The 10th IEEE Non-Volatile Memory Systems and Applications Symposium, NVMSA 2021

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

Chien-Chung Ho[1,2], **Wei-Chen Wang[3]**, Te-Hao Hsu[1], Zhi-Duan Jiang[1], and Yung-Chun Li[3]

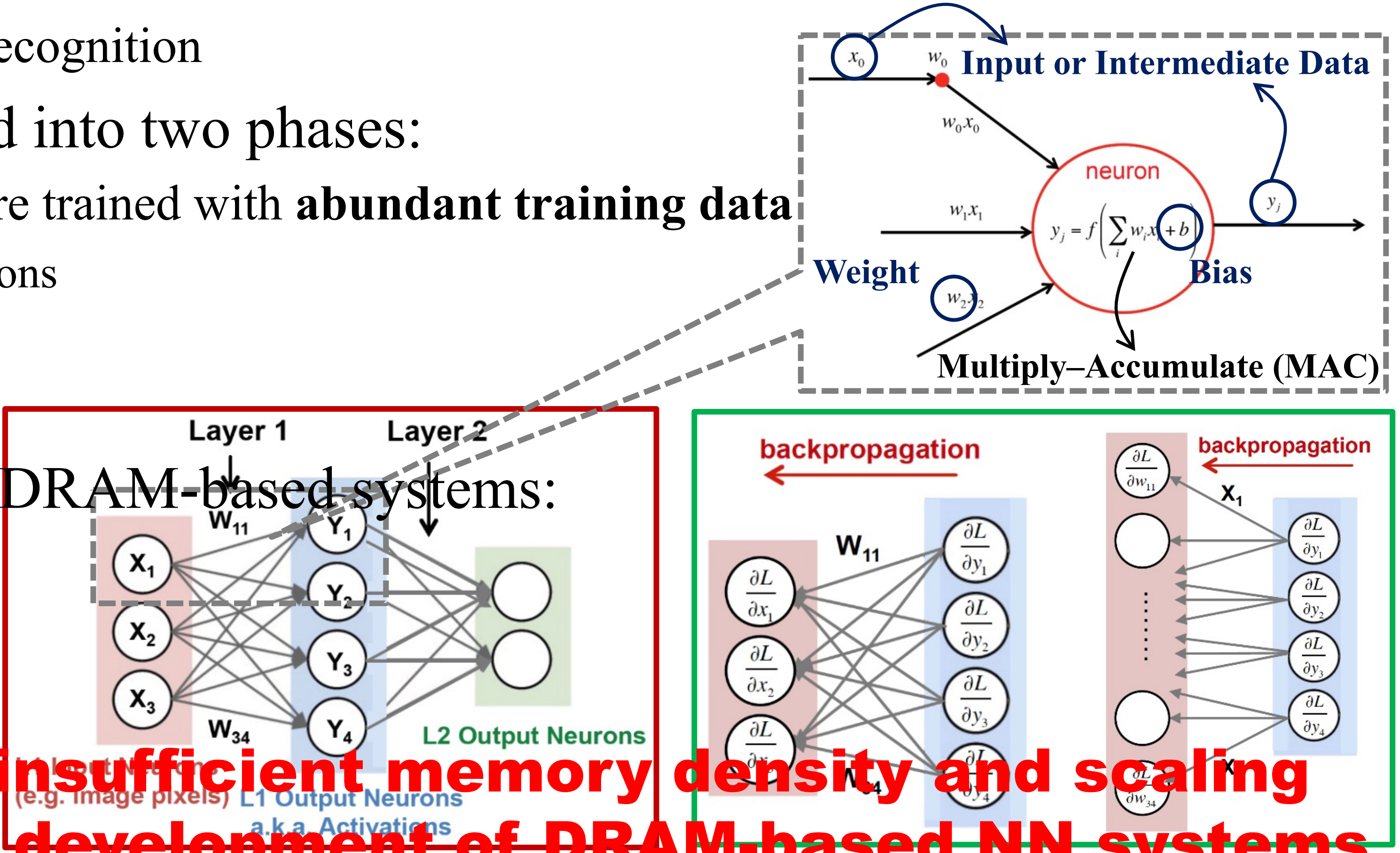
[1] National Chung Cheng University
[2] National Cheng Kung University
[3] Macronix International Co., Ltd.

Outline

- ***Introduction***
  - ***Background and Motivation***
- AppWOM: Approximate WOM Code Method
- Performance and Experiment
- Conclusion

Neural Network Trends

- NNs reveal a significant impact in many different applications/domains
  - E.g., object detection and image recognition
- The usage of NN can be divided into two phases:
  - Training phase: weight and bias are trained with **abundant training data**
    - Forward and backward propagations
  - Inference phase:
    - Forward propagation only
- Limitation of training NN over DRAM-based systems:
  - Low density
  - High unit cost
  - Significant leakage power



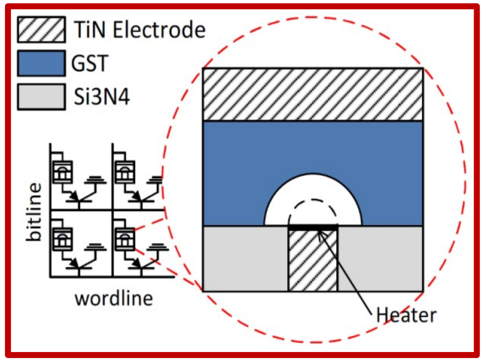
Forward Propagation

Backward Propagation

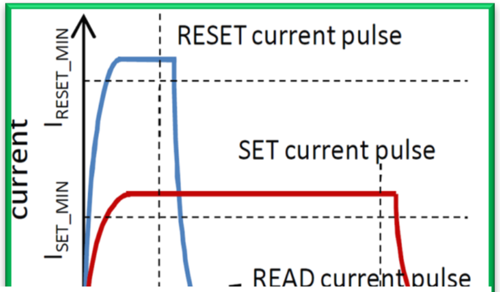
The large leakage power, insufficient memory density and scaling difficulty issues restrict the development of DRAM-based NN systems

Potential of NVM-based Training Solutions

- To address issues of DRAM-based NN trainings, NVM-based systems, e.g., phase change memory (PCM), gradually grab people's attention due to their good properties
  - Large memory density
  - Near-zero leakage power
  - Short read latency
- However, PCM has some inherent drawbacks, compared to DRAM
  - Higher program energy
  - Worse endurance
  - Longer write latency



A PCM Array

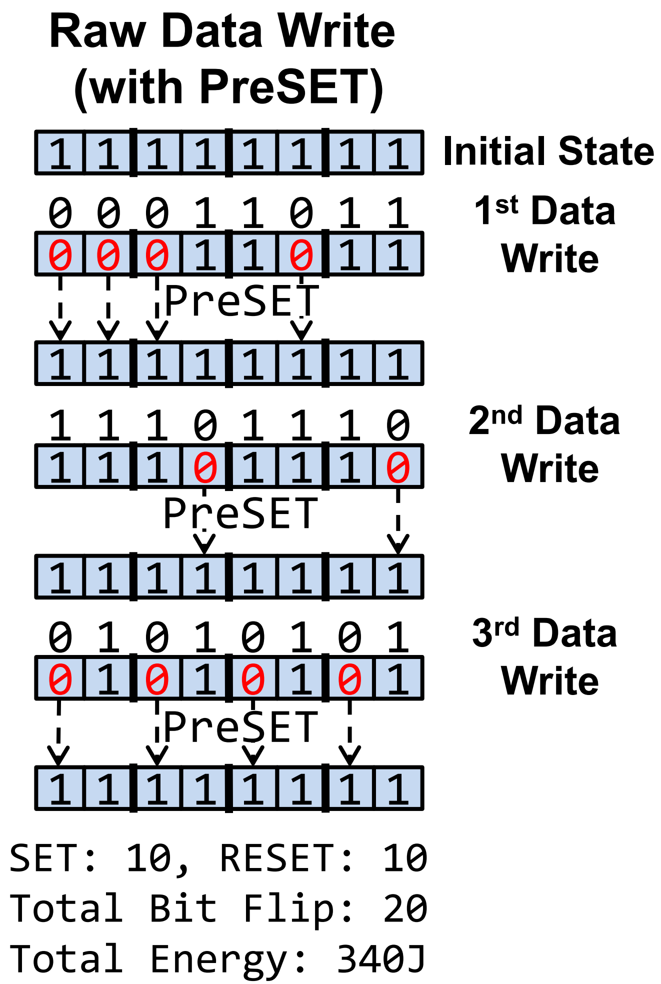


Thus, this work aims at Enhancing Energy, Endurance and Performance of Neural Network Training on NVM-based Systems

The Asymmetric Write Issue

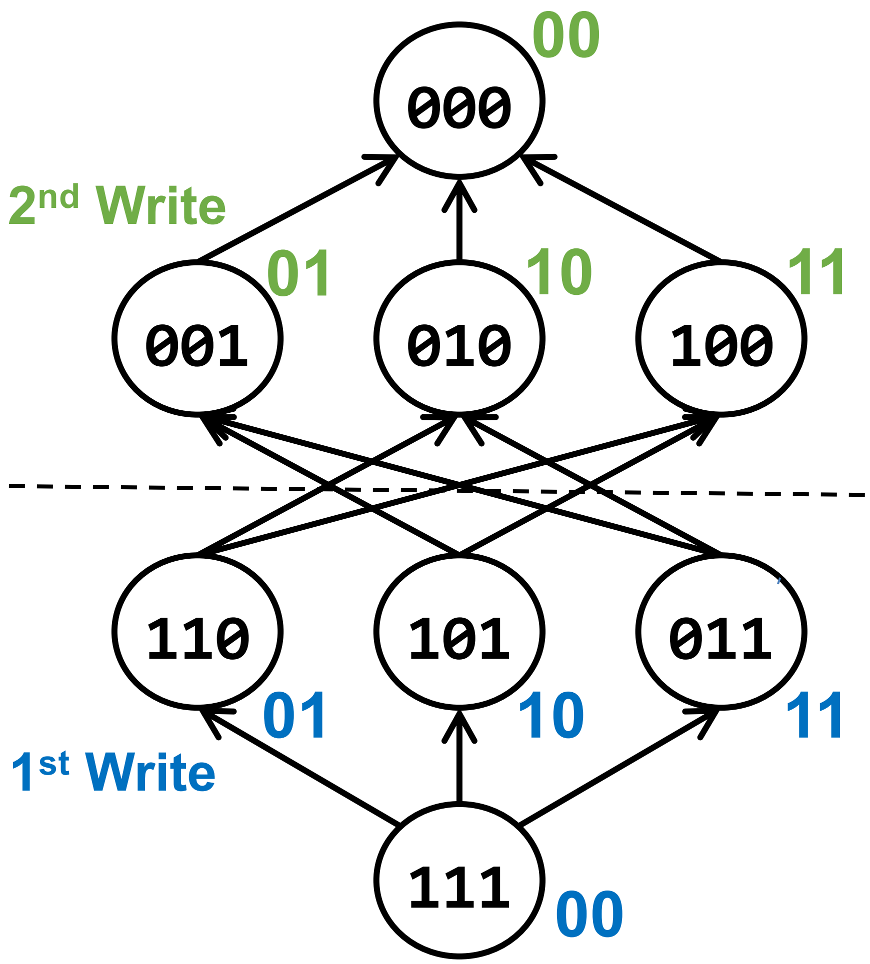
How PCM Issues are handled?

- The **write asymmetric issue** is due to the two write operations
  - RESET program PCM cells from '1' to '0'
    - Shorter write latency but more energy consumption
  - SET programs PCM cells from '0' to '1'
    - Longer write latency but less energy consumption
- ***"PreSET"*** proposes to proactively invoke SET all the data bits into '1's during memory bank idle period
  - It only executes RESET operation during the memory write period
  - It effectively enhances the write performance
- PreSET however generates **a large number of bit flips** on PCM cells
  - It thus incurs the energy consumption and lifetime issues



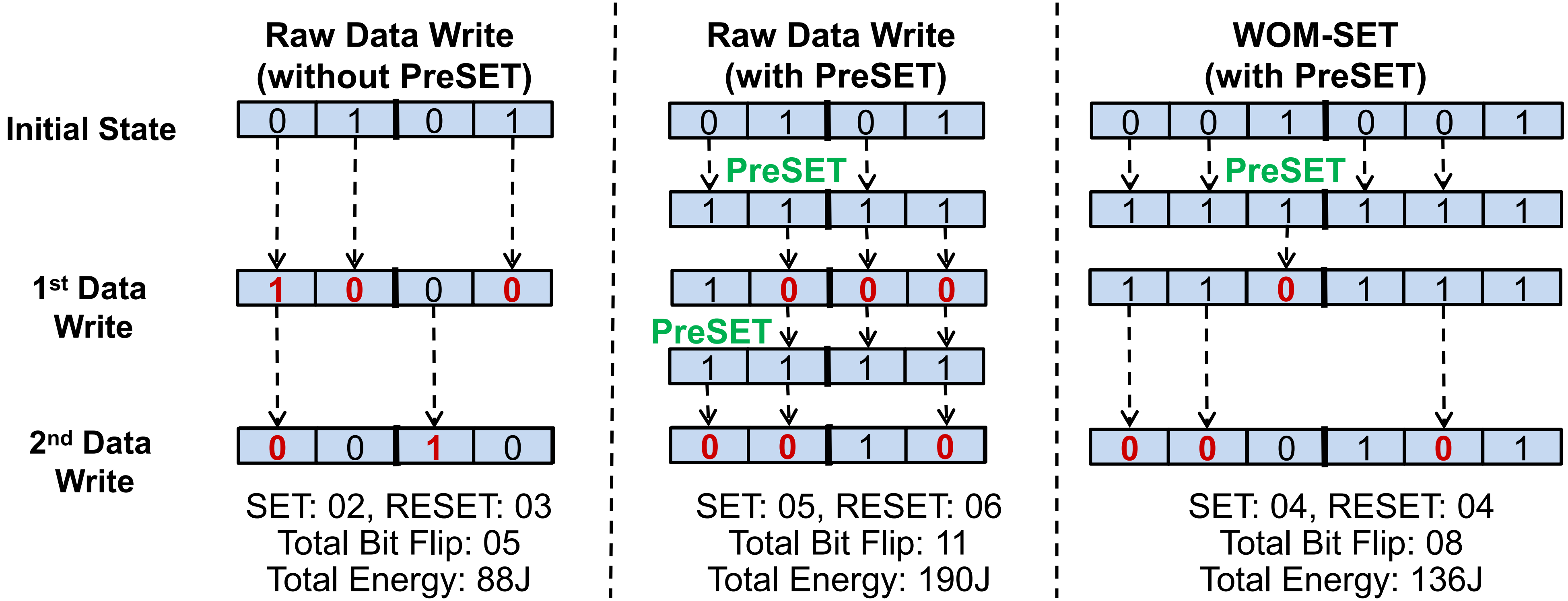
Write-Once Memory SET (WOM-SET)

- Write-once memory (WOM) code
  - A "<*v*>^{t}/*n* **WOM-code**" is a coding scheme that uses ***n* bits** to represent one of ***v* values** so that the WOM can be written in a total of ***t* times**
- WOM-SET combines both advantages of WOM code and PreSET method to improve not only write performance but also write energy efficiency
  - Encode every two-bit data into three-bit data
  - Program the three-bit encoded data on PCM cells
  - Apply PreSET to cells after the 2nd write is finished



<2>²/3 WOM-code Example

Comparison on Energy and Bit-flips



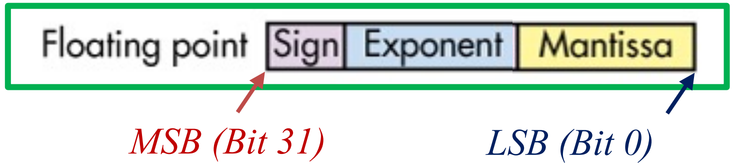
- Baseline approach provides an energy-efficient solution, but it encounters the **uneven bit-flip distribution** issue
- WOM-SET **decreases the number of bit flips** and **limits the energy consumption** without sacrificing the write performance, compared to the basic and PreSET approaches

WOM-SET could result in the space overhead and then lead to the worse endurance problem on PCM

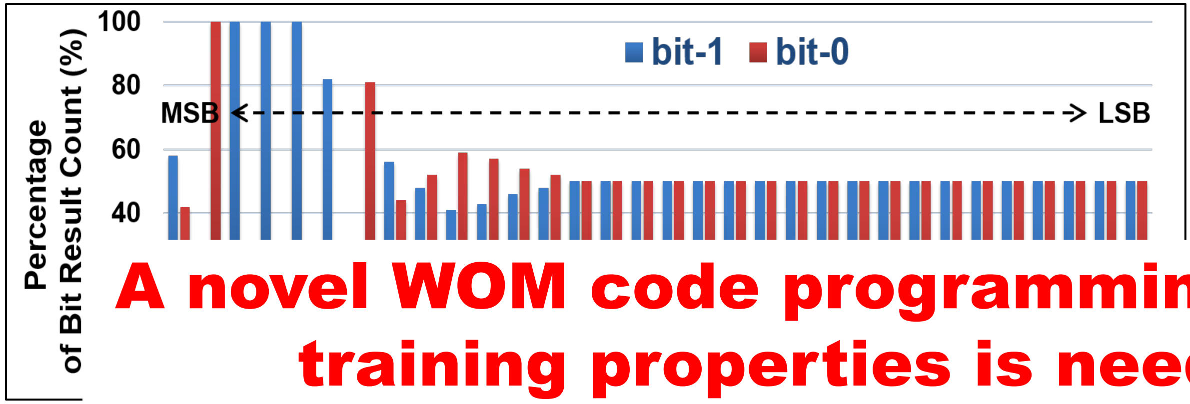
Is WOM-SET Applicable to NN Training?

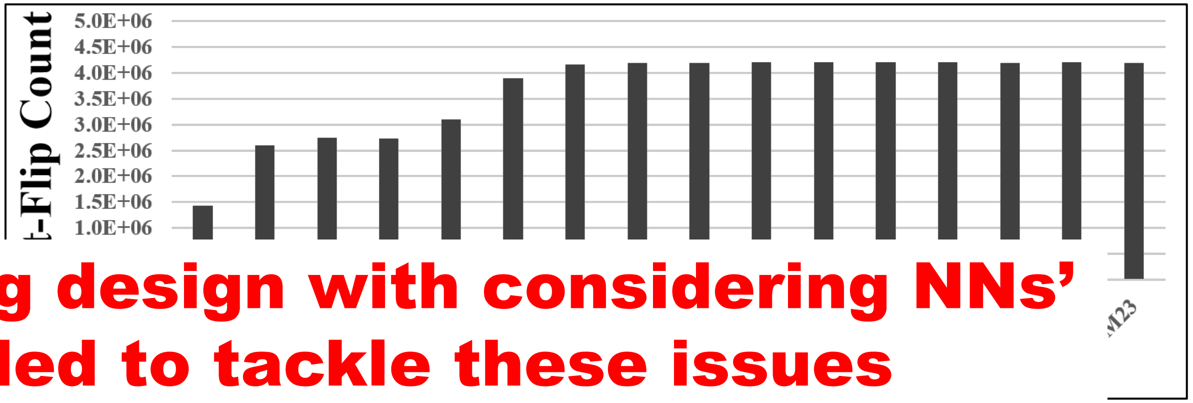
- Statistic of bit result of weights and biases when DenseNet-BC is trained on CIFAR-10 dataset, and it is found that
  - The bit-flip on MSBs accumulates **unevenly**
  - The bit-flip on LSBs accumulates **very balanced**

IEEE-754 floating point



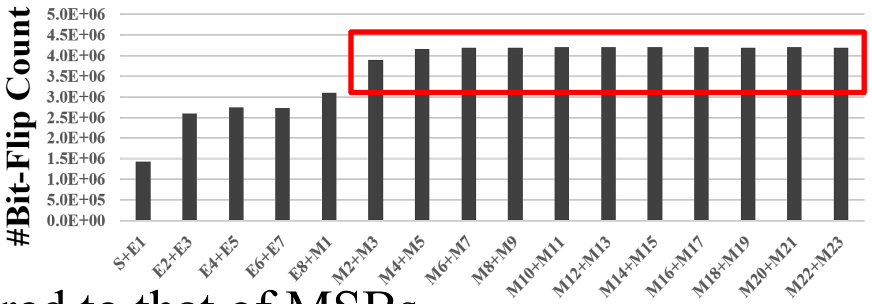
- The number of **bit-flip accumulation among encoded LSBs are very large** when applying WOM code for training NNs on NVM-based system
  - Energy consumption and endurance issues of NVM-based system with adopting WOM code can be still deteriorated





A novel WOM code programming design with considering NNs' training properties is needed to tackle these issues

Observation and Motivation



- Observations
  - The number of total bit flips is dominated by the mantissa part
  - LSBs of weights and biases are more vulnerable to alter, compared to that of MSBs

It implies the energy, endurance and performance of NVM-based training can be enhanced if accumulation rate of bit flip on LSBs can be effectively eased

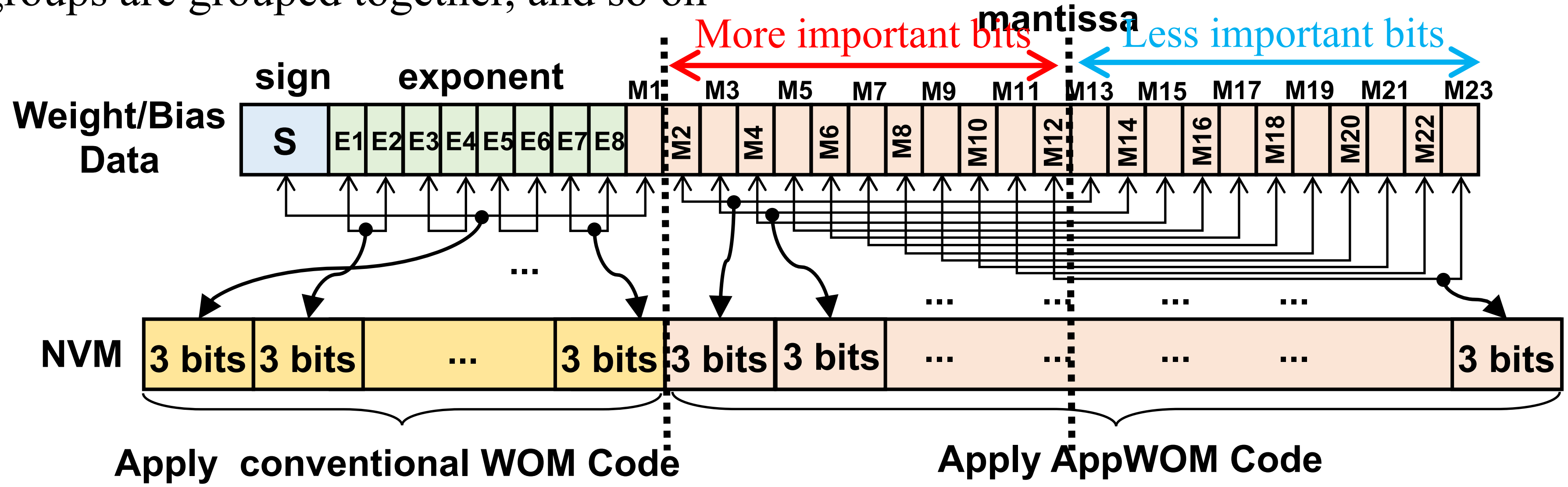
- NN is well-known for its approximate computing and fault tolerance properties
  - It uses a lot of data to train and then features to judge the result
- This motivates us to propose an **approximating WOM code** design **with exploiting approximate or fault tolerance property**, so as to
  - To skilfully create more write chances for WOM encoding processes by ignoring some updates on the less important data
  - To effectively reduce the number of total bit flips over all PCM cells
  - To cautiously maintain and balance the even bit-flip accumulation for all PCM cells

Outline

- Introduction
- AppWOM: Approximate WOM Code Method
  - Design Concept
  - AppWOM: Approximate WOM Code Method
  - Wear-aware Ping Pong Policy
- Performance and Experiment
- Conclusion

Design Concept of AppWOM

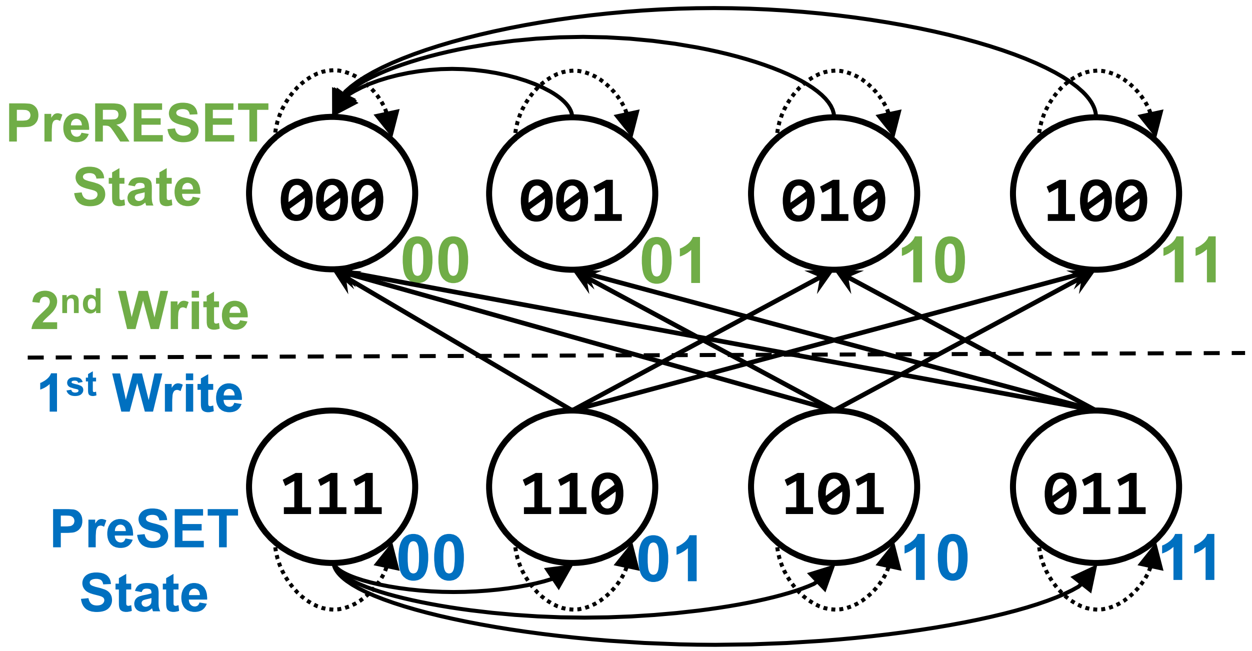
- The main idea is to *reserve the update chances of less important bits for serving the oncoming updates on the more important bits*
  - The less important bits: the latter 11 bits of mantissa
  - The more important bits: the former 11 bits of mantissa
- Bits of weights and biases are partitioned, regrouped, and stored on the NVM
  - The first bit of both groups are grouped together, the second bit of both groups are grouped together, and so on



AppWOM Code Method

- AppWOM **updates the encoding state** for the encoded group *only when the important bit or both of two bits need to be updated*
  - It **keeps the encoding state unchanged** and ignore the update request *when it only needs to update the unimportant bit of encoded groups*
  - Some path of original WOM code can be discarded according to the above rules
- AppWOM code method can be derived by the following steps:

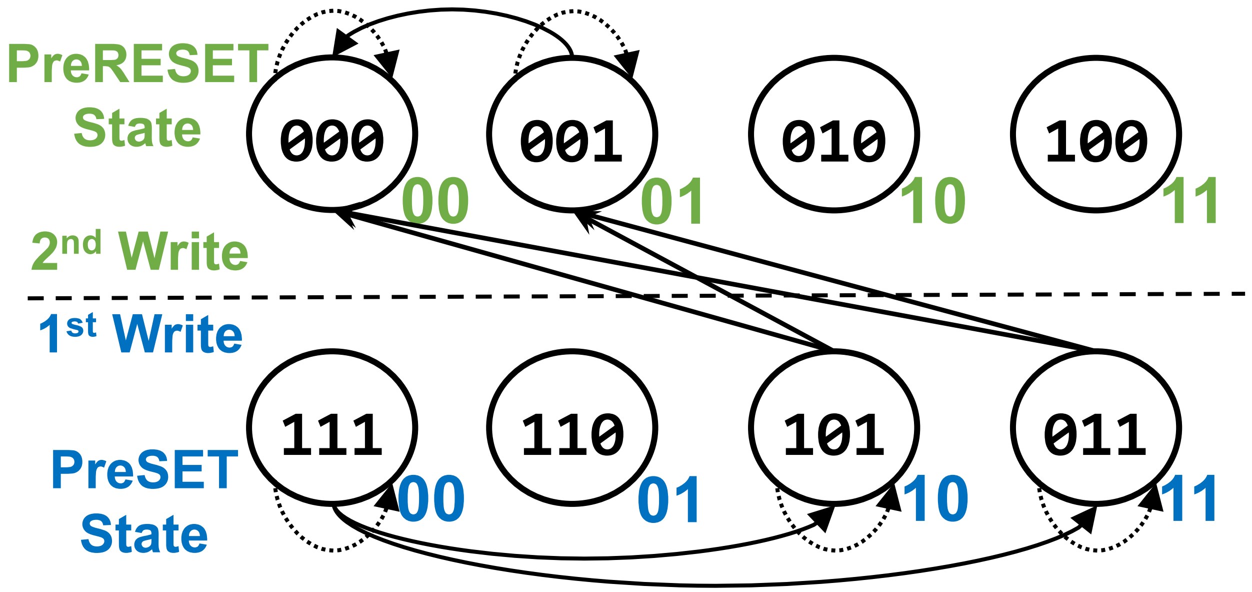
0. Original state:



AppWOM Code Method

- AppWOM **updates the encoding state** for the encoded group *only when the important bit or both of two bits need to be updated*
  - It **keeps the encoding state unchanged** and ignore the update request *when it only needs to update the unimportant bit of encoded groups*
  - Some path of original WOM code can be discarded according to the above rules
- AppWOM code method can be derived by the following steps:

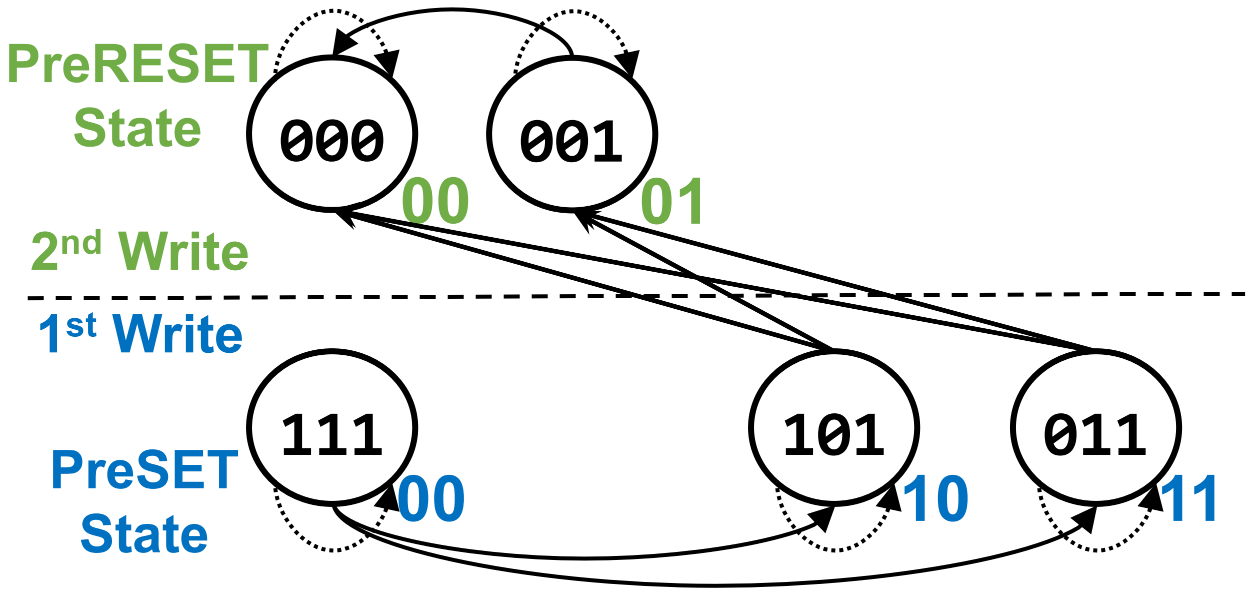
1. Remove the paths which represent the change on only the unimportant bit:



AppWOM Code Method

- AppWOM **updates the encoding state** for the encoded group *only when the important bit or both of two bits need to be updated*
  - It **keeps the encoding state unchanged** and ignore the update request *when it only needs to update the unimportant bit of encoded groups*
  - Some path of original WOM code can be discarded according to the above rules
- AppWOM code method can be derived by the following steps:

2. Remove the unused states:

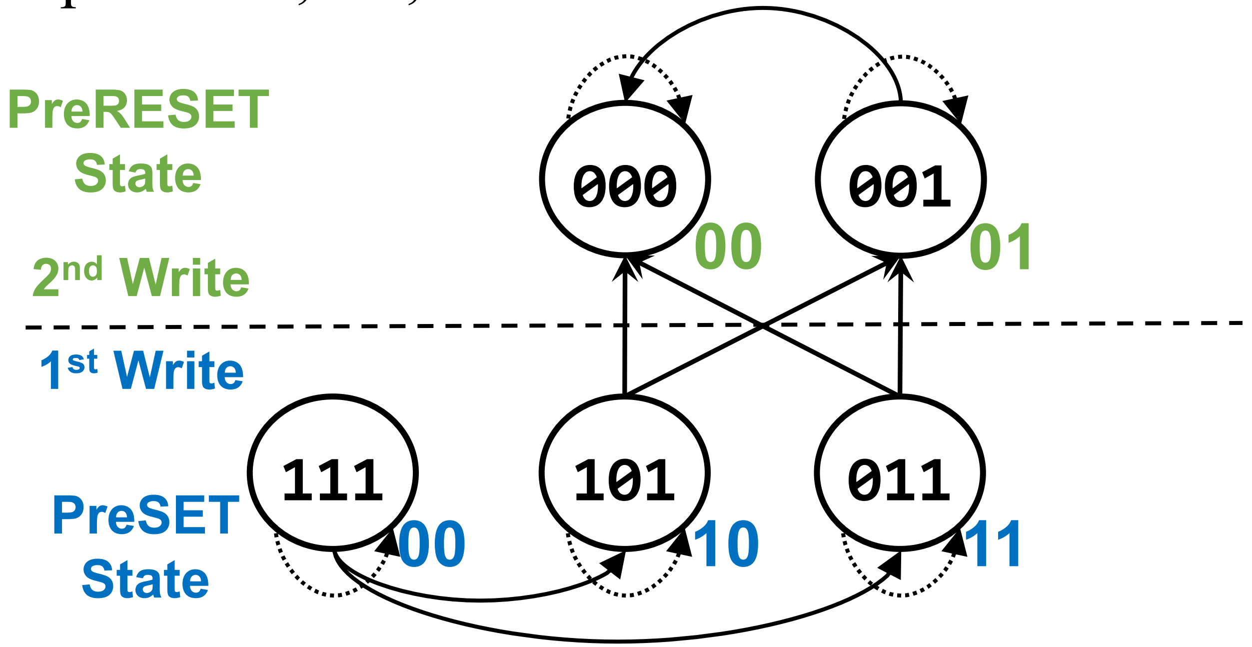


AppWOM Code Method

- AppWOM **updates the encoding state** for the encoded group *only when the important bit or both of two bits need to be updated*
  - It **keeps the encoding state unchanged** and ignore the update request *when it only needs to update the unimportant bit of encoded groups*
  - Some path of original WOM code can be discarded according to the above rules
- AppWOM code method can be derived by the following steps:

3. Rearrange states to achieve **AppWOM$_{10}$**:

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

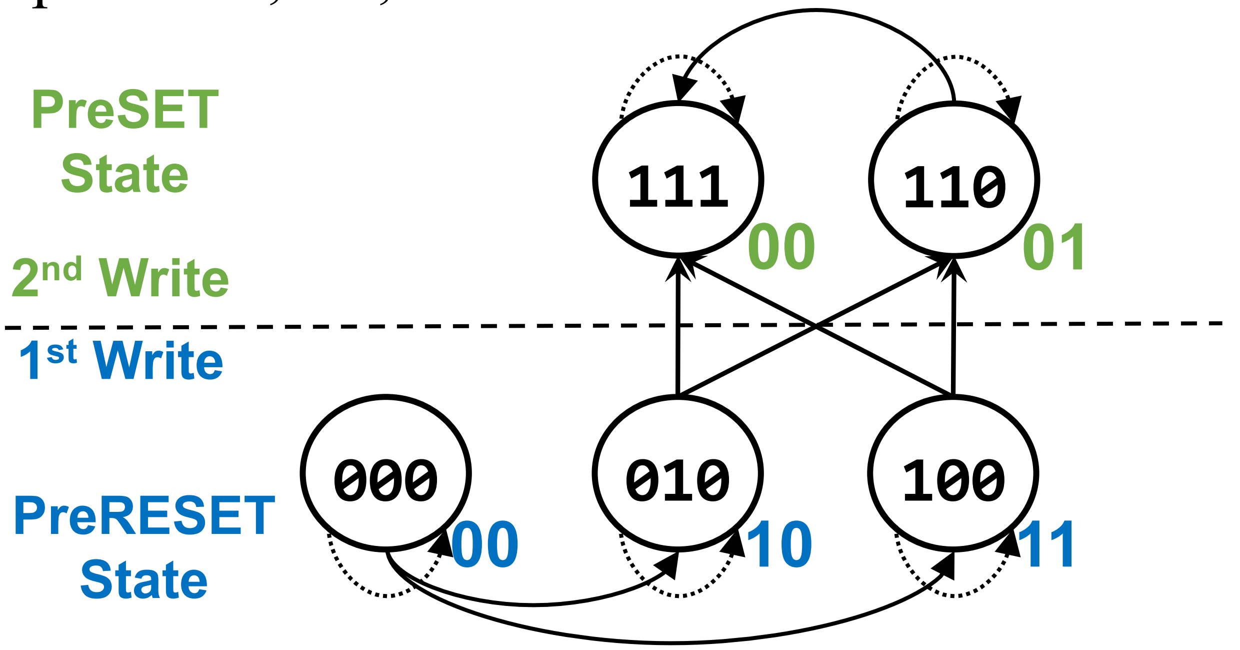


AppWOM Code Method

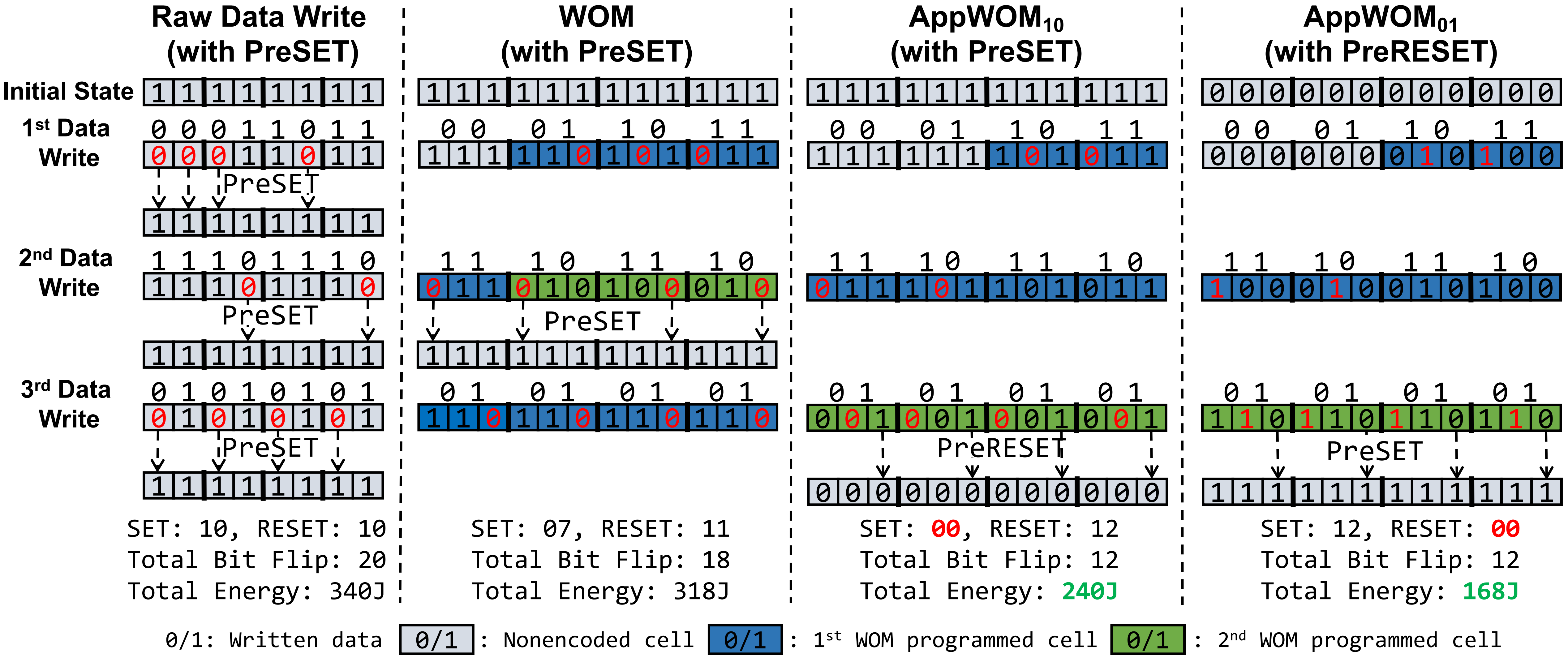
- AppWOM **updates the encoding state** for the encoded group *only when the important bit or both of two bits need to be updated*
  - It **keeps the encoding state unchanged** and ignore the update request *when it only needs to update the unimportant bit of encoded groups*
  - Some path of original WOM code can be discarded according to the above rules
- AppWOM code method can be derived by the following steps:

4. Reverse direction to achieve **AppWOM$_{01}$**:

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

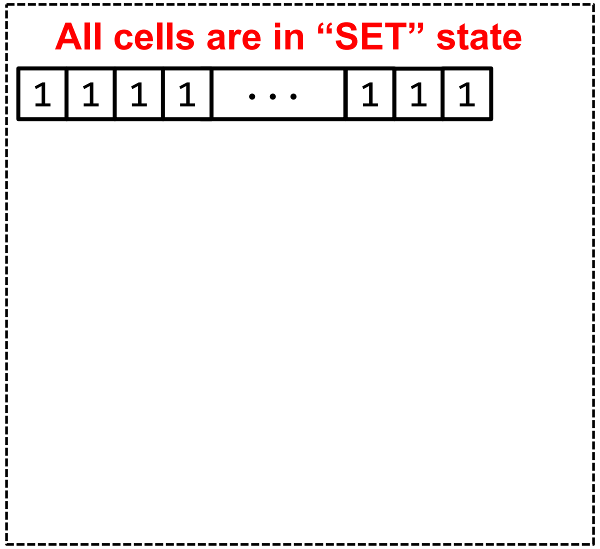


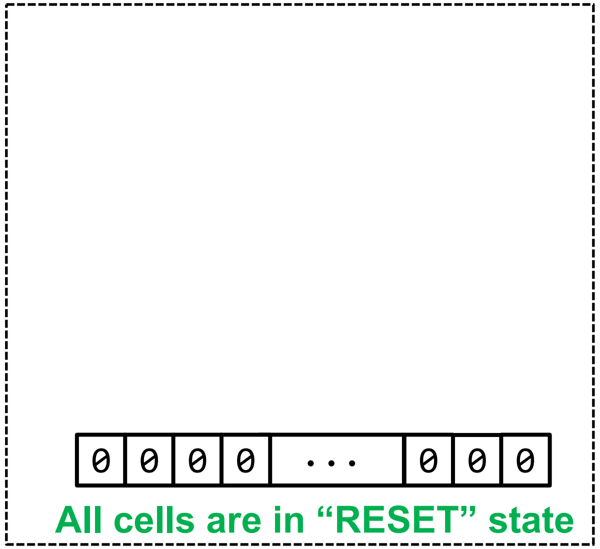
Example of Using AppWOM Code



Wear-aware Ping Pong Policy - 1

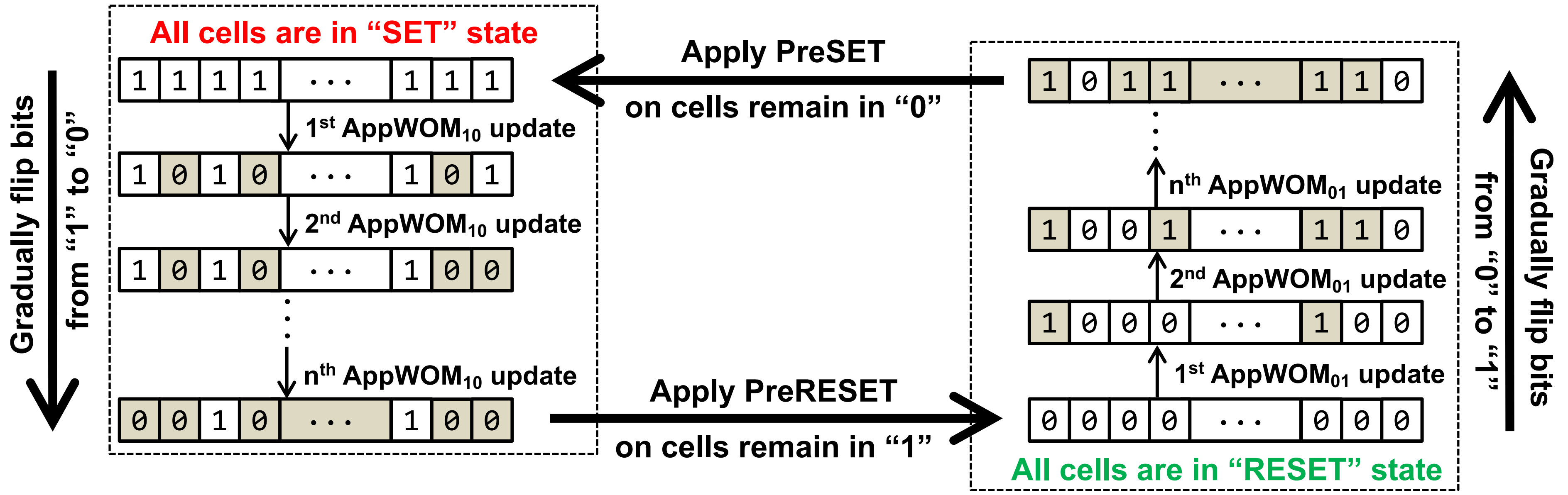
- A wear-aware ping pong policy is proposed to control the flip of all PCM cells
  - To allow the proposed AppWOM_{10} and AppWOM_{01} codes be smoothly applied on cells with the different initial states
- All the PCM cells will be divided into two regions in our target environment
  - Cells of **PreSET region** are **initiated with PreSET** operation (i.e, **value state "1"**)
  - Cells of **PreRESET region** are **initiated with PreRESET** operation (i.e, **value state "0"**)





Wear-aware Ping Pong Policy - 2

- Taking cells of **PreSET region** as an example:
- 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
  - The wear-aware ping pong policy applies 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 evenly exhaust their bit-flip usage



Outline

- Introduction
- AppWOM: Approximate WOM Code Method
- Performance and Experiment
- Conclusion

Experiment Setups

- Evaluation metrics
  - Bit-flip accumulation
  - Energy consumption
  - Lifetime (number of weight updates)
  - Performance (write latency)
- Approaches in Comparison:
  - PreSET: PCM-based system with adopting PreSET
  - PreRESET: PCM-based system with adopting PreRESET
  - WOM-SET: PCM-based system with adopting WOMSET
  - AppWOM: PCM-based system with adopting AppWOM

- Parameters of evaluated NNs

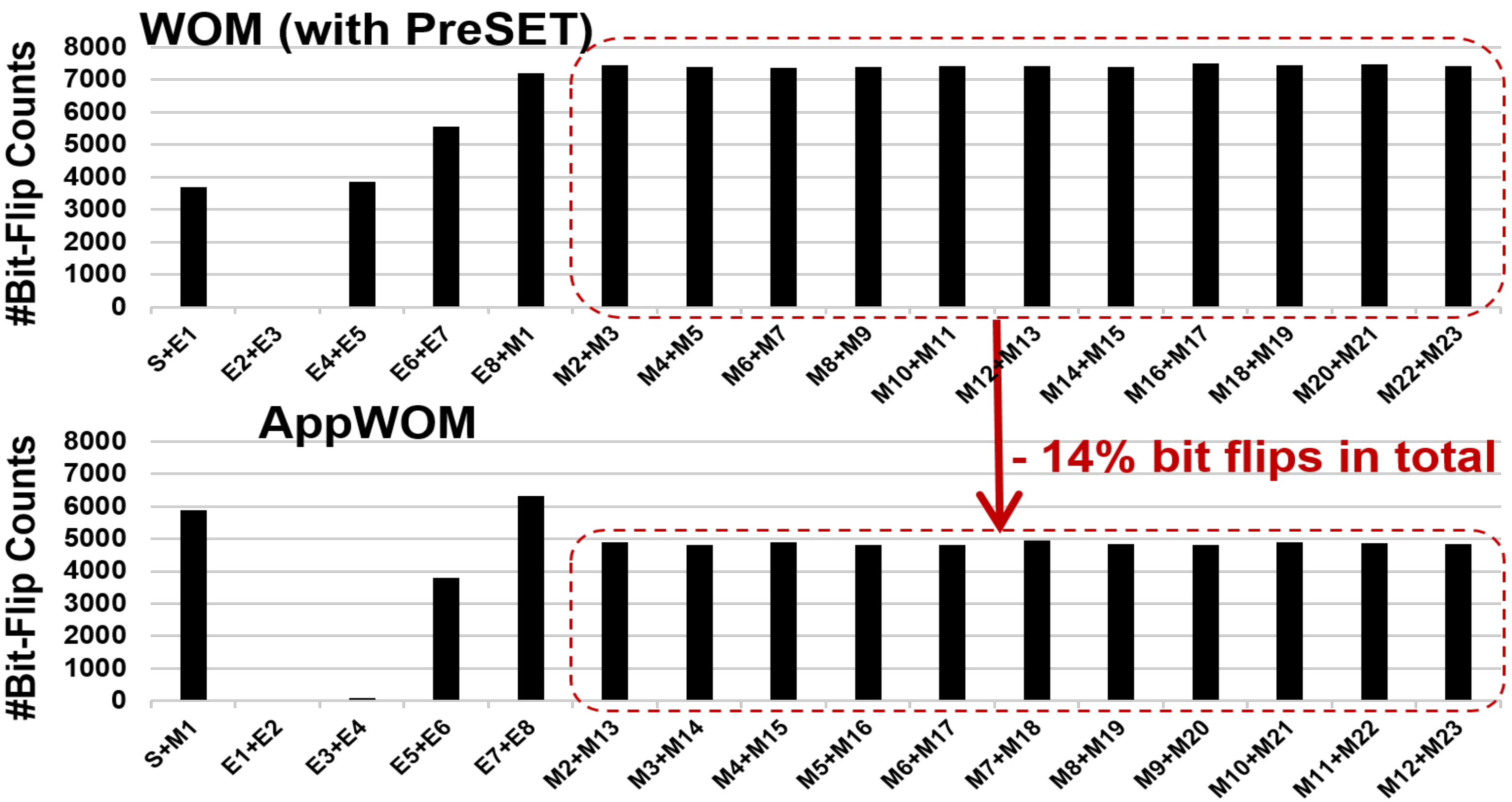
NN Model	Dataset	Base Learning Rate	Mini-batch Size	Number of Iteration
LeNet	MNIST	0.1	128	47k
GoogLeNet	MNIST	0.001	256	58k
DenseNet-BC	CIFAR-10	0.0001	256	50k

- Parameter setups of the adopted PCM

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

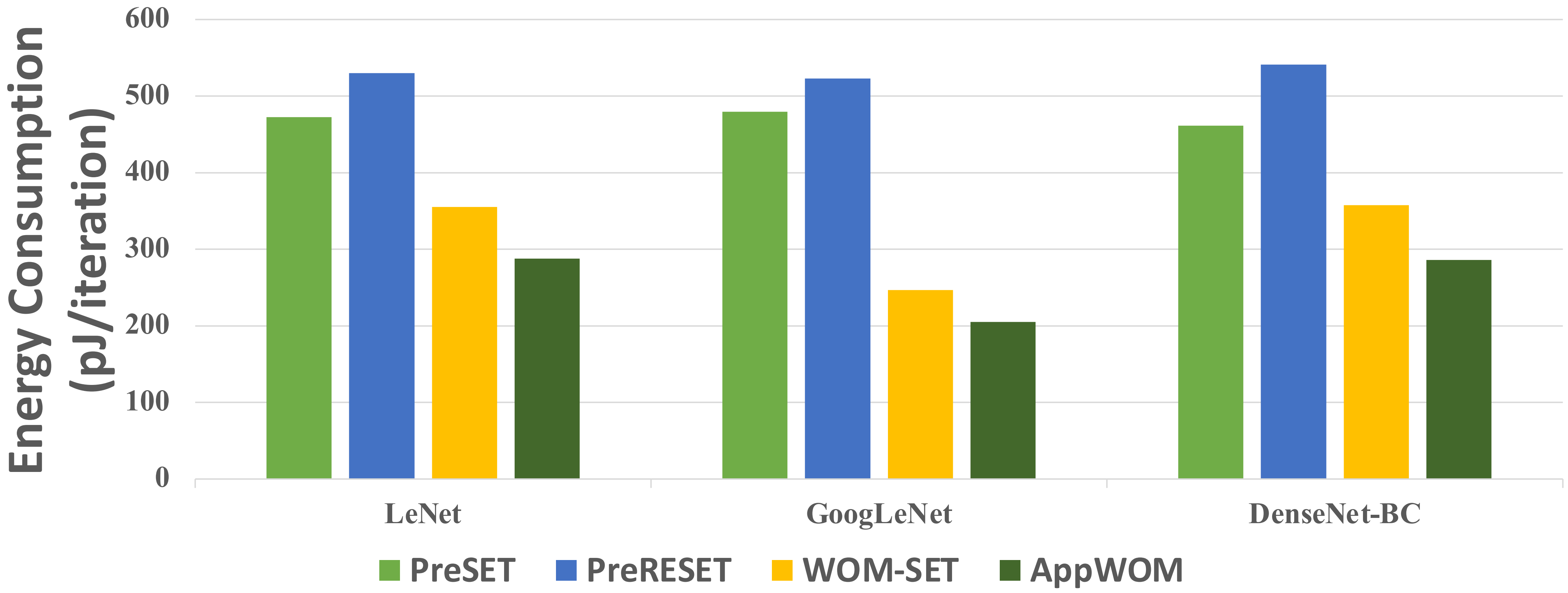
Bit-flip Accumulation Results

- The bit-flip number of the proposed approach can be decreased by up to 14%
  - The reduction of bit-flip count is especially remarkable on the least significant bits with the support of approximate WOM code method and wear-aware ping pong update policy



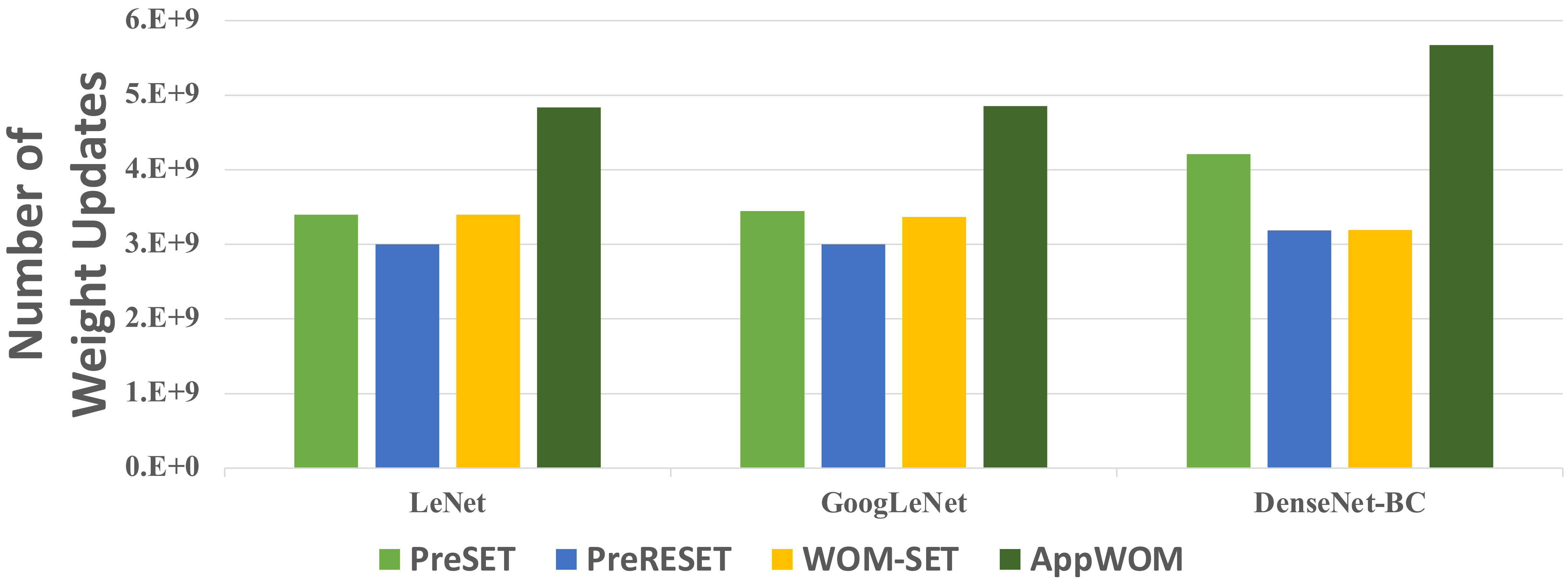
Energy Consumption Results

- Comparing to the PreSET, PreRESET, and WOM-SET approaches, the energy consumption of our proposed AppWOM design can be effectively reduced by up to 57%, 61%, and 20%, respectively



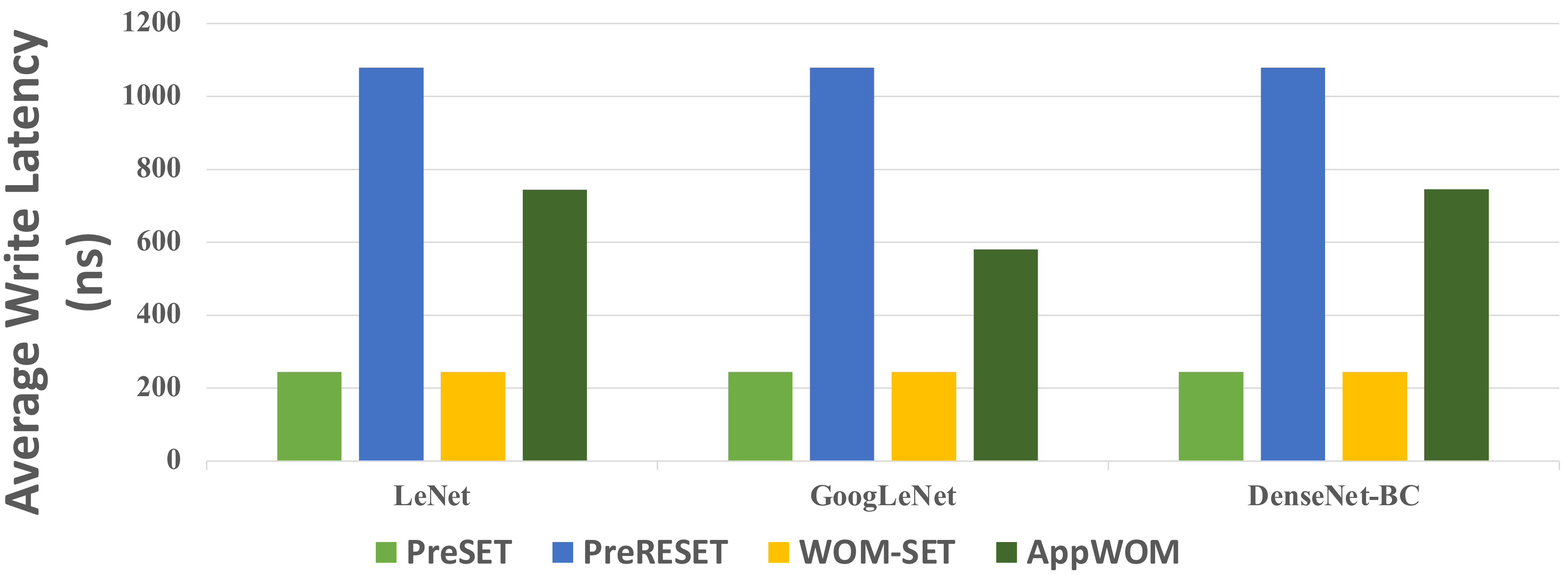
Endurance Results

- Comparing to other three approaches, the endurance results of PCM with adopting AppWOM 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.



Performance Results

- 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
  - Our proposed design wisely exploits the PreSET and PreRESET operations, and could achieve the decent write performance when every write request comes



Outline

- Introduction
- AppWOM: Approximate WOM Code Method
- Performance and Experiment
- Conclusion

Conclusion

- An **AppWOM** method and a **wear-aware ping pong update policy** are proposed to skillfully create more write chances for WOM encoding processes by **ignoring some updates on the less important data**

- 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 demonstrate that we could:
  - Improve the energy consumption by up to 61%
  - Enhance the endurance and write speed by up to 78%, and 46% respectively

- The proposed NVM-based design could enable neural network training with not only high density but also high performance

Contact Information:

Wei-Chen Wang
raymondwang@mxic.com.tw

Chien-Chung Ho
ccho@gs.ncku.edu.tw

Question & **A**nswer

Thank you for your attention