Energy Efficient Approximate Storing to MRAM for Deep Neural Network Tasks in Edge Computing

Yoshinori Ono
Graduate School of Science and Engineering
Shibaura Institute of Technology
Tokyo, Japan
ma19013@shibaura-it.ac.jp

Kimiyoshi Usami
Department of Computer Science and Engineering
Green Innovation Center
Shibaura Institute of Technology
Tokyo, Japan
usami@shibaura-it.ac.jp

Abstract - On-chip learning is gaining attention in edge devices. In addition, a magnetic RAM (MRAM) is a promising memory technology for edge devices because of low leakage energy. However, the high write energy is a disadvantage of MRAM. For minimizing the write energy, we propose an approximate storing approach to MRAM for learning tasks of deep neural networks (DNN). The proposed approach writes the weight and bias data to NVM approximately on each epoch with the fine-grained adjusted write time. Simulation results with image recognition DNN applications have demonstrated the write energy can be reduced range from 9% to 37% while negligible (< 0.5%) accuracy loss.

I. Introduction

Edge computing devices are spreading world-wide at tremendous speed because of developing Internet of Things (IoT), smartphones and wearable devices. In addition, the effectiveness of machine learning (ML) such as deep neural network (DNN) has led to developing a lot of systems and services featuring DNNs. This trend also has led to increasing the researches to perform DNN tasks on edge computing devices.

DNN tasks are divided into two main categories: learning and inference. Learning task on edge devices (on-chip learning) was considered unsuitable compared with inference task because of their high computational cost and enormous energy dissipation. Most of the researches did training task in cloud servers and did inference task on edge devices by transferring a training model from the cloud server. However, sending the private data to servers poses a security problem. In addition, transferring the input data captured at the edge device to the server and sending back the training model to the edge device consumes energy dissipation. Because of this, researches on on-chip learning with domain specific and private data have begun to emerge recently [1-2].

On-chip learning at edge devices requires more energy-efficient technologies than the learning at the server, because the edge devices need to operate with limited energy. Another challenge is the memory capacity in on-chop learning. Since the learning tasks require the memory system for error propagation and weight updating, the memory capacity required for learning task is significantly greater than that for inference task [2]. Modern convolutional neural networks (CNN) which is one of the DNNs use millions of weights and activations, leading to critical challenges for both computation and data transmission [3].

Under these circumstances, the memory system becomes more important in on-chip learning processors. In most computer systems, SRAM and DRAM are used for the memory systems. However, these have to be kept powered-on to keep the data, resulting in consuming energy. Dynamic energy for refresh operations in DRAMs and leakage energy in SRAMs are consumed during the power-on state. Due to the spread of IoT and mobile devices, lower energy dissipation is required, and hence new memory systems that consume less energy have been studied.

Among new memory systems, a non-volatile memory (NVM) especially employing a magnetic tunnel junction (MTJ) is promising technology [4]. MTJ has the property that the resistance value changes by manipulating the orientation of electron's spin in magnetic materials. By taking the advantage of this property, MTJ can store data. In addition, MTJ can reduce leakage current compared to SRAM because data in MTJ is not lost even with power-off. As MTJ-based memory systems, a spin transfer torque magnetic RAM (STT-MRAM) technology has been studied so far. However, MTJ has a disadvantage that the write operation consumes significantly larger energy than the read operation [5]. Hence, it is required to reduce the write energy in various aspects such as circuit structures, architecture and operation methods.

Approximate computing (AC) techniques reduce energy and execution time by allowing incorrect results using the property that information can be recognized in spite of noisy data. AC has been gaining traction as a computing paradigm for a wide range of cognitive applications that aim to extract deep insight from vast quantities of data. Since DNNs have robustness to noise and resiliency to numerical errors, researches on the application of AC to DNNs have been reported [6-7]. AC is also effective for memory systems. In particular, "precision scaling" is a well-known AC technique to be applied to memory systems. It allows us to reduce computation and storage resources by tailoring the bit width of data [8-9]. AC technique based on the precision scaling, which is related to the proposed approach, is discussed in Section 2.

In this paper, we propose an approach to augment the capability of the precision scaling to reduce the write energy of MRAM for DNN applications. The contributions of this paper are summarized as follows:

- We proposed a new energy-efficient approximate storing approach for MRAM. By employing precision scaling ideas, we split bits of the floating-point (FP) fraction part into groups and perform the write operation to each group with different write times to MRAM to minimize the write energy.
- We applied the proposed approach to the learning task of DNNs on edge devices. The weight and bias data are stored in MRAM and repeatedly updated at every epoch by using the proposed approach.
- We executed simulations using a statistical model for the write time of MRAM and three different network types of DNN. Through this simulation, we quantitatively investigated the trade-off between the write energy and accuracy when changing the write time and the bit groups.
- We demonstrated that the proposed approach allows us to reduce the write energy by 25% in CNN and by 9% in MobileNetV2 with less than 0.5% accuracy loss as compared to a non-AC technique. By relaxing the accuracy loss to 7.5%, the write energy can be reduced by 38% in CNN and by 19% in MobileNetV2 by the proposed approach.

The rest of the paper is organized as follows: Section 2 introduces the prior works. Section 3 describes a proposed precision scaling approach of FP numbers for MRAM. Section 4 shows the application of the proposed approach to weights and bias data of DNNs. Section 5 describes the simulation setup for the evaluation. The results of simulation are presented in Section 6. Section 7 concludes the paper.

II. Prior Works

A. *Precision Scaling for Floating-Point Number*

Nowadays most embedded applications involving numerical computations with large dynamic range are performed using *binary64* (double-precision: "FP64") or *binary32* (single-precision: "FP32") FP formats, described by the IEEE 754 standard. However, the execution of FP operations emerges as a major contributor to the energy dissipation. To provide a compromise between energy cost and dynamic range, IEEE 754 introduces a 16-bit format referred to as *binary16* (half-precision: "FP16").

In recent ML researches, *bfloat16* (16-bits brain floating point: "BF16") was introduced as new FP types (Fig. 1 and Table I). The paper [10] described that its chief advantages are (i) ease of replacing FP32 by BF16 in DNNs while retaining correct DNNs operation, (ii) improved performance relative to FP32 due to greater memory bandwidth, (iii) software can easily implement BF16 with existing FP32 instructions using zero-padding for converting BF16 numbers to FP32 or masking and shifting for converting FP32 numbers to BF16. Although BF16 offers less precision than FP16, it is better suited to support deep learning tasks due to enough dynamic range [11].

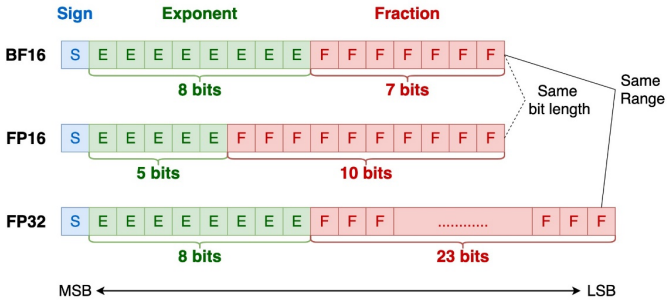


Fig. 1 Floating-Point Formats

TABLE I Dynamic Range of Floating-Points

FP Type	Min.	Max.
BF16	$\approx 9.2 \times 10^{-41}$	$\approx 3.4 \times 10^{38}$
FP16	$\approx 5.9 \times 10^{-8}$	$\approx 6.5 \times 10^{4}$
FP32	$\approx 1.4 \times 10^{-45}$	$\approx 3.4 \times 10^{38}$

B. *Quality-Configurable Non-volatile Memory*

Studies of employing bit precision AC technique for reducing energy of NVM systems have been reported. *Ranjan et al.* explored structures and characteristics of STT-MRAM and studied how to apply AC effectively [12]. They proposed a quality-configurable memory (QCMEM) which can control read/write current and period automatically by detected quality with the extended instruction set architecture. In [12], several approximate techniques for read/write operations have been described. Among them, we focus on the write energy in this paper. This is because the results in [12] showed that the write energy is one to two orders of magnitude larger than the read energy.

To mitigate write energy, they experimented to reduce the write time in exchange for some write failures. In the experiment, they introduced an automatic tuning framework which can control output quality using the bit groups and the quality field and applied to ML processes. The bit group is a set of bit data obtained by dividing one data into several groups. For example, if there is 8-bit data, we can divide into 4 groups each of which has 2 bits. Needless to say, there are many positions for dividing bits. The quality field is designated to set the error probabilities for each bit group. They reported that the number of bit groups is an important parameter for controlling trade-off between the output quality and overhead to write. Although they set 4-bit groups in their experiment, the best way to divide bit groups was not discussed.

C. *Fine-Grained Splitting Bit Groups and Implementation*

To resolve the problem of [12], Authors of [13] proposed a new splitting bit groups for fine-grained precision scaling. They defined the "bit split position" (BSP) to realize fine-grained and freely adjusting the position of the dividing bits in the integer value. By employing the precision scaling concept, they allowed the different write times for the bit groups split

by BSP. In addition, to implement the BSP approach to an LSI chip with non-volatile flip-flops (NVFFs), they also proposed the "store-domain" for setting the different write time for each group with simpler peripheral circuits than [12] and fit to the bit groups. The concept of store-domain was originally introduced in [14] to divide NVFFs into some groups and control them independently. Each store-domain can perform read/write (restore/store) operation independently. However, the target application in [13] was image data and image processing.

III. Proposed Approach for Floating Point Numbers

A. Splitting bit groups in Floating Point Fraction Part

First, we extend the BSP ideas to the fraction part in floating point numbers for DNN processes (Fig. 2). Like the approach in [13], there are only two groups of bits, the left group (Group L) and the right group (Group R), but we can adjust the position where to separate into groups. In Fig. 4, dotted arrows are the candidates of BSP. In this paper, the BSP is defined as the number of bits of Group R. With this splitting method, we investigated how adjusting the position of dividing the groups and the NVM's write time for each group affects the trade-off between energy and test accuracy of DNNs.

The main difference from [13] is the data types. Although they only considered the integer values for image processing, we propose an extension for FP for the DNN processes in this paper. From [11], the exponent part is more important than the fraction part in order to maintain dynamic range. Therefore, we apply BSP only to the fraction part.

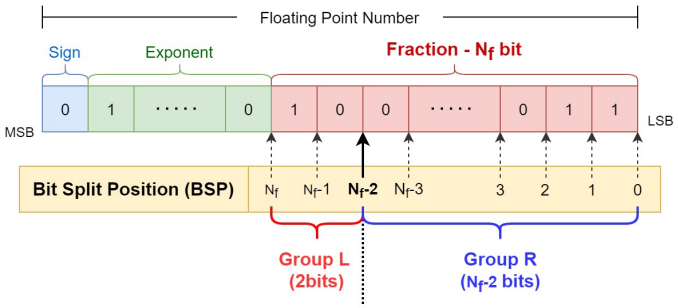


Fig. 2 Bit split position (BSP) for FP values

B. Non-volatile Memory with Store Domain

Because modern CNNs use millions of weights and activations [3], DNN processors especially treating the modern CNNs require a large-scale memory. Hence, in this paper, we chose STT-MRAM instead of non-volatile flip-flops as the memory system. However, The basic structure of STT-MRAM is depicted in Fig. 3.

First, we allow it to control the write time for each store-domain. Next, we assign a bit group to each store-domain. By these approaches, we can easily realize the bit groups and different write time for each bit group. Fig. 4 shows the conceptual diagram of assigning bit groups to store-domains. When N_f-bit fraction of FP data are store into NVFFs, it is divided into Group L and Group R. Data assigned to Group L are written to one of the store-domains (e.g. store domain No. 0) to use the "long write time", while data assigned to Group R are written to another store-domain (e.g. store domain No. 1) to use the "short write time".

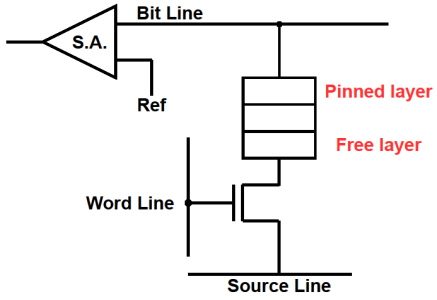


Fig. 3 Spin transfer torque (STT) MRAM

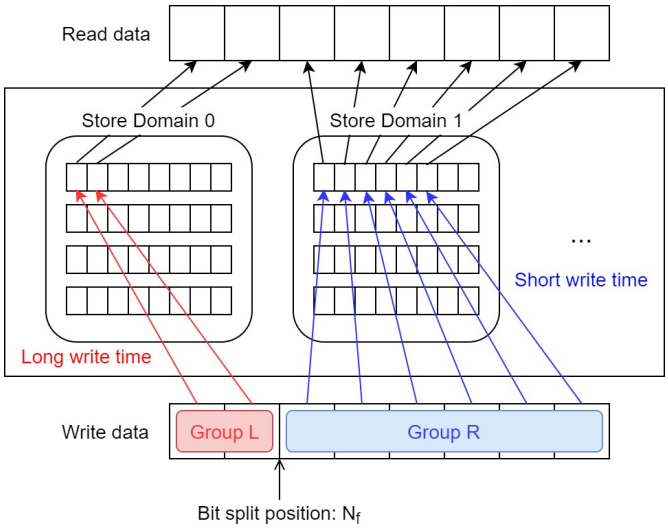


Fig. 4 Assigning bit groups to store domains

IV. Application of Proposed Approach to DNNs

In this chapter, we describe how to achieve low energy dissipation using the proposed approach for DNN applications. We also describe the advantage of applying this approach to DNNs.

A. Learning vs. Inference

In this paper, we focus on the learning tasks rather than the inference tasks in DNNs. In DNN processes, memory accesses for the inference tasks are mainly reading weights and biases. There is few write operation to update them. Meanwhile, the learning tasks perform the write operation at backpropagation and updating weights. In addition, even if an instantaneous power surge occurs during the training,

the updated weight data are retained in the NVM and does not volatilize. It is advantageous to allow us to resume the training from the timing of the power surge. Therefore, we chose the learning tasks as a target of the proposed approach.

B. Target Data for Approximate Storing

We restrict the data stored in MRAM to the weight and bias data in this paper. The capacity of MRAM is not enough to store all neural data while learning tasks. [15] reported the implementation of 1Gbit (≈128Mbyte) STT-MRAM in 28nm FDSOI technology. However, all neural data of modern DNNs require the memory capacity on the order of gigabytes depending on the input data size and the batch size. The weight and bias data are in the order of megabytes.

V. Simulation Settings

A. Device-level Settings

In a pre-processing, we evaluated the write energy dissipation per an MTJ by SPICE simulation in a 65nm process technology. The supply voltage VDD was assumed to be 1.2V. All memory data was initialized to "0" in preprocessing. In addition, we specified the distribution function for T_{th} variations. We defined T_{th} as the threshold time for the successful write operation to MTJ. If the write time is longer than T_{th}, we assumed that data are successfully stored, otherwise it fails. Among proposed distribution functions for the write time of STT-MRAM, we chose the simplest model [16] to use a normal distribution in our experiment. We assumed that T_{th} varies in a normal distribution with the average value of 8ns and the standard deviation σ of 3ns.

It is also important to set candidates of long and short write times. T_L is defined as the long write time for Group L and T_R is defined as the short write time for Group R. Fig. 5 shows the error rate of an STT-MRAM as a function of the write time. The error rate is 0.004% when it is set to 15ns. By considering recent MRAM papers [17-18], we assume that an MRAM is made to operate at the clock frequency of 200MHz. Since the clock cycle time is 5ns, we control T_L and T_R in multiples of 5ns. T_L requires 20ns as reported in [13] to maintain lower error rates for higher bits in data. In addition, because the error rate of 15ns is very close to that of 20ns (Fig. 5), we omitted 15ns for the candidate of T_R. Therefore, we chose 20ns for T_L and [10ns, 5ns, 0ns (No write)] for T_R.

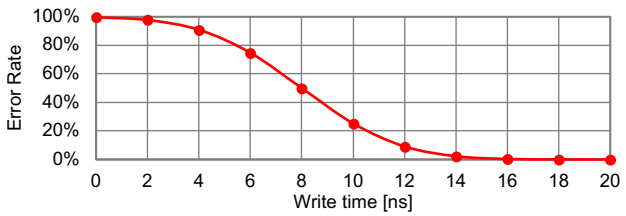


Fig. 5 Write time vs. error rate of an STT-MRAM

B. System-level Settings and Application Benchmarks

We employed TensorFlow to perform DNN applications. TABLE II summarizes the application benchmarks of image recognition that we used in this paper. We prepared 60,000 training data and 10,000 test data at every benchmark. The total epoch count was set to 20 and the batch size was set to 128. We simulated the system on an Intel i9-10900K CPU and two Nvidia GeForce RTX 2080 Super GPUs (linked by Scalable Link Interface) with the Ubuntu 20.04 LTS Operating Systems.

The flow of approximate storing is the following: (i) we get the weight and bias data of the model for each epoch, (ii) the data are decomposed into bit data, (iii) the success or failure of the store are determined based on the write time and BSP settings, (iv) writing back the weight and bias data to the model. We built this flow into the TensorFlow processes and performed it on every epoch.

In addition, our target is on-chip learning. We require a data format with low energy even if we ignore some accuracy degradation. Because of this, we set FP type to BF16 when getting weight and bias data in the simulation. Therefore, the fraction part has 7 bits (N_f = 7 bits).

TABLE II Application Benchmarks

Network Type	Layers	Dataset	Params*
Multi-layer Perceptron (MP)	3**	MNIST	118K
CNN	4***	CIFAR-10	122K
MobileNetV2 (MN) [3, 19]	16****	Fashion-MNIST	719K

* Parameters include weights and biases
** The number of Fully Connected layers
*** The number of Convolutional layers
**** The number of Residual bottleneck layers [19]

VI. Results

A. BSP vs. Test Accuracy

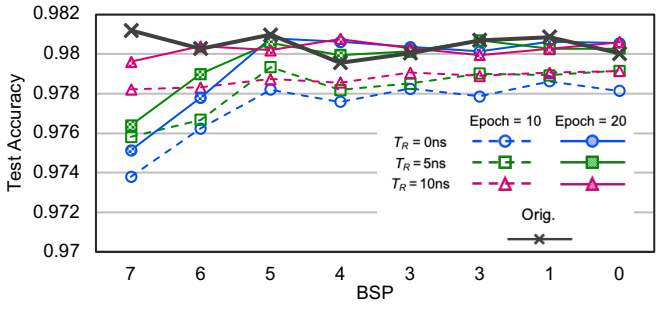
Fig. 6 shows the test accuracy for BSP. The basic trend of all network types is that the test accuracy becomes higher when BSP gets closer to the MSB side (7) and that becomes lower when BSP gets closer to the LSB side (0).

The accuracy shows a similar trend for Multi-layer Perceptron (MP) and CNN. When T_R = 0ns (blue dots) and 5ns (green square), the accuracy of MP and CNN at BSP = 6 or 7 is decreased. Meanwhile, those of MobileNetV2 (MN) is different. When T_R = 0ns (blue dots) and 5ns (green square), decreasing the accuracy occurs at BSP = 4 to 7. In addition, when T_R = 10ns (pink triangle), it occurs at BSP = 5 to 7.

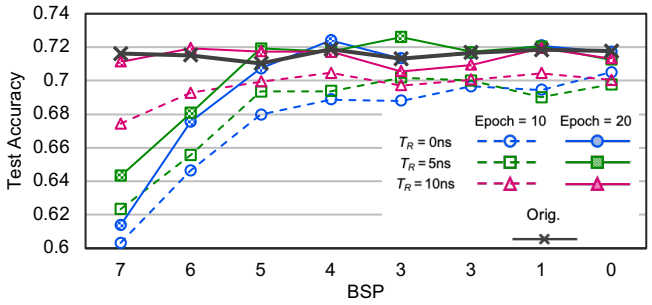
Moving BSP to the MSB side and the fewer T_R mean that the low-precision bits are not accurately stored and truncated. Therefore, for MP and CNN models, we have estimated that the accuracy is maintained without writing correct data because they have a common thing that they have fewer layers, fewer parameters and a simple model structure. In contrast, MN model, which has a lot of parameters and a complex

structure, seems to be required the more precision. The LSB-side bits require to be written precisely for complex network types.

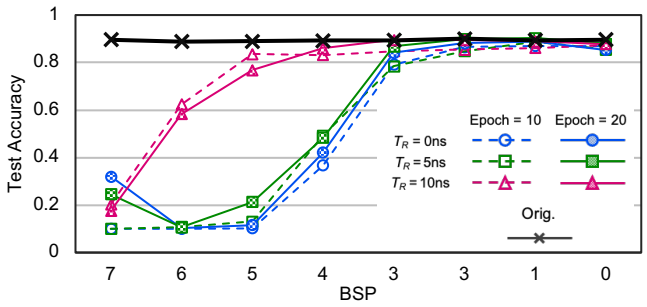
The dataset type is related to the range of the test accuracy. Meanwhile, by comparing with MP and CNN, MNIST is a simple dataset, but CIFAR-10 is a complex dataset. Hence, the dataset type may not affect the trend of BSP. However, it should be noted that the image size of datasets affects BSP since the large image size leads to large parameters and a complex model. In addition, it also should be noted that FP32 may be required for more complex network types and the large size datasets.



(a) Multi-layer Perceptron (MP)



(b) CNN



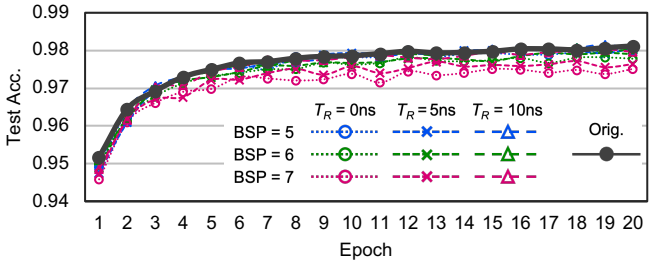
(c) MobileNetV2 (MN)

Fig. 6 BSP vs. test accuracy for each network

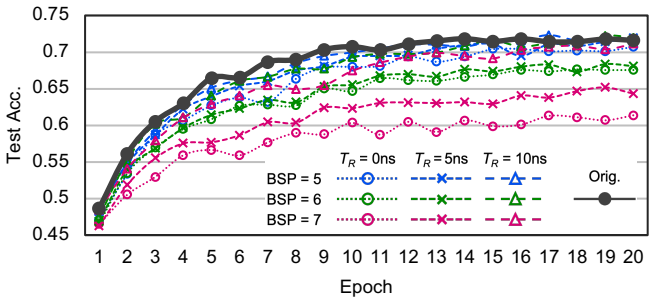
B. *Epoch vs. Test Accuracy*

Fig. 7 shows the test accuracy of each epoch. In MP and CNN, the accuracy has improved with progression of epochs and has been converged. In addition, the accuracy is affected by BSP and T_R and decreases from the original (i.e. writing all bits with 20ns) in particular when BSP gets closer to the MSB and T_R gets closer to 0. In contrast, the accuracy of MN when BSP = 4 and T_R = 0ns or 5ns moves up and down and does not converge. Complex models require the carefully settings of BSP and the write time.

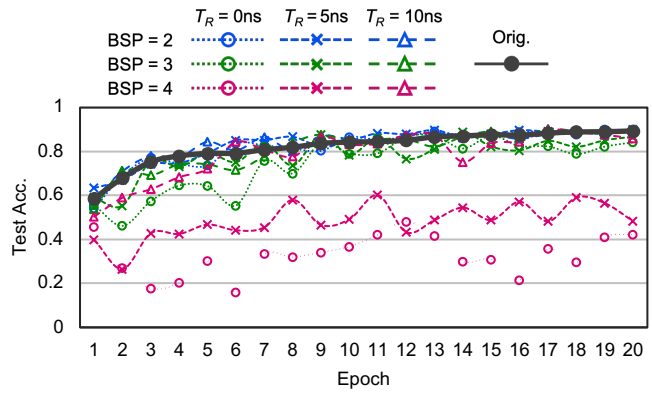
We tried the simulation to investigate whether toggling on/off of the approximate storing by epochs could further improve energy efficiency. However, it did not lead to improvements in the write energy.



(a) Multi-layer Perceptron (MP)



(b) CNN



(c) MobileNetV2 (MN)

Fig. 7 Epoch vs. test accuracy for each network

C. *Energy Dissipation*

TABLE III shows BSP and the write time to minimize the write energy in BF16 and Fig. 8. shows the total write energy when using BF16. The described energy is normalized to the case with no approximation. Accuracy Constraints is defined the difference of the test accuracy from no approximation.

Across all benchmarks, the proposed approach achieves total energy benefits ranging from 9%-38% for virtually no loss (< 0.5%) in the test accuracy. When the accuracy

constraints are relaxed to < 2.5% and < 7.5%, the energy benefits further increase to 13%-44% and 19%-44% respectively. The proposed approach achieves 21-86% write energy reduction in only the fraction part for virtually no loss (< 0.5%).

The majority of settings for energy minimization are T_R = 0ns. Meanwhile, to achieve accuracy constrains < 0.5% in MN, we require to set BSP = 2 and T_R = 5ns. We found that some low-precision bit writing is necessary for complex models.

TABLE III BSP and Write Time for Energy Minimization

Type	**MP**			**CNN**			**MN**		
Acc. Const.*	**<0.5%**	**<2.5%**	**<7.5%**	**<0.5%**	**<2.5%**	**<7.5%**	**<0.5%**	**<2.5%**	**<7.5%**
BSP	6	7	7	5	6	7	2	2	3
T_R (ns)	0	0	0	0	0	0	5	0	0

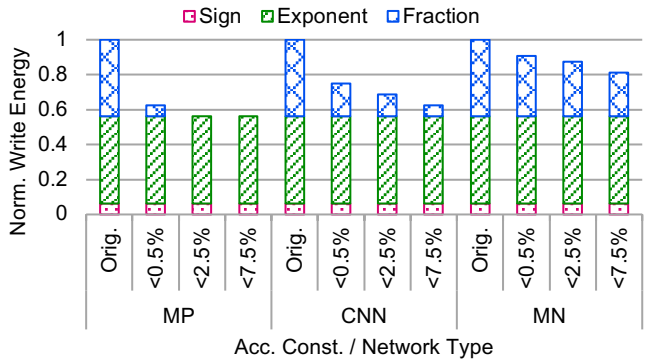


Fig. 8 Minimum Write Energy on BF16

VII. Conclusions and Future Works

For on-chip DNN learning, MTJ-based NVM systems are a promising component on future edge devices. However, MTJ-based NVM has a problem that the write energy is high. Therefore, we proposed an energy efficient approximate storing approach to reduce the write energy of NVM. We used the concept of precision scaling for the floating-point fraction part and applied it when writing weights and biases to NVM in the DNN learning tasks. We clarify that a lot of write energy can be reduced maintaining the test accuracy by applying the appropriate settings to each network type. As the future work, we will make an actual chip implementation of this system. In addition, we will investigate the advantage of preparing more than 2 BSPs, which could lead to further improvement in energy dissipation.

Acknowledgements

This work was supported by SIT Research Center for Green Innovation.

This work was also supported by VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration with SYNOPSYS Corporation.

References

[1] J. Lee, et al., "LNPU: A 25.3TFLOPS/W Sparse Deep-Neural-Network Learning Processor with Fine-Grained Mixed Precision of FP8-FP16", ISSCC 2019, pp. 142-144, Feb. 2019.

[2] J. Park, et al., "A 65nm 236.5nJ/Classification Neuromorphic Processor with 7.5% Energy Overhead On-Chip Learning Using Direct Spike-Only Feedback", ISSCC 2019, pp. 140-142, Feb. 2019.

[3] Z. Yuan, et al., "A 65nm 24.7μJ/Frame 12.3mW Activation-Similarity-Aware Convolutional Neural Network Video Processor Using Hybrid Precision, Inter-Frame Data Reuse and Mixed-Bit-Width Difference-Frame Data Codec", ISSCC 2020, pp. 232-234, Feb. 2020.

[4] L. Torres, et al., "Trends on the application of emerging nonvolatile memory to processors and programmable devices", ISCAS 2013, pp. 101-104, Aug. 2013.

[5] D. Chabi, et al., "Ultra low power magnetic flip-flop based on checkpointing/power gating and self-enable mechanisms", IEEE Trans. on Circ. and Sys., Vol. 61, No. 6, pp.1755-1765, Jun. 2014.

[6] C. Chen, et al., "Exploiting approximate computing for deep learning acceleration," DATE 2018, pp. 821-826, Mar. 2018.

[7] M. Masadeh, et al., "Using Machine Learning for Quality Configurable Approximate Computing," DATE 2019, pp. 1575-1578, May 2019.

[8] M. A. Breuer, "Multi-media applications and imprecise computation", 8th Euromicro Conference on Digital System Design (DSD'05), pp. 2-7, Sep. 2005.

[9] K. Cho, et al., "eDRAM-based Tiered-Reliability Memory with applications to low-power frame buffers", ISLPED 2014, pp. 333-338, Aug. 2014.

[10] N. Burgess, et al., "Bfloat16 Processing for Neural Networks", ARITH 2019, pp. 88-91, Jun. 2019.

[11] G. Henry, et al., "Leveraging the bfloat16 Artificial Intelligence Datatype For Higher-Precision Computations", ARITH 2019, pp. 69-76, Jun. 2019.

[12] A. Ranjan, et al., "Approximate storage for energy efficient spintronic memories", DAC 2015, June 2015.

[13] Y. Ono, et al., "Energy Efficient Approximate Storing of Image Data for MTJ Based Non-volatile Memory", NVMSA 2020, Aug. 2020.

[14] K. Usami, et al., "Energy Efficient Write Verify and Retry Scheme for MTJ Based Flip-Flop and Application", NVMSA 2018, pp. 91-98, Aug. 2018.

[15] K. Lee, et al., "1Gbit High Density Embedded STT-MRAM in 28nm FDSOI Technology", IEDM 2019, pp. 2.2.1-2.2.4, Dec. 2019.

[16] Y. Zhang, et al., "Electrical modeling of stochastic spin transfer torque writing in magnetic tunnel junctions for memory and logic applications", IEEE Trans. Magn., vol. 49, no. 7, pp. 4375–4378, Jul. 2013.

[17] M. Natsui, et al., "An FPGA-Accelerated Fully Nonvolatile Microcontroller Unit for Sensor-Node Applications in 40nm CMOS/MTJ-Hybrid Technology Achieving 47.14μW Operation at 200MHz", ISSCC 2019, pp. 202-204, Feb. 2019.

[18] Y. Chih, et al., "A 22nm 32Mb Embedded STT-MRAM with 10ns Read Speed, 1M Cycle Write Endurance, 10 Years Retention at 150°C and High Immunity to Magnetic Field Interference", ISSCC 2020, pp. 222-224, Feb. 2020.

[19] M. Sandler, et al., "MobileNetV2: Inverted Residuals and Linear Bottlenecks", CVPR 2018, pp. 4510-4520, Jun. 2018.