13.95 mJ
Latency/Batch	**1.92 ms**	**1.92 ms**	8.21 ms	**32.74 ms**
Energy efficiency (TOPS/W)	**3.83**	**4.17**	**1.23**	**0.94**

* The energy consumption for storing the activations is included
** The energy consumption for weight transfer is included for hybrid precision synapse. It is averaged to each batch

As for the latency, the training systems solely based on RRAM shows much higher latency due to the latency for storing the activations into NT-arrays, especially for writing 2-bit RRAM with write-verify. The area cost for hybrid precision synapse is similar to that of 1-bit RRAM although the cell area of the capacitor synapse is significantly larger than a RRAM cell. It can be attributed to the facts that the area of an array is dominated by the periphery circuits and that the capacitor synapse can store multiple bits in one cell.

VI. Conclusions

In this paper, *in-situ* DNN training architecture with hybrid precision synapse is designed and system level benchmark is conducted. First, by incorporating hardware properties into Tensorflow simulation, it is concluded that hybrid precision synapse can achieve about 90% training accuracy even when considering the data retention degradation and variation of the capacitor. NeuroSim benchmark reveals that training systems based on hybrid precision show significantly lower latency and energy consumption than those solely based on eNVMs with similar area cost.

Acknowledgements

This work is supported by ASCENT, one of the SRC/DARPA JUMP research centers and SONY research.

References

[1]. P. Chi et al., "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2016.
[2]. A. Shafiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2016.
[3]. L. Song, X. Qian, H. Li and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017.
[4]. M. Cheng et al., "TIME: A training-in-memory architecture for RRAM-based deep neural networks," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 834-847, 2019.
[5]. S. Yu and P. Chen, "Emerging memory technologies: recent trends and prospects," in *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, pp. 43-56, 2016.
[6]. X. Sun and S. Yu, "Impact of non-Ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 3, pp. 570-579, 2019.
[7]. P. Chen et al., "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 194-199.
[8]. G. W. Burr et al., "Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power)," *IEEE International Electron Devices Meeting (IEDM)*, 2015, pp. 4.4.1-4.4.4.
[9]. S. Ambrogio et al., "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 558, no. 7708, p. 60, 2018.
[10]. X. Sun, P. Wang, K. Ni, S. Datta, and S. Yu, "Exploiting hybrid precision for training and inference: A 2T-1FeFET based analog synaptic weight cell," *IEEE International Electron Devices Meeting (IEDM)*, 2018.
[11]. Y. Luo, P. Wang, X. Peng, X. Sun and S. Yu, "Benchmark of ferroelectric transistor based hybrid precision synapse for neural network accelerator," in *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*. doi: 10.1109/JXCDC.2019.2925061
[12]. X. Peng, S. Huang, Y. Luo, X. Sun and S. Yu "DNN+NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," *IEEE International Electron Devices Meeting (IEDM)*, 2019.
[13]. Y. Li et al., "Capacitor-based cross-point array for analog neural network with record symmetry and linearity," *IEEE Symposium on VLSI Technology*, 2018.
[14]. P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067-3080, Dec. 2018.
[15]. X. Peng, R. Liu and S. Yu, "Optimizing weight mapping and data flow for convolutional neural networks on RRAM based processing-in-memory architecture," *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019.
[16]. S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," 2018, arXiv:1802.04680.
[17]. S. Yin, et al. "Monolithically integrated RRAM and CMOS based in-memory computing for efficient deep learning," *IEEE Micro*, 2019.
[18]. W. Wu, et al. "A methodology to improve linearity of analog RRAM for neuromorphic computing." *IEEE Symposium on VLSI Technology*, 2018.
[19]. J. Woo, et al. "Improved synaptic behavior under identical pulses using AlOx/HfO2 bilayer RRAM array for neuromorphic systems." *IEEE Electron Device Letters*, 37.8 (2016): 994-997.
[20]. S. H. Jo, et al. "Nanoscale memristor device as synapse in neuromorphic systems." *Nano Letters*, 10.4 (2010): 1297-1301.