Flash Memory Array for Efficient Implementation of Deep Neural Networks

Runze Han, Yachen Xiang, Peng Huang,* Yihao Shan, Xiaoyan Liu, and Jinfeng Kang*

The advancement of artificial intelligence applications is promoted by developing deep neural networks (DNNs) with increasing sizes and putting forward higher computing power requirements of the processing devices. However, due to the process scaling of complementary metal–oxide–semiconductor technology approaches to the end and the bottleneck of data transmission in the von-Neumann architecture, traditional processing devices are increasingly challenging to meet the requirements of deeper and deeper neural networks.

In-memory computing based on nonvolatile memories has emerged as one of the most promising solutions to overcome the bottleneck of data transmission in the von-Neumann architecture. Herein, systematic implementation of the novel flash memory array-based in-memory computing paradigm for DNNs from the device level to the architecture level is presented. The methodology to construct multiplication-and-accumulation units with different structures, hardware implementation schemes of various neural networks, and the discussion of reliability are included. The results show the hardware implementations of the flash memory array-based in-memory computing paradigm for DNN own excellent characteristics such as low-cost, high computing flexibility, and high robustness. With these advantages, in-memory computing paradigms based on flash memory arrays show significant benefits to achieve high scalability and DNNs’ energy efficiency.

1. Introduction

The recent advances in artificial intelligence benefit from neural networks with a deeper depth and an increasing size of weights.[1] Therefore, to get better performance, a more considerable amount of calculation is required. Simultaneously, the development of the Internet of Things (IoT) technology is pushing computing to the edge.[3,4] Performing high-performance neural network computation in edge devices with limited area and energy requires greater computing capabilities of the processor. However, the energy efficiency and throughput of current processors based on complementary metal–oxide–semiconductor (CMOS) technology reach a plateau due to the slowing of Moore’s law.[5]

Traditional processors generally adopt the von-Neumann computing architecture, where the processing unit and the memory unit are separated.[6] In the computing process, the processing unit needs to fetch data from the memory unit and consumes a significant amount of energy and time. At least 50% of the total energy is dissipated in the data shuffling process.[7] With the neural network size increases, the processing unit needs to fetch more weight data from the memory unit during the computing process, making the separating problem severe. Tremendous efforts with advanced manufacturing technologies[7–9] and innovative computing architectures[10–12] have been proposed to overcome these problems. One approach is to reduce the physical distance between the processing unit and the memory unit. For example, using through-silicon via (TSV) technology can stack the memory and logic together to reduce the read latency and the dissipated power. However, the data shuffling process still exists in this approach and can only alleviate the data transmission challenges to some extent. To solve this problem more deeply, in-memory computing architectures based on nonvolatile memories[13,14] are proposed to avoid the data transmission process during computing.

Emerging nonvolatile memories like resistive random-access memory (RRAM), phase-change memory (PCM), magnetic random-access memory (MRAM), and ferroelectric random-access memory (FeRAM) are all two-terminal elements with the potential of fast and low-power switching.[15] In addition to the data storage function, these memories are now found to have the potential to perform computing tasks.[16–23] Due to the two-terminal structure, these memories can be used to build a high-density crossbar array, where each intersection in the array is the memory unit, corresponding to the weight in the neural network. However, to fabricate crossbar arrays with a larger size, the memory element is usually serially connected with a transistor.[24] During the switching process, the select transistor needs to provide enough current to guarantee the resistance switching. Therefore, the switching current of the memory cell limits the
scaling of the select transistor. At the same time, the researchers are now still working on to get reliable memory devices with low switching current. As a consequence, the improvement in the density of the in-memory computing array remains a major challenge.

As one of the traditional nonvolatile memories, the flash memory uses the number of electrons in the floating gate to tune the threshold voltage. If the tunneling oxide is thin enough, the charges in the floating gate can easily leak out, thus seriously deteriorating the reliability of the flash memory. Therefore, the flash memory was thought to reach its physical limit and the increase in the flash memory density (number of bits per unit area) will no longer be available. However, with the development of 3D integration technology and the multi-bit cell technologies, flash memory density is steadily promoted, increasing around 40% per year. The flash memory also has strong consolidated know-how and mature manufacturing technology. Benefiting from these factors, flash memory has regained the attention of researchers and been considered as one of the candidates to realize in-memory computing. In 2017, an analog neural network using the embedded flash memory technology is experimentally demonstrated with 1000 times reported performance improvement to the TrueNorth chip. Since then, various researches have been conducted to use flash memory array to implement neural networks ranging from deep neural networks (DNNs) to spiking neural networks. Apart from the architecture research, at the device level, floating gate transistors with extremely low operating current \( I_{on} \) (mean) \( = 2 \) nA, \( I_{eff} < 0.5 \) pA are also developed, which makes flash memory more competitive for low-power IoT applications. As the programming and erasing operation of the floating gate transistor typically consume a significant amount of time and energy, the implementation of flash memory in neural network applications are now focused on the inference process.

In this article, we will comprehensively introduce how to build neural networks based on flash memory array from device to system level. First, we will introduce how to construct a multiply-and-accumulation (MAC) unit with the floating gate transistors for single-bit or multi-bit cells. We then explain how to implement neural networks with digital inputs or analog inputs using the flash memory array. Finally, the reliability issues, including device retention and variation, are discussed.

2. MAC Unit

The MAC unit consists of multiplication and accumulation operations. The operation usually involves two vectors, the pair elements in the two vectors are multiplied, and the multiplication results are accumulated to get the final result. The operation conducted by the MAC unit lies in the critical path of the neural network systems as the essential of the convolutional layer or fully connected layer is the vector–matrix multiplication. A fast and energy-efficient MAC unit is decisive for the efficient hardware implementation of neural network computations. In this section, two types of MAC units based on flash memory array with single-bit per cell and multi-bit per cell are introduced.

2.1. Single-Bit MAC

The foundation of the current computing system is the binary operation, as the basic components of floating-point or fixed-point operations are binary operations. To realize the binary MAC operation, we first need to implement binary multiplication, which is indeed an AND logic operation. Figure 1a,b show two typical realization schemes of multiply operation based on the floating gate transistor. \( V_{ds} \) is the voltage between the gate and the source nodes, \( V_{ds} \) is the voltage between the drain and the source nodes. \( I_{ds} \) is the current flows through the drain side to the source side. Figure 1a shows the scheme with one input applied from the gate node. The operands are the threshold

![Figure 1](image-url)

**Figure 1.** Implementation schemes of multi-bit MAC operations based on flash memory array: a) With fixed \( V_{ds} \), \( I_{ds}-V_{gs} \) curves of the flash cell with different \( V_{ds} \); b) With fixed \( V_{gs} \), \( I_{ds}-V_{ds} \) curves of the flash cell with different \( V_{ds} \); c) Binary inputs with WL as one input; d) Binary inputs with SL as one input; e) Time-division multiplexing scheme; f) The flash memory array-based analog computing.
voltage of the floating gate transistor \( (A) \) and the applied voltage \( V_{gs} (B) \). For \( A \) equals to value “1” or “0”, \( V_{th} \) is high or low, denoted by \( V_{th,\text{high}} \) or \( V_{th,\text{low}} \). For \( B \) equals to value “1” or “0”, the \( V_{gs} \) is \((V_{th,\text{high}} + V_{th,\text{low}})/2 \) or 0. During the multiply operation, a fixed \( V_{ds} \) is applied to provide drive voltage. When the threshold voltage of the device is high, the applied \( V_{gs} \) cannot turn on the transistor and the current is kept low. From \( I_{ds}-V_{gs} \) curves, as shown in Figure 1a, it can be concluded that only when \( A \) and \( B \) are equal to “1”, the output current \( I_{ds} \) representing \( C \) is high and represents value “1”. Figure 1b shows another scheme with one input applied from the drain node. The difference with Figure 1a is that the operand \( B \) is denoted by the applied voltage \( V_{ds} \). During the multiply operation, \( V_{gs} \) is fixed, and the amplitude is in the middle of \( V_{th,\text{high}} \) and \( V_{th,\text{low}} \). This guarantees when \( V_{ds} \) is applied (A equal to “1”), the \( I_{ds} \) flows through the “1” cell is high, and the \( I_{ds} \) flows through the “0” cell is kept low. Based on these two realization schemes of the multiply operation, two crossbar-like architectures are designed to execute the vector–matrix multiplication computing. Figure 1c shows the MAC unit with the input vector applied through word lines (WLs). The input vector is converted to the applied voltages to the WLs connected to the gates of flash cells. The elements in the matrix are represented by the threshold voltages of flash cells. The source lines (SLs) are connected to fixed voltage sources. If the SLs are all connected, then the currents flow through cells in the array will accumulate in the SL. The SL may not withstand such a large current. The large current may also cause serious IR drop problems and make the applied voltages on flash cells at different positions of the flash memory array different. To avoid these problems, the SLs are separated. As shown in Figure 1a, the current flows through the flash cell represents the multiplication result of the pair elements of the input vector and the matrix. According to Kirchhoff’s Law, current flows through flash cells in the common bit line (BL) sum up at the end of the BL, and the summed current represents an output result. As output currents produce simultaneously, the output results can be computed parallel. Figure 1d shows the MAC unit with an input vector applied through SLs. Compared with the realization scheme shown in Figure 1c, the WLs are used to input control signals. Here, the gates of all cells can be connected together, as the currents flow through flash cells in the gate node is usually negligible, summing them up will not produce excessive current in the WL and cause a considerable IR drop problem. As shown in Figure 1b, the current flows through the flash cell represents the multiplication result of the pair elements of the input vector and the matrix. The multiplication results sum up in the BL following Kirchhoff’s law.

2.2. Multi-Bit MAC

To meet the requirements of the algorithm and improve the computing efficiency of the flash memory array-based neural network, the multi-bit MAC operation based on multi-bit flash array is essential. However, with the increase in storage states of each flash cell, the distribution of multi-bit flash becomes tighter and the overlapping among neighboring states emerges, which leads to the computing error of MAC operations.\(^{[42]}\)

To achieve precise and spaced drain–current (\( I_d \)) distributions, the dynamic \( V_{ds} \) programming method is proposed. During the programming process, the \( V_{ds} \) is kept as a constant value (9 V), the pulses with larger amplitude (\( V_{ds} \)) are applied to tune the \( I_d \) (0–30 \( \mu \)A, increment: 2 \( \mu \)A) of a flash cell to the range of \((I_{target}, I_{target} \times (1 + E_{Rd}))\), where \( I_{target} \) and \( E_{Rd} \) denote the target \( I_d \) and the programming error of the first stage of programming, respectively. Then the pulses with smaller \( V_{ds} \) are applied to achieve precisely programming. By using the method, the 4-bit storage of flash with less than a 0.5% mean programming error rate \( \text{ER} = |I_{target}/I_{target}| \) is achieved. The related work is described in ref. \([31]\) in detail.

To realize multi-bit MAC operations based on a multi-bit flash array, the time-division multiplexing scheme and the flash memory array-based analog computing are proposed, as shown in Figure 1e. In detail, for the time-division multiplexing scheme (Figure 1e), the multi-bit input vector is applied to the WL of the flash array bit-by-bit during several clock cycles. At each clock cycle, the MAC operations between the binary input vector and multi-bit storage matrix are achieved. The currents along BLs denote MAC operations, and then the result is stored. The computing of different input bits would generate a cluster of corresponding results. After the multi-bit input vector is processed by the flash array, the results are multiplied with the corresponding coefficient \( 2^{k-1} \) and added to obtain the computing result of the multi-bit MAC operations. Then the results are read out by analog-to-digital converters (ADCs). Another approach to achieve multi-bit MAC operations is the flash memory array-based analog computing. As shown in Figure 1f, different from the digital computing (WL: input) described earlier, the input vector is applied to the SL of the multi-bit flash array. Because \( I_{ds} \) is approximately proportional to the drain-to-source voltage (\( V_{ds} \)) when \( V_{ds} \) is small, the voltages applied to the SLs (0–\( V_{ds} \)) representing the multi-bit input vector that can be analog signals. The control signal applied to WLs determines whether the analog computing is executed. The result of MAC operations is defined as the currents along BLs and is obtained in one clock cycle. Approximate computing is a computation technique which trades energy and computing time with the accuracy of output.\(^{[43]}\) Therefore, the flash memory array-based analog computing is the approximate computing essentially. Although the MAC operations account for a large proportion of the operations in DNNs, benefited from the strong robustness of DNN, the impact of the computing errors of MAC operations on the performance of DNN is negligible.\(^{[44]}\) Therefore, the flash memory array-based analog computing is promising for the hardware implementation of DNN.

In summary, the flash memory array-based mixed-signal computing with the time-division multiplexing scheme is applicable for the precise computing scenario. However, the massive overhead from analog–digital/digital–analog (AD/DA) converters is introduced. For example, an 8-bit analog-to-digital converter (ADC) in ref. \([45]\) consumes 126.75 \( \mu \)W of area and 35 \( \mu \)W of power. Despite that the time-division multiplexing of ADCs is applied, more than 85% of the area and energy consumption limit the improvement of the energy efficiency of the flash memory array-based mixed-signal computing system.\(^{[46]}\) The flash memory array-based analog computing eliminates the additional overhead of AD/DA converters, logic circuits and registers of the
flash memory array-based mixed-signal computing at the computation accuracy expense.\(^\text{[10]}\)

Comparing the multi-bit MAC with the single-bit MAC, the multi-bit MAC consumes less area as a single cell can store more bits. However, as the distribution of the multi-bit storage states is tighter, the probability of errors occurred in the hardware computing results is higher than the single-bit MAC. A trade-off between the hardware cost and the required accuracy is needed when choosing the proper realization scheme of MAC.

### 3. Neural Networks Implementations

We will introduce how to implement the computation in convolutional neural networks (CNNs) or spiking CNNs with the flash memory array in the following. The rules of mapping weights to the flash memory array, data transferring between different layers, and data converting circuit designs will be explained in detail. For better versatility, here we consider the implementation of neural networks that have negative weights.

#### 3.1. Convolutional Neural Network

This section divides into two sub-sections. The first sub-section introduces the mapping method from the neural network to the flash memory array and the realization of the digital input scheme. As the mapping rule of the analog scheme is similar to the digital scheme, the second sub-section focuses on the computation of flash memory array with analog input and output.

##### 3.1.1. Digital Input

CNNs are composed of convolutional layers and fully connected layers. The convolutional layers extract the input image features, and the fully connected layers classify the features. The convolutional layer typically has multiple input channels and output channels. The convolutional results of input channels are accumulated to get the output result. Suppose the convolutional layer contains \(s\) input channels with an image size of \(m \times m\) and \(t\) output channels. Under this condition, the number of kernels is \(s \times t\). Suppose the kernel size is \(k \times k\), and the stride is 1, then the size of the output image is \((m-k+1) \times (m-k+1)\). The input image \(i\) \((1 \leq i \leq s)\) is denoted by \(X^i\), the output image \(j\) \((1 \leq j \leq t)\) is denoted by \(Y^j\), and the kernel corresponding to the input image \(i\) and output image \(j\) is denoted by \(K^{ij}\). The element in the \((u, v)\) position of the output image \(j\) is calculated by Equation (1)

\[
Y^j_{u,v} = \sum_{i=1}^{s} \sum_{p=-[q-1]}^{[q-1]} \sum_{q=-[p-1]}^{[p-1]} X^i_{p,q} \cdot K^{ij}_{k-p,u,k-q,v}
\]  

(1)

The elements in the receptive fields of the input image are multiplied with corresponding elements of the kernel and then sum up. The results corresponding to each input image are accumulated to obtain the final output result. To better illustrate the computation, a typical realization of the convolutional layer computation in the first clock is shown in Figure 2a. To implement the computation of the convolutional layer in the flash computing array, flash cells representing kernels corresponding to the same output image are arranged in the same BL. For each kernel, the elements are arranged from \(k_0\) to \(k_1\) along the vertically downward direction. The reason for doing this is that, in the convolutional computation, the kernel needs to rotate 180 degrees first and then perform multiplication and addition operations, as shown in Equation (1). As the kernel size is \(k \times k\) and the convolutional results of \(s\) input images need to be accumulated, the number of WLs is \(s \times k \times k\). Figure 2b shows how to realize the binary convolutional computing using the flash computing array. During the computation, drive voltage supply sources of a fixed amplitude are connected to the WLs. As the currents flow through different BLs represent different output images, the computing of the elements sharing the same position in each output image can be finished simultaneously. The ADCs here are used to convert the analog computing results in the form of current to the digital outputs. As ADC consumes a great amount of area and power, multiple BLs are connected to one ADC, and the converting process is executed in series.

---

Figure 2. a) Computing process to get an element in the output image in the convolutional layer at the time clock 1. The receptive fields of each input image are multiplied with corresponding kernel elements and added to get the corresponding elements of the output image; b) Hardware implementation of the convolutional layer based on the flash computing array.
The number of BLs connected to one ADC is determined by the area and power requirements of the application. At each clock, the elements in the corresponding receptive fields are computed and calculate one element in the output image. Therefore, to calculate the output image, the receptive fields need to shift \((m-k+1) \times (m-k+1)\) times.

For fully connected layers in CNN, it contains multiple input nodes and output nodes, as is shown in Figure 3. The output result \(y_j\) is calculated by Equation (2)

\[
y_j = \sum_{i=1}^{m} x_i \cdot k_{ij}
\]

(2)

The element \(k_{ij}\) is represented by the threshold voltages of the flash cell in row \(i\) and column \(j\). The current flows through the flash cell \(k_{ij}\) represents the multiplication result of \(x_i\) and \(k_{ij}\). The currents are sum up in BL\(_i\) and represent the output result \(y_j\).

### 3.1.2 Analog Input

In the hardware neural network implementation with digital input, the output of the flash computing array is current, whereas the input is in the form of voltage. Therefore, additional interface circuits, including signal converters, logic circuits, and registers\(^{[47]}\) are required. These circuits consume a tremendous amount of area and energy (\(>85\%)^{[48]}\) and limit the efficiency gains of the flash memory array-based in-memory computing system. For neural networks with a larger scale, more signal-converting circuits need to be applied to process the increasing amount of intermediate data between layers of DNN, which brings more significant overhead for the hardware implementation. To reduce the significant overhead brought by the signal-converting process between layers, a novel hardware implementation with analog input and output is proposed.

The hardware implementation of flash memory array-based analog neural network includes the system structure and circuit details are shown in Figure 4. Due to the existence of negative weights in neural network, the differential pairs are used to stored values (Figure 4a). Positive weights and negative weights are stored in odd and even rows, respectively. Therefore, the subtraction result of currents between two adjacent BLs carried out by the subtraction (SUB) circuit denotes the result of MAC operations. Figure 4b shows the detail of the SUB circuit, which is implemented based on current mirrors. The static power is adjustable by modifying the quiescent current source \(I_R\). Furthermore, according to the functions of different kinds of layers in the neural network, two types of blocks are proposed (Figure 4c,d). The outputs of the input and intermediate layers need to be processed by the nonlinear function such as the logarithmic function \(f(x) = 1/(1 + \exp(-x))\) or the rectified linear unit (ReLU) \(f(x) = \max(0, x)\). The output layer of the neural network is implemented by the activation (Act.) circuit, as shown in Figure 4e. Specifically, the output layer of the neural network is mainly used for recognition. Therefore, the result of MAC operations of the output layer is passed into the recognition (RECG) circuit to carry out the recognition task (Figure 4d). The detail of the RECG circuit is shown in Figure 4e, which is essentially the integrate-and-fire neuron\(^{[49]}\). The integrator of the RECG circuit is used to integrate the current \(i_y\), and then the output of the integral \(V_{in}\) is compared with the preset reference voltage \(V_{set}\). The outputs of RECG circuits in the output layer depend on \(i_y\), \(V_{set}\) and the integration time \(t_{set}\) (Figure 4f).

The node corresponding to the RECG circuit with the output of “1” at \(t_{set}\) is regarded as the recognition result.

### 3.2 Spiking CNN

The CNN and the spiking CNN are all trained by the same neural network. Therefore, the basic structures of CNN and the spiking CNN are the same. The main difference between the two kinds of CNNs is that the numerical values are coded using spikes (0/1) instead of voltage amplitude in the spiking CNN. In that case, the input of each layer in SCNN is in the form of binary voltage and can be implemented by a single flash array. ADCs/DACs are no longer needed in the spiking CNN. The hardware cost of the signal transforming process can be significantly reduced.

---

**Figure 3.** Schematic of the fully connected layer operation on the flash memory array. Vector \(x\) is input from WLs in the form of voltage pulses, matrix \(k\) is stored as the threshold voltage of the floating gate transistor, output \(y\) is output from BLs in the form of current.
Figure 5a shows the structure of the flash memory array-based spiking neural network, which is divided into four types of blocks: sampling block, flash array, neuron, and spike counter. During the period of $0\rightarrow t$, the sampling block takes samples of the normalized input image ($0\rightarrow 1$) multiple times. The sampling period is $T$, and the number of sampling is $t/T$. The input spikes $(0/1)$ applied to the flash array are generated during each sampling. The sampling block takes samples of the normalized input image $(0\rightarrow 1)$ and generates input spikes $(0/1)$ during $0\rightarrow t$ with a time step of $T$. The available sampling methods include random sampling, Gaussian sampling, and Poisson sampling, [50] determined by the application scenario. The input spikes and the weights stored in the flash array are multiplied and accumulated to obtain the result of MAC operations in the form of current. Then the neuron circuit (Figure 5b) integrates the current of BL. If the integrated voltage ($V_{in}$) is larger than the reference threshold voltage ($V_{TH}$), the integrator generates a high-voltage level and triggers the D-type flip-flop to produce a spike $(1)$. Meanwhile, the neuron is reset to the initial state ($V_{in} = 0$). If $V_{in}$ is lower than $V_{TH}$, $V_{in}$ is maintained until the next sampling, and the neuron would not produce the spike to the next layer $(0)$. Therefore, the output of the neuron is 1 or 0, which means that the input of the subsequent layer is binary and ADCs/DACs are no longer needed in the spiking CNN. The simulation results of the neuron circuit are shown in Figure 5c. $R$, $C$, and $T$ are set as 250 kΩ, 1 pF, and 20 ns, respectively. It is found that pulses with the amplitude of 0.9 V and the duty cycle of 50% (red line) enable the trigger and reset of the neuron at $t = 70$ ns. However, if the duty cycle is decreased to 17% (blue line), the neuron could not generate the spike at $t = 70$ ns. The main reason for the fact is that the discharge of the capacitor of the integrator leads to a decrease of $V_{in}$. In other words, $V_{in}$ is not maintained during multiple samplings. For the recognition task based on the spiking CNN, the problem is destructive for the performance (accuracy) of the neural network. [51] To address the problem, the preset $V_{TH}$ is reduced to compensate for the discharge of the neuron. [51] During the period of $0\rightarrow t$, N spike counters of the output layer count the pulses of each node respectively, and the node with the largest number of pulses corresponds to the recognition result.

In summary, the flash memory-array based spiking neural network eliminates AD/DA converters, which occupy more than 85% of the hardware cost and the energy consumption in the flash memory array-based mixed-signal computing system. In addition, the output of each layer in the flash memory array-based spiking neural network depends on the relative value of $V_{in}$ and $V_{TH}$ instead of the precise value. In that case, the computing errors induced by the noise can be minimized as long as the relative value of $V_{in}$ and $V_{TH}$ is not changed. According to the simulation results in ref. [51], the tolerance of image noise of the spiking CNN can be greatly enhanced as shown in Figure 5d.

### 4. Impact of Flash Cell Reliability on Neural Network

When applying flash memory array into in-memory computing applications, there still exist challenges to the practical application.
Figure 5. a) Structure of flash memory array-based spiking neural network; b) Schematic and c) simulation results of the neuron circuit. d) The dependence of the accuracy on the noise intensity with 30% noise density. Reproduced under the terms and conditions of the CC-BY 4.0 license.\cite{51}

Copyright 2020, The Author(s). Published by IEEE.

Figure 6. a) Resnet-18 neural network architecture with the frxup initialization\cite{53} Compared with the standard architecture, the batch normalization layers which have unique parameters for different output channels are replaced with the bias and scale parameters applying to all the output channels; b) The top-1 and top-5 accuracy of ImageNet dataset with different variations; c) The relationship between the top-1 accuracy and the number of cells involved in one computing clock.
In this section, the challenges include device variation and retention behavior are discussed in detail.

4.1. Device Variation

In the fabrication process of the flash memory array, random variations of critical dimensions, thickness, and doping exist. These uncertain factors cause $V_{th}$ variation among flash cells and bring a challenge to distinguish output results. For example, the currents from flash cells representing output “1” range from 0.6 to 1 μA. The sum operation is realized according to Kirchhoff’s law, the current summation of two “1” cells with 1 μA is 2 μA, and the corresponding value is 2. The current summation of three “1” cells with 0.6 μA is 1.8 μA, and the corresponding value is 3. Theoretically, the current representing value 3 should be larger than the one representing value 2. However, due to the device variation, the current representing value 3 may be lower than the one value 2, causing the hardware computing results different from the theoretical results. To overcome this challenge, we can precisely tune the flash cell with the write–verify writing scheme. However, to implement such a scheme, additional circuits, energy, and time are needed. Although the neural networks have a certain tolerance to the computing error, we can adjust the tolerable variation range.

To better explain this problem, we explore the variation influence on the recognition accuracy of the ImageNet dataset using the classic Resnet-18 architecture with the $x$up initialization. The network structure is shown in Figure 6a. Here, the impact of device variation is confined to the inference process. Compared with the standard Resnet-18 architecture, this architecture eliminates batch normalization layers and is more hardware-friendly for the neural network computation. The weights are quantized to signed 5 bits, and the input are quantized to positive 6 bits. During the computation executed by the flash memory array, only one bit of input and one bit of the weight are involved. The multiplication and accumulation results are multiplied with corresponding coefficients. The variation here is denoted by the standard current deviation divided by the mean current. The relationship of the recognition accuracy with the current variation is shown in Figure 6b. With variation increases, the boundaries between adjacent results are increasingly blurred, and the recognition accuracy is reduced compared with the one when there is no current variation. When the variation does not exceed 0.1, the recognition accuracy loss is less than 2%. For some applications, the recognition accuracy is sufficiently high. Therefore, the requirements of practical applications determine the variation range of the flash cells. For the neural network application, when the array size increases, the accuracy is usually decreased. Therefore, to improve the recognition accuracy, the cells corresponding to one output channel can be divided into groups and computed sequentially. The sequentially computing results are summed to get the final result. The number of cells in each separated group decreases, the overlap probability between adjacent computing results decreases, thus increasing the computing accuracy, as shown in Figure 6c. Also, as shown in Figure 6c, when the device variation increases, the improvement of the decreasing number of cells per group increases. However, with a decreased number of cells, the number of groups increases, as the computing is executed sequentially, the computing speed is decreased. Therefore, a trade-off between the recognition accuracy and the speed is needed when adopting this method.

4.2. Retention Behavior

Multi-bit data storage per cell is crucial to improve the computing capacity and reduce the cost per bit. However, with the number of bits stored in one cell increases, the sensing margin between adjacent values become tighter, which may deteriorate the performance of flash memory array-based neural network. Therefore, the impacts of the retention behavior of the multi-bit flash on the performance of flash memory array-based neural network were studied.

To better understand the retention behavior of the multi-bit flash, the flash array with 4-bit $I_d$ distributions is baked at 175/200/225 °C for 1 × 10⁵ s. The measured retention behavior of 16 states of 50 flash cells at 225 °C is shown in Figure 7a, in which gray lines are raw data of 50 flash cells, and solid lines are the mean currents of different states. With the increasing baking time, the electrons stored in the floating gate are ejected into the channel through the Fowler–Nordheim (FN) tunneling and trap-assisted tunneling (TAT). Therefore, the threshold voltage.

![Figure 7](image-url)
(V_{ds}) of the flash cell decreases and \( I_d \) increases at fixed \( V_g \) and \( V_{ds} \) with time. Although the mean \( I_d \) of different states show the same trend, there is a slight difference. As there are more electrons stored in the floating gate of the flash cell, the states with low \( I_d \) suffer from more severe charge loss and retention degradation. It is also found that the distribution is spread as the baking time increases (gray lines). Figure 7b shows the measured deviation of \( I_d \) (\( \sigma(I_d) \)) of states "0100", "0111", "0110" and "0111" (\( I_d \) = 8, 10, 12, and 14 \( \mu \)A) at different time (10^8s, 10^7s, 5 \times 10^7s, 10^6s, 5 \times 10^6s, 10^5s). It is found that \( \sigma(I_d) \) increases with the time and the overlapping among neighboring states becomes more severe.

To study the impact of the retention of multi-bit flash on the performance of the flash memory array-based neural network, the 11-layer (5 convolution layers, 5 pooling layers, and 1 fully connected layer) flash memory array-based DNN for Cifar-10 recognition is simulated. The schematic of the network is shown in Figure 8a. The weights of the network are trained by CPU and then quantized to 4 bits. The quantized weights are stored in the flash array using two flash cells as the synapse. Figure 8b shows the recognition accuracy of the neural network at different baking times at 175/200/225°C. It is observed that the accuracy of the network reaches 83.64% at 10^5 s (225°C). To eliminate the adverse effects of the retention degradation on the performance of the neural network as far as possible, 4 flash cells are used to store the weight.\(^{(33)}\) In detail, the states of each flash cell stored are decreased from 4 bits to 3 bits. Simultaneously, to promote the accuracy of the network, every pair of the flash cells store the most/least significant bit (MSB/LSB) of the weight, respectively. By using the optimized scheme, the recognition accuracy reaches 83.64% at 10^5 s (225°C), enhancing the reliability of the flash memory array-based neural network greatly.

5. Conclusion

We systematically introduce how to implement neural networks from the bottom device level to the architecture level. First, the MAC operating principle with different input forms is proposed. Based on the MAC operating principle, the hardware implementation methods of the CNN and the spiking CNN are given. Finally, the reliability of the flash memory array-based neural network including the device variation and the retention behavior are discussed. The results showed that the flash memory array has high computing flexibility and can implement different kinds of neural networks. The flash memory cell can also be programmed to a precise state and can hold the states at a temperature below 200°C. With a steady increase in the integrated density of the flash memory and mature technology, the flash memory array will become a strong competitor to implement efficient neural network computations.

Acknowledgements

This work was supported in part by the National Key Research and Development (2019YFB2205100, 2018YFA0701501), NSFC (61841404 and 61421005).

Conflict of Interest

The authors declare no conflict of interest.

Keywords

analog neural networks, flash memory, in-memory computing, multiplication-and-accumulation units, neural networks, spiking neural networks

Received: July 15, 2020
Revised: October 3, 2020
Published online: December 4, 2020

References

[1] V. Sze, Y. H. Chen, T. J. Yang, J. S. Emer, Proc. IEEE 2017, 105, 12.
[2] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, Y. Shi, Nat. Electron. 2018, 1, 4.
[3] J. Tang, D. Sun, S. Liu, J. L. Gaudiot, Computer 2017, 50, 10.
[4] M. T. Yazici, S. Basurra, M. M. Gaber, Big Data Cogn. Comput. 2018, 2, 26.
[5] M. M. Waldrop, Nature 2016, 530, 144.
[6] J. V. Neumann, IEEE Ann. Hist. Comput. 1993, 15, 27.
Runze Han received his B.E. degree from University of Electronics Science and Technology of China, Chengdu, China, in 2015 and Ph.D. degree in microelectronics from Peking University, Beijing, China, in 2020. He is now a postdoc at the Institute of Microelectronics, Peking University and currently focuses on in-memory computing research based on flash memory.

Peng Huang received his B.S. degree from Xidian University, Xi’an, China, in 2010 and Ph.D. degree in microelectronics from Peking University, Beijing, China, in 2015. He is currently a research associate in the Institute of Microelectronics, Peking University. His research interest is in-memory computing and neural network computing based on novel devices.

Jinfeng Kang received his B.S. degree in physics from Dalian University of Technology in 1984, and Ph.D degree in solid-state electronics from Peking University in 1992. He is full professor of Electronics Engineering Computer Science School, Peking University with research interest in novel nano devices for computing and data storage.