Approximate Programming Design for Enhancing Energy, Endurance and Performance of Neural Network Training on NVM-based Systems

Chien-Chung Ho[1,2,4], Wei-Chen Wang[3,6], Te-Hao Hsu[1,8], Zhi-Duan Jiang[1,5], and Yung-Chun Li[3,7]
[1] Department of Computer Science and Information Engineering, National Chung Cheng University
[2] Department of Computer Science and Information Engineering, National Cheng Kung University
[3] Macronix Emerging System Lab, Macronix International Co., Ltd.
{[4]ccho, [5]u07410078}@cs.ccu.edu.tw, {[6]raymondwang, [7]monixslee}@mxic.com.tw, [8]xude32@gmail.com

***Abstract*—Recently, it is found non-volatile memories (NVMs) offer opportunities for mitigating issues of neural network training on DRAM-based systems by taking advantage of its near-zero leakage power and high scalability properties. However, it brings the new challenges on energy consumption, lifetime and performance degradation caused by the massive weight/bias updates performed during training phases. To tackle these issues, this work proposes an approximate write-once memory (WOM) code method with considering the characteristics of weight updates and error tolerability of NNs. In particular, the proposed method aims to effectively reduce the number of writes on NVMs. The experimental results demonstrate that great enhancement on energy consumption, endurance and write performance can be simultaneously achieved without sacrificing the inference accuracy.**

I. Introduction

In recent years, neural networks (NNs) reveal a significant impact in many different application domains, especially in the computer vision and machine learning applications, such as object detection and image recognition. Before adopting neural networks to every corner in the world for the inference, vendors and developers have to first train neural network models with abundant training data. Nevertheless, the large leakage power, insufficient memory density and scaling difficulty issues restrict the development of the conventional DRAM-based systems. NVM-based systems then grab people's attention due to their large memory density and near-zero leakage power [17], [19]. However, NVM has inherent drawbacks, such as the higher program energy, worse endurance and longer write latency. These problems should be resolved before putting the NVM-based systems/devices in practice so as to obtain a feasible solution for neural networks. Although the large-density NVM with near-zero leakage power is suitable for training NNs, how to maintain the comparable energy consumption, endurance and performance is still challenging. This motivates this work to explore solutions about how to integrate the characteristic of NNs with management of NVMs for resolving the aforementioned issues. Especially, we are highly interested in investigating the approach which exploits the error-tolerable and approximate computing properties of NNs.

A neural network consists of multiple layers, and each layer usually contains a large number of artificial neurons. In general, the usage of neural networks can be categorized into two phases: the *training phase* and the *inference phase*. In the training phase, neural networks execute the forward propagation, which computes the input data through a series of linear (e.g., multiply-accumulate) and non-linear (e.g., pooling) computations in the hidden layers, and eventually obtains the output results. NNs then update weights and biases in the backward propagation so as to let the output results better fit the true data and acquire higher accuracy. Once the neural network is well-trained, it would be applied to edge devices for the inference tasks. However, NN computation usually involves massive weights and/or intermediate data which lead to excessive data movement and computation overheads. To minimize the NN model size and enhance the inference performance for inference tasks, lots of excellent works had been proposed in the past few years. Some researchers proposed the quantization method to quantize data into lower bit precision [10], [14], others proposed the pruning approaches to prune the redundant nodes and layers [18], [20], and still others presented the compact neural network architectures [1], [6]. Except for the inference phase enhancement, lots of efforts had been paid on the training phase improvement. For instance, to enhance the training performance with higher training speed, some researchers proposed to parallelize computing processes of the NN training over single or multiple GPUs [15], [24]. Besides, some researchers proposed the optimization approaches to realize on-device training/learning [7], [8], and some proposed to virtualize the memory usage of NNs so as to fit NNs into the DRAM capacity [13]. It is worth noting that the proposed approaches in this work are orthogonal to most of the aforementioned NN optimization techniques.

It has been established that the conventional DRAM-based system encounters slow scaling, large leakage power and insufficient capacity issues. These issues become severe when the training neural networks are applied on DRAM-based devices since the training tasks will generate much more needs on the memory capacity and performance, compared to the general inference tasks. Therefore, the non-volatile memory (NVM), especially phase change memory (PCM), soon becomes a potential solution for the systems with adopting the neural network training, mostly because of their high density and near-zero leakage power. However, PCM has larger write power, slower write speed and worse endurance, compared to DRAM. To address these issues, researchers have paid great efforts to improve performance and reliability for PCM-based devices/systems. For instance, some researchers focused on proposing the various programming techniques for enhancing the performance of PCM, such as Flip-N-Write [16], 2Stage-Write [9], Partial-SET [3], PreSET [12] and WOM-SET [22]. In recent years, researchers further started to leverage NVM and PCM for neural network applications, such as the hot-spot suppression approach [4] on NVM-based systems. However, it could be adopted only for the inference phase. Meanwhile, some researchers proposed innovative designs for training neural networks on NVM-based or PCM-based systems [5], [21]. Nonetheless, the proposed approach in [5] would degrade the training and write performance, while the proposed approach in [21] only optimized the write performance and endurance. That is, none of the proposed approaches aimed to simultaneously improve the energy consumption, endurance and write performance for training NNs on NVM-based devices or systems.

This research is motivated by the scalability problems of running neural networks over conventional DRAM-based systems. Although NVM provides nice features, such as high density and near-zero leakage power it also leads to the severe energy, lifetime, and performance problems. The objective of this work is to achieve the energy, endurance and performance enhancement of NVM-based devices while the comparable NN accuracy is still maintained. We aim at exploring how to characterize and analyze neural networks and proposing the management design with taking the observed characteristics of neural networks (e.g., error-tolerability and approximate computing properties) into considerations. We further evaluated the feasibility and capability of the proposed design through a series of experiments, and the results demonstrate that our proposed design could enhance the energy consumption by up to 61%, and improve the endurance and performance by up to 78%, and 46% respectively.

978-1-6654-2375-5/21/$31.00 ©2021 IEEE

The rest of this paper is organized as follows. Section II presents background and research motivation. Section III demonstrates our proposed approximate programming design. Section IV shows the capability of our proposed design. Section V concludes this work.

II. Background and Research Motivation

A. WOM Code Method on NVM-based System

Applying neural network training on NVM-based systems can help in diminishing the inherent issues happened on DRAM-based systems, e.g., slow scaling, large leakage power, and insufficient capacity. However, PCM-based system could incur the new issues such as asymmetric write performance, larger write energy consumption and worse endurance. Typically, the *SET* and *RESET* operations are used to programmed PCM cells from '0' to '1' and from '1' to '0' respectively, and the latency of SET operation is about 8× slower than that of RESET operation [12]. Besides, both of SET and RESET operations are usually simultaneously executed on PCM cells since a memory line contains hundred of bits, and thus the write performance of PCM is mainly hampered by the slower one which is the SET operation. To resolve the write asymmetric problem, a "PreSET" approach is proposed to proactively SET all the data bits into '1's during memory bank idle period, and then the memory controller only has to execute RESET operation during the memory write period [12]. As a result, PreSET can effectively enhance the write performance since we simply have to execute the faster RESET operation in each write period.

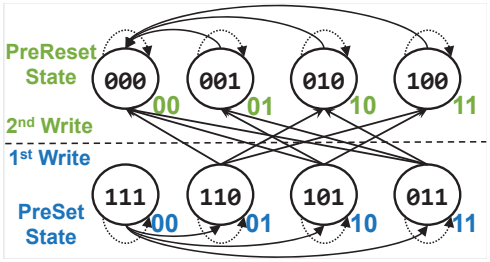


Fig. 1. The encoding rule of WOM-SET.

Although applying PreSET helps improve the write performance, it also results in generating a large number of bit flips on the programmed cells and thus incurs the energy consumption and lifetime issues. To ease the negative effects caused by applying PreSET, a Write-Once-Memory-SET (referred to as WOM-SET) was proposed to enhance not only write performance but also write energy efficiency for the PCM [22]. The WOM-SET approach adopts WOM code method to encode every two-bit data into three-bit data, and then program the three-bit encoded data on PCM cells, as shown in Figure 1. Similar to the PreSET approach, the WOM-SET also applies SET operation to let all the data bits be turned into '1's during memory bank idle period. For every two updates/writes on the two-bits data, it first follows the rule on the "1^{st} write" stage to program the three-bit data into the corresponding state for the first write/update. For the second write/update on cells which have been applied with the first write, it switch to follow the rule on the "2^{nd} write" stage to program the three-bit data into the corresponding state. Note that, if the original data of the first write and second write are same, WOM-SET will not reprogram the data and just leave the data as the "1^{st} write" encoding state. WOM-SET can be better explained with the example in Figure 2. It is found that the WOM-SET approach could decrease the number of bit flips and thus limits the energy consumption without sacrificing the write performance, compared to the basic and PreSET approaches. To sum up, WOM-SET could achieve high write performance and low energy consumption simultaneously due to the reduced bit-flip accumulation. However, WOM-SET approach could result in the space overhead since it creates one bit overhead for every two-bit data. It can result in the worse endurance problem on the PCM.

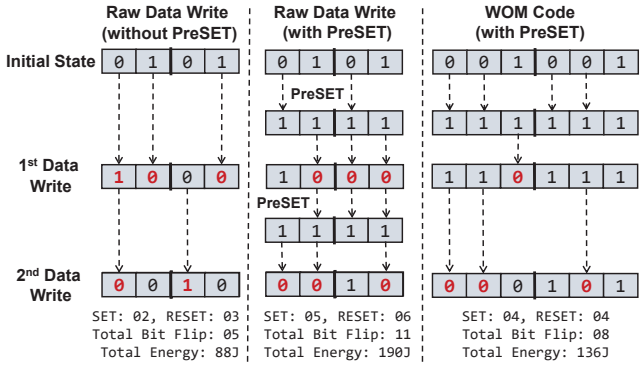


Fig. 2. Comparison of bit-flip count and energy consumption on PCM with applying the different approaches.

B. Research Motivation

Although adopting WOM-code approach could effectively improve energy consumption and lifetime of NVM, we find a totally different story when it comes to neural network training applications. A unique phenomenon of bit accumulation among various weight bits is found since training techniques usually use smaller learning rates to adequately adjust weights and biases of neural networks. Figure 3 shows the bit result[1] statistics of weights and biases when DenseNet-BC is trained on CIFAR-10 dataset, where the x-axis denotes different bits of weights and biases and the y-axis denotes the percentage bit result count. It is observed that except for the sign bit (i.e., bit 0), the most significant bits (MSBs) accumulates unevenly, while the least significant bits (LSBs) accumulates very balanced. In other words, due to the small learning rate when training neural networks, the MSBs rarely change while the LSBs change frequently.

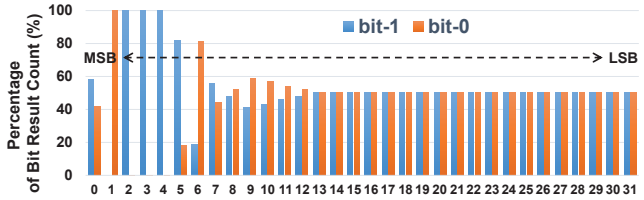


Fig. 3. Statistic of bit result of weights and biases when DenseNet-BC is trained on CIFAR-10 dataset.

The effectiveness of applying WOM code for training NNs on NVM-based system can be evaluated with the results shown in Figure 4, where the x-axis denotes 16 groups of the encoded 32-bit weights and biases while the WOM code is adopted, and the y-axis denotes the number of bit-flip count when DenseNet-BC is trained on CIFAR-10 dataset. Note that, in Figure 4, S denotes the sign bit, E_n denotes bit n of the exponent part, and M_n denotes bit n of the mantissa part. It is found that the bit-flip accumulation among different encoded bits is still unbalanced. For instance, the encoded MSBs (e.g., $E2 + E3$) barely flip, but the encoded LSBs (e.g., $M22 + M23$) flip frequently. Furthermore, the number of bit-flip accumulation among encoded LSBs are very large. It implies that the energy consumption and endurance issues of NVM-based system with adopting WOM code can be still deteriorated. To this end, we come up with a novel but challenging idea. Since neural network is well-known for its approximate computing property, is it

[1]To simplify the discussion, we assume the data are stored with IEEE-754 32-bit floating-point format when training neural networks, but the observation and our proposed approach can be extended to other data precision format with limited modification.

possible to propose an appropriately revised WOM code programming design with exploiting the approximate computing for training on NVM-based system, so as to not only improve energy consumption, endurance and write performance, but also maintain comparable neural network accuracy.

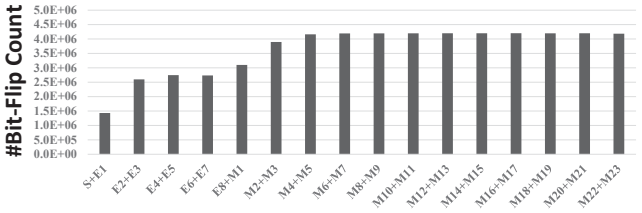


Fig. 4. Bit-flip accumulation of weights and biases encoded by WOM code.

This work is inspired by the challenges in leakage power and memory density problems on conventional DRAM-based devices when training neural networks. Even though exploiting NVM-based system can resolve the problems, it further causes worse energy consumption, endurance and write performance issues. Compared to PreSET, adopting WOM code on NVM-based system to program weights and biases of neural networks can improve write latency and reduce the number of bit-flip accumulation simultaneously; however, we observe that not only the bit-flip accumulation over different data bits is unbalanced, but also the total amount of bit-flip accumulation can be extremely large when training neural networks. Such characteristics would further significantly increase energy consumption and deteriorate endurance on NVM-based devices. The technical problems then fall on (1) how to reduce redundant bit flips when program weights and biases of neural networks with adopting WOM code, (2) how to balance the uneven bit-flip accumulation among different data bits of weights and biases, and (3) how to wisely combine the programming design of NVM with the approximate computing characteristic of neural networks without sacrificing neural network accuracy. In the following section, we shall propose to consider the properties of neural networks and non-volatile memory jointly. Particularly, we will present the approximate programming designs for neural networks on NVM-based system with adopting WOM code.

III. Approximate Programming Design for Energy, Endurance and Performance Enhancement

A. Design Overview

As we mentioned in previous sections, this work adopts PCM as the material of NVM main memory. Hereafter we refer to NVM and PCM interchangeably when there is no ambiguity. To improve the energy and lifetime for neural networks on NVM-based system, the main challenge falls on how to effectively reduce and evenly distribute the number of bit flips over all PCM cells (i.e., NVM cells). As a result, the objective of this section is to propose an approximate programming design to address the mentioned issues. To be more specific, in Section III-B, an approximate WOM code (referred to as "AppWOM") method with considering characteristics of weight and bias updates of NNs and concept of approximate computing will be presented. The proposed AppWOM will be used to (1) skilfully create more write chances for WOM encoding processes by ignoring some updates on the less important data, (2) effectively reduce the number of total bit flips over all PCM cells, and (3) cautiously maintain and balance the even bit-flip accumulation for all PCM cells. With the proposed AppWOM, the energy consumption and uneven bit-flip issues can be significantly limited and bounded, and it thus improves the energy consumption and endurance for NNs on NVM-based system. Later in Section III-C, a wear-aware ping pong policy, which integrates the management of PreSET and PreRESET operations with the proposed AppWOM, will be presented to further address how the number of bit flips and energy usage can be further improved and limited.

B. AppWOM: Approximate WOM Code Method

1) Design Concept: Lessons learned from Figure 3 and Figure 4 are twofold. First, the bit change rate on LSBs (i.e., the mantissa part of IEEE-754 floating-point format) of weights and biases is much higher than that of MSBs. This implies that LSBs of weights and biases are more vulnerable to alter, compared to that of MSBs. Second, the number of total bit flips are dominated by that of accumulated bit flips on the mantissa part even the WOM code is adopted. As a result, our primary objective is to ease and reduce the accumulation rate of bit flip on LSBs (i.e., mantissa part) so that the energy consumption and uneven bit-flip accumulation issues can be resolved.

As reported in existing work, neural networks have the error-tolerable characteristics [5], [11], [21]; that is, NNs can maintain the accuracy even data of some less important bits are deliberately ignored. In the mantissa bits in the IEEE-754 floating-point format, the former half of mantissa bits represent the larger data values, and thus they are much important bits than the latter half of mantissa bits. Such an observation inspires us to propose an approximate WOM (referred to as AppWOM) code design, which considers the importance of the different bits and integrates considerations with encoding processes of the conventional WOM code. *The main idea of AppWOM is to reserve the update chances of less important bits (i.e., the latter half of mantissa bits) for serving the oncoming updates on the more important bits (i.e., the former half of mantissa bits).* To realize this idea, bits of weights and biases are partitioned, regrouped, and stored on the NVM, as shown in Figure 5. To be more specific, the 22 mantissa bits except for the first bit are partitioned into two groups: the 11 former bits are categorized to important bit group, and remaining 11 latter bits are categorized to unimportant bit group. The first bit of both groups are grouped together, the second bit of both groups are grouped together, and so on. They are then stored on the NVM and applied with AppWOM code for the purpose of reducing and easing the bit-flip accumulation. On the other hand, since the MSBs (i.e., one sign bit, 8 exponent bits, and the first mantissa bit) are highly related to the data precision, it needs to carefully and precisely manage the updates on these bits. As a result, they are divided into groups of two, stored on the NVM, and handled with applying the conventional WOM code, so that the accuracy of NNs will not be significantly affected.

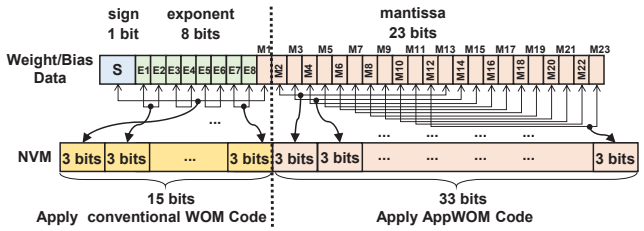
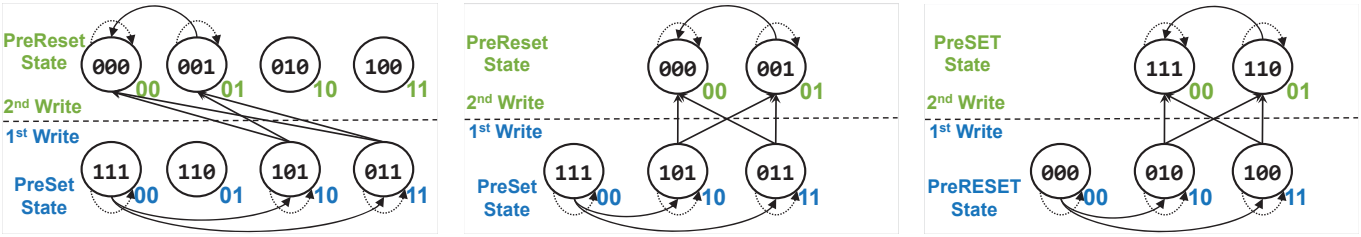


Fig. 5. How weights and bias are partitioned, grouped, stored, and managed in the proposed AppWOM design.

2) AppWOM Code Method Design: This section presents how to construct AppWOM code method with taking the importance of different bits into considerations. As shown in Figure 5, the 22 mantissa bits except for the first bit are partitioned and regrouped, e.g., mantissa bits $M2$ and $M13$ are grouped and written with the proposed AppWOM code. *Compared to the conventional WOM code method, the core idea of our proposed AppWOM is to update the encoding state for the encoded group only when the important bit or both of two bits need to be updated. In other word, the proposed AppWOM code keeps the encoding state unchanged and ignore the update request while it only needs to update the unimportant bit of encoded groups.* Following this idea, the conventional WOM code method can be revised and refined to construct the AppWOM code, and the procedure of how to construct the proposed AppWOM code is shown



(a) Remove the paths which represent the change on only the unimportant bit.

(b) AppWOM$_{10}$ is used to serve updates on PCM cells being applied with PreSET operations, i.e., all bits are '1'.

(c) AppWOM$_{01}$ is used to serve updates on PCM cells being applied with PreRESET operations, i.e., all bits are '0'.

Fig. 6. How to construct the proposed AppWOM code.

in Figure 6. The procedure is as follows: Taking the conventional WOM code shown in Figure 1 as the initial state. According to the core idea of AppWOM, if it only needs to update the unimportant bit, i.e., the latter bit, of the encoded group, the AppWOM will directly ignore the update requests and keep the encoding state in the same state. As a result, the paths which represent the change on only the unimportant bit can be removed from the original WOM code method, as shown in Figure 6(a). It thus results in the situation that some encoding states are not likely to be reached while the idea of AppWOM is applied. As a result, the proposed AppWOM code method can be derived by removing these unreachable states, as shown in Figure 6(b). Since PCM cells can be applied with PreSET or PreRESET operations, the proposed AppWOM code is further extended to two types for serving writes on PCM cells with the different initial states. The AppWOM$_{10}$ is used to serve updates on PCM cells being applied with PreSET operations, i.e., all bits are '1'. The AppWOM$_{01}$ code can be obtained by inverting the value on states of AppWOM$_{10}$ code, and it is used to serve updates on PCM cells being applied with PreRESET operations, i.e., all bits are '0', as shown in Figure 6(c).

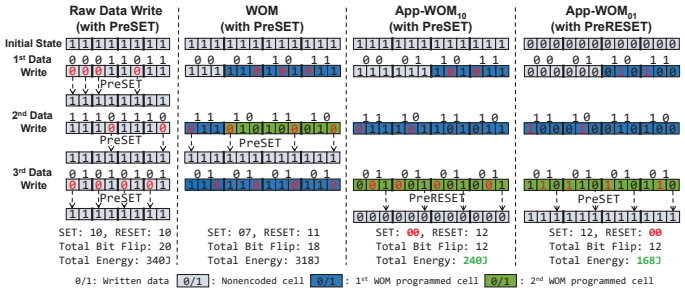


Fig. 7. An example of the proposed interleaving region allocation strategy.

How the energy consumption and the number of accumulated bit flips are improved by the AppWOM can be explained with the examples shown in Figure 7. The four programming approaches over PCM are compared, and they are raw data write with PreSET, WOM with PreSET, AppWOM$_{10}$ with PreSET, and APPWOM$_{01}$ with PreRESET respectively. Suppose they are all in the PreSET or PreRESET state initially, and each of them is used to serve 8-bits data updates for weights and biases of NNs. Let the energy to execute the PreSET and PreRESET are 14 J and 20 J respectively [2], and the following three updates to the weight/bias data are 00011011, 11101110, and 01010101. It is found that the raw data write approach suffers from the serious bit-flip accumulation and energy consumptions issues since it results the 20 bit flips in total and consumes 340 J of the energy consumption. This is because this approach needs to apply PreSET operations on PCM cells before they are programmed next time, and it thus results in the negative effects over NVM-based system. It is also observed that the bit-flip accumulation and energy consumption issues can be slightly improved while the convention WOM code is applied. Although the WOM code method can ease the bit-flip accumulation problem by executing two updates on the same allocated PCM cells, it still needs to apply PreSET operations after the two updates of the allocated PCM cells are exhausted. *The previous results reveal two facts, and they are as follows: (1) The conventional approaches can not completely turn all the bits to opposite value, and thus result in the uneven bit-flips accumulation issue. (2) Applying the PreSET operation on the changed PCM cells deteriorates the uneven bit-flips accumulation issue, and further results in the additional energy consumption.* Fortunately, these issues can be effectively resolved while the proposed AppWOM code design is applied. As shown in Figure 7, the bit-flip number of both AppWOM$_{10}$ and AppWOM$_{01}$ are 12, and the energy consumption of AppWOM$_{10}$ and AppWOM$_{01}$ are 240 J and 168 J respectively. The significant improvement on both energy consumption and bit-flip accumulation is due to the proposed AppWOM code design might have more update chances for serving the oncoming writes, compared to the conventional WOM code design. In this example, PCM cells with applying the proposed AppWOM code can be used to execute three updates since some unimportant updates are ignored while the second data write comes. In addition, different from previous approaches which always apply PreSET on the changed PCM cells, the PCM cells which remain in '1' will be applied with PreRESET operations to completely turn all the bits to opposite value. These PCM cells will then be used to serve the oncoming updates with applied the AppWOM$_{01}$ code method. As a result, the energy consumption and bit-flip accumulation issues can be effectively resolved. Note that, as astute reader might point out that some PCM cells might store data under the different encoding stage on the memory line while the proposed AppWOM is adopted. How can we identify what is the correct stage of the 3-bits read data In fact, this can be known by counting the number of '1' or '0'. For PCM cells with applying AppWOM$_{10}$ code, they are in the first and second write stage if the number of '1' on the 3-bits read data is 2 and 1 respectively. Without loss of generality, this approach can be also applied on PCM cells with applying AppWOM$_{10}$ code. Such an identification approach only needs one additional read which is neglectable while it is compared to the write latency of PCM cells.

C. Wear-aware Ping Pong Policy

This section proposes a wear-aware ping pong policy to control the flip of all PCM cells so that the proposed AppWOM$_{10}$ and AppWOM$_{01}$ codes can be smoothly applied for cells with the different initial states. In addition, we need to address the problem that WOM-related approaches could result in the additional space overhead since they creates one-bit overhead for every two-bit data. It can result in the worse endurance problem on the PCM. In our target environment, all the PCM cells will be divided into two regions. One is the PreSET region and the other is the PreRESET region. For all PCM cells of the PreSET region, they are initiated with applying the PreSET operation, and all of their initial value state are '1'. On the other hand, for all PCM cells of the PreRESET region, they are initiated with applying the PreRESET operation, and all of their initial value state are '0'. Taking PCM cells in the PreSET region as example, they are used to serving the update of weights and biases.

As depicted in Figure 8, these PCM cells will be gradually turned into the opposite value, i.e., from '1' to '0'. After being n^{th} updated by the AppWOM$_{10}$ code, some of these PCM cells can afford one or more updates but some of them can not. Since some of them had exhausted their update chances, they need to be applied PreSET or PreRESET operation for refreshing its data state. *At this moment, the wear-aware ping pong policy deliberately chooses to apply PreRESET to the PCM cells which remain in the data state '1' and forcedly turn these used PCM cells to the opposite value so as to even exhaust their bit-flip usage, as shown in left-hand side of Figure 8.* With such a wear-aware ping pong policy, the issues of the uneven bit-flip distribution and redundant bit-flip accumulation can be limited. Furthermore, those cells being applied with PreRESET can all be in the RESET state, and they thus can be used to serve the oncoming write with applying the AppWOM$_{01}$ code under the proposed wear-aware ping pong policy. The similar handling processes can be repeated, and then all the PCM cells of the PreRESET region will be turned into the opposite value, i.e., from '0' to '1', as shown in right-hand side of Figure 8. In conclusion, with the integration of the proposed AppWOM and wear-aware ping pong policy, PCM cells of both PreSET and PreRESET regions will be continuously switched. PCM cells of the same region can be gradually changed from '1' to '0', then from '0' to '1', and so on. As a result, the proposed wear-aware ping pong policy limits and eases the processes of bit-flip accumulation over all PCM cells, and it also avoids the unnecessary bit-flip accumulation and energy consumption caused by applying PreSET operations on the encoded PCM cells.

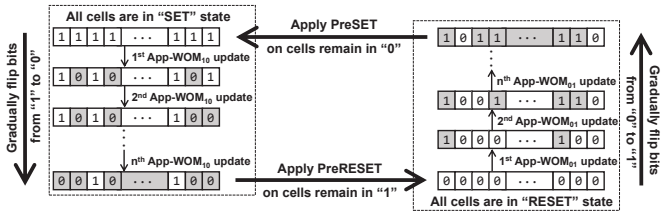


Fig. 8. Energy-efficient and wear-aware update policy.

IV. Experimental Evaluation

A. Experiment Setup

In this section, a series of experiments were conducted to evaluate the performance of our proposed approximate programming design. Our proposed AppWOM and the other compared approaches are implemented and evaluated in Caffe [23] and the NVM simulator [21]. The simulation platforms are NVIDIA GTX 1080 GPU and Intel i7-7700 CPU. The proposed AppWOM design is compared with the PCM-based system with adopting PreSET [12] (referred to as PreSET), the PCM-based system with adopting PreRESET (referred to as PreRESET), and the PCM-based system with adopting WOM-SET [22] (referred to as WOM-SET). Table I shows the parameters and setups of the investigated neural network, and Table II demonstrates the configurations of evaluated PCM-based main memory.

TABLE I. Parameters of the evaluated NN.

NN Model	Dataset	Base Learning Rate	Mini-batch Size	Number of Iteration
LeNet	MNIST	0.01	128	47K
GoogLenet	MNIST	0.001	256	58K
DenseNet-BC	CIFAR-10	0.0001	256	50K

In the experiment results, we will present the reduction of bit-flip counts on PCM, so as to completely and deeply analyze how the proposed design reduces redundant writes and balance uneven writes. Besides, the evaluated results of write energy consumption will be presented so as to understand how the proposed design improve the write energy efficiency of training neural network on NVM-based system. After that, we will present the endurance results so as to know how effectively the proposed approximate programming design enhance the PCM lifetime; here, we define the lifetime as the number of write cycle during the time period from starting training process to any of the memory cells is worn out. Lastly, we will demonstrate the performance results, in terms of the average write latency of training neural network models, so as to understand how the proposed design affect the efficiency of training neural network on NVM-based system.

TABLE II. Experiment setup of PCM-based main memory [2].

Read latency	125ns	Read energy	2pJ/bit
SET latency	1μs	SET energy	13.5pJ/bit
RESET latency	125ns	RESET energy	19.2pJ/bit
Endurance	10^8	Capacity	32GB

B. Experiment Results

Firstly, to full understand how the proposed design reduce redundant writes, Figure 9 presents the bit-flip counts of the proposed approximate programming design under the investigated neural network. The results are obtained when DenseNet-BC is trained on CIFAR-10 dataset. The y-axis denotes the number of bit-flip counts, and the x-axis denotes different data bits. Compared to the baseline approach which is shown in Figure 3, we observe that the bit-flip number of the proposed approach can be decreased by up to 14%, and the reduction of bit-flip count is especially remarkable on the least significant bits. The rationale behind such a great improvement is because with the support of approximate WOM code method and wear-aware ping pong update policy, we excessively reduce the redundant bit flips by programming insignificant data approximately. Note that, even though the bit-flip counts of few groups (e.g., $E2+E3$ and $E4+E5$) are still less, this uneven bit-flip problem could be simply resolved by adopting a general wear leveling approach.

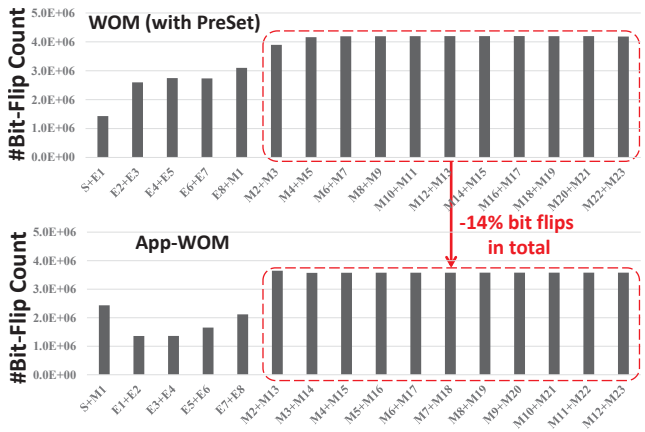
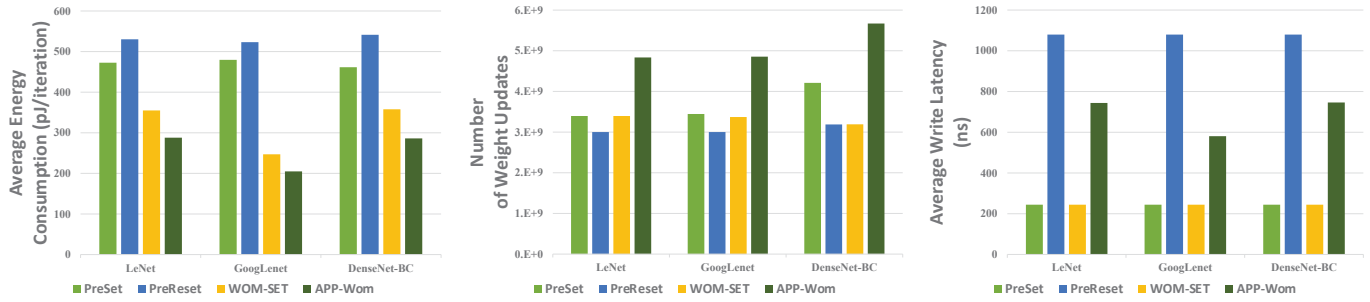


Fig. 9. Bit-flip results when DenseNet-BC is trained on CIFAR-10 dataset.

Figure 10(a) presents the energy consumption comparison of the four investigated approaches under LeNet, GoogLnet, and DenseNet-BC. The y-axis denotes the average energy consumption, and the x-axis denotes the evaluated approaches. Comparing to the PreSet, PreReset, and WOM-SET approaches, we find that the energy consumption of our proposed APP-WOM design on PCM-based system can be effectively reduced by up to 57%, 61%, and 20%, respectively. The rationale behind this is that our proposed approximate programming design on PCM-based system could significantly decrease the number of redundant writes, and thus the energy consumption is also reduced due to the less amount of programmed data.

Figure 10(b) demonstrates the endurance comparison of various approaches under the investigated neural networks, where the y-axis denotes the lifetime of PCM in terms of number of weight updates, and the x-axis denotes the four approaches. Comparing to



(a) Energy consumption results. (b) Endurance results. (c) Write performance results.

Fig. 10. Experiment results.

the PreSet, PreReset, and WOM-SET approaches, it is observed that the endurance results of PCM with adopting APP-WOM and wear-aware ping pong update policy are improved by up to 42%, 78%, and 78% respectively. This is because of the balance of uneven writes and reduction of redundant writes.

Figure 10(c) shows the performance comparison of the four investigated approaches under the investigated neural network, where the y-axis denotes the average write latency of training neural networks and the x-axis denotes the evaluated approaches. Comparing to the PreReset approach, it is observed that our proposed approximate programming design can effectively get 31% – 46% reduction of the average write latency. It is because our proposed design wisely exploits the PreSET and preRESET operations, and could achieve the decent write performance when every write request comes.

V. Conclusion

The NVM-based system seems to provide great solutions, but it also arises energy consumption, endurance and write performance issues. Even though adopting WOM code method on NVM-based systems could help in improving the performance and endurance, it further results in NVM-unfriendly features, such as redundant and uneven writes. To resolve these problems, this work proposes an approximate programming design to enable training neural networks on NVM-based devices. Specifically, an AppWOM method and a wear-aware ping pong update policy are proposed to skilfully create more write chances for WOM encoding processes by ignoring some updates on the less important data. In addition, the proposed design effectively reduces the number of total bit flips and cautiously maintains the even bit-flip accumulation for all PCM cells. The experiment results are very encouraging. The experimental results show that our proposed design could enhance the energy consumption by up to 61%, and improve the endurance and performance by up to 78%, and 46% respectively, with the comparable validation accuracy of neural networks. For the future work, we will evaluate our proposed approximate programming design on larger datasets and more complex neural network models to further verify the effectiveness of the proposed design.

References

[1] A. G. H. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*, abs/1704.04861, 2017.

[2] B. C. L. et al. Architecting Phase Change Memory As a Scalable Dram Alternative. *SIGARCH Comput. Archit. News*, Jun 2009.

[3] B. L. et al. Partial-SET: Write Speedup of PCM Main Memory. In *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '14, pages 53:1–53:4, 2014.

[4] C. W. et al. Hot-Spot Suppression for Resource-Constrained Image Recognition Devices With Nonvolatile Memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2567–2577, Nov 2018.

[5] F. M. et al. Power- and Endurance-Aware Neural Network Training in NVM-Based Platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2709–2719, Nov 2018.

[6] F. N. I. et al. SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and <1MB Model Size. *CoRR*, abs/1602.07360, 2016.

[7] H. C. et al. TinyTL: Reduce Memory, Not Parameters for Efficient On-Device Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 11285–11297, 2020.

[8] J. L. et al. MCUNet: Tiny Deep Learning on IoT Devices. *CoRR*, abs/2007.10319, 2020.

[9] J. Y. et al. Accelerating Write by Exploiting PCM Asymmetries. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 282–293, Feb 2013.

[10] M. C. et al. BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations. *CoRR*, abs/1511.00363, 2015.

[11] M. D. et al. On-chip Deep Neural Network Storage with Multi-level eNVM. In *Proceedings of the 55th Annual Design Automation Conference*, page 169, 2018.

[12] M. K. Q. et al. PreSET: Improving Performance of Phase Change Memories by Exploiting Asymmetry in Write Times. *SIGARCH Comput. Archit. News*, 40(3):380–391, jun 2012.

[13] M. R. et al. vDNN: Virtualized Deep Neural Networks for Scalable, Memory-efficient Neural Network Design. In *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-49, pages 18:1–18:13, 2016.

[14] M. R. et al. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. *CoRR*, abs/1603.05279, 2016.

[15] O. Y. et al. Multi-GPU Training of ConvNets. *CoRR*, abs/1312.5853, 2013.

[16] S. C. et al. Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance. In *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 347–357, Dec 2009.

[17] S. C. et al. Efficient GPU NVRAM Persistence with Helper Warps. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[18] S. H. et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv preprint arXiv:1510.00149*, 2015.

[19] W. K. et al. Nvwal: Exploiting nvram in write-ahead logging. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, page 385–398, 2016.

[20] W. W. et al. Learning Structured Sparsity in Deep Neural Networks. In *Advances in Neural Information Processing Systems 29*, pages 2074–2082. 2016.

[21] W. W. et al. Achieving Lossless Accuracy with Lossy Programming for Efficient Neural-Network Training on NVM-Based Systems. *ACM Trans. Embed. Comput. Syst.*, 18(5s):68:1–68:22, Oct. 2019.

[22] X. Z. et al. WoM-SET: Low Power Proactive-SET-based PCM Write Using WoM Code. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*, ISLPED '13, pages 217–222, 2013.

[23] Y. J. et al. Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, 2014.

[24] A. Krizhevsky. One Weird Trick for Parallelizing Convolutional Neural Networks. *CoRR*, abs/1404.5997, 2014.