Binary ReRAM-based BNN first-layer implementation

Mona Ezzadeen[1], Atreya Majumdar[3], Sigrid Thomas[1,2], Jean-Philippe Noël[2], Bastien Giraud[2], Marc Bocquet[4], François Andrieu[1], Damien Querlioz[3], and Jean-Michel Portal[4]

[1]Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France, mona.ezzadeen@cea.fr
[2]Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France
[3]Université Paris-Saclay, CNRS, Centre de Nanosciences et de Nanotechnologies, 91120 Palaiseau, France
[4]Aix-Marseille Univ., CNRS, IM2NP, Marseille, France

***Abstract*—The deployment of Edge AI requires energy-efficient hardware with a minimal memory footprint to achieve optimal performance. One approach to meet this challenge is the use of Binary Neural Networks (BNNs) based on non-volatile in-memory computing (IMC). In recent years, elegant ReRAM-based IMC solutions for BNNs have been developed, but they do not extend to the first layer of a BNN, which typically requires non-binary activations. In this paper, we propose a modified first layer architecture for BNNs that uses k-bit input images broken down into k binary input images with associated fully binary convolution layers and an accumulation layer with fixed weights of $2^{-1}, ..., 2^{-k}$. To further increase energy efficiency, we also propose reducing the number of operations by truncating 8-bit RGB pixel code to the 4 most significant bits (MSB). Our proposed architecture only reduces network accuracy by 0.28% on the CIFAR-10 task compared to a BNN baseline. Additionally, we propose a cost-effective solution to implement the weighted accumulation using successive charge sharing operations on an existing ReRAM-based IMC solution. This solution is validated through functional electrical simulations.**

***Index Terms*—Binary Neural Network (BNN), convolution, accumulation, charge sharing**

I. Introduction

The rapid growth of the Internet of Things (IoT) market and the increasing performance of AI algorithms have placed conflicting demands on neuromorphic edge accelerators. These devices have limited storage capacity, but traditional neural network architectures often require large amounts of memory to store their synaptic weights. Binary Neural Networks (BNNs) are a solution to solve this problem, as they use binary weights and activations to greatly reduce the memory footprint while maintaining high accuracy. BNNs also simplify the computation process by replacing full-precision multiplications with XNOR operations and the accumulation operation with bit counting operations (popcount) [1], [2]. However, the large amount of data movement between processor cores and memories can lead to significant energy consumption [3], known as the von Neumann bottleneck. In this context, In-memory-computing (IMC) based accelerators, based on non-volatile memories such as ReRAM [4]–[11], are promising solutions, as they perform computations natively inside the memory, allowing for energy-efficient, ultra-parallel neuromorphic operations and eliminating the von Neumann bottleneck. Additionally, the non-volatility of these memories offers a power-off capability with an excellent retention [12], which is crucial for edge devices with limited energy budgets.

In recent years, the field of BNN IMC has seen significant advancements in the use of ReRAM-based solutions [13]–[16]. Despite these developments, many existing solutions are not equipped to handle the first layer of a BNN, which requires non-binary activations. Other approaches, utilizing SRAMs [17], [18], have proposed either complex analog circuits for the first layer or thermometer encoding, but these solutions lack the power-off and instant-on capabilities offered by non-volatile memories such as ReRAM.

In this paper, we propose a novel first-layer architecture for BNNs that utilizes exclusively binary MAC operations, making it fully compatible with state-of-the-art ReRAM-based In-Memory Computing (IMC) accelerators [16]. We thoroughly evaluate the performance of our proposed architecture on CIFAR10 tasks [19] and use approximate computing to further reduce operation counts and memory capacity. Additionally, we propose a hardware implementation of our approach utilizing ReRAM-based in/near-memory computing.

The main contributions of this paper are:

- The development of a modified first layer architecture for BNNs, that is fully binary, compatible with ReRAM-based IMC accelerators. This is achieved by breaking down a k-bit input image into k binary input images

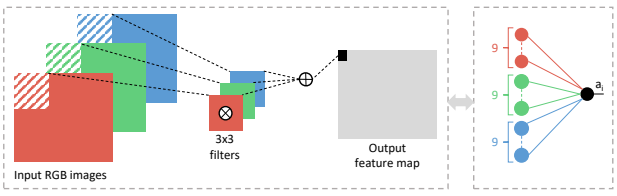

Fig. 1. First Layer CNN classical architecture, with one feature map channel output. The convolution of an input image with a given filter at a certain position can also be thought of as a fully connected neuron with 27 inputs (3×3×3) and one output activation

This work was supported by European Research Council starting grant through the My-CUBE (820048) and NANOINFER (reference: 715872) projects, and a France 2030 grant managed by the French ANR (ANR-22-PEEL-0010).

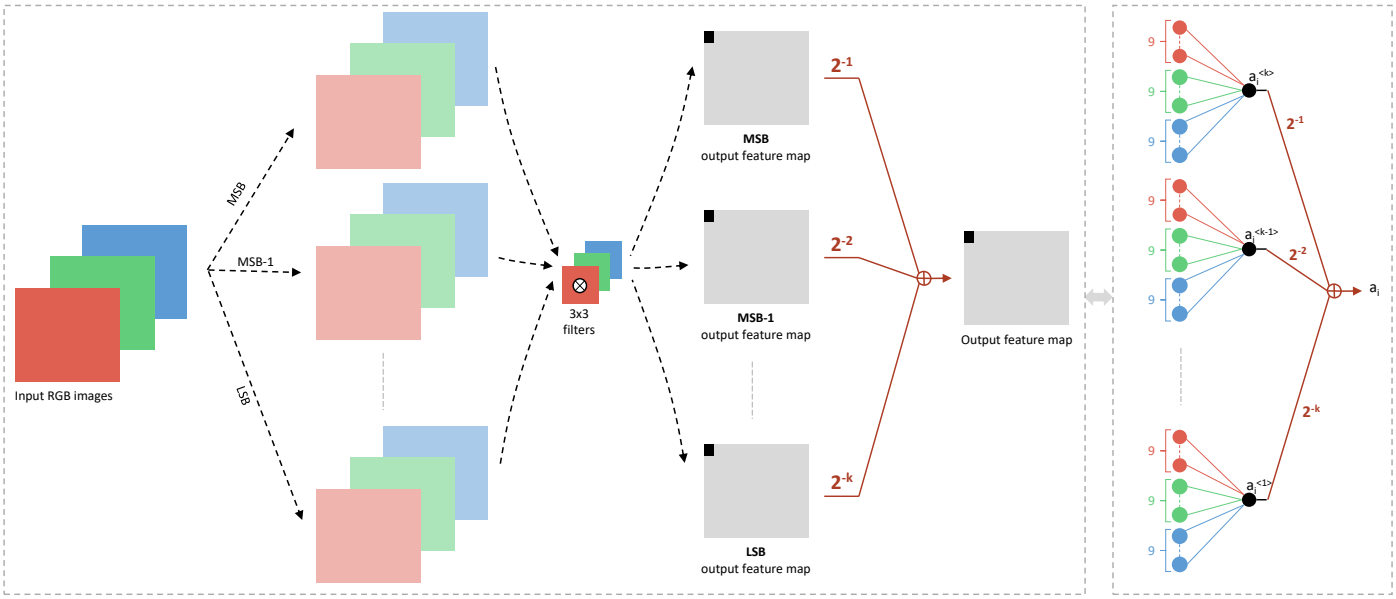

Fig. 2. Proposed architecture : a k-bit input image is split into k binary input images with their associated fully binary convolution layer, to obtain k partial feature map $a_i^{<j>}$ and then an weighted accumulation layer is introduced to obtain the output feature map a_i.

and implementing fully binary convolution layers and an accumulation layer with fixed weights.

- The use of approximate computing principles to reduce the number of operations required for this modification, by truncating 8-bit RGB pixel code to the 4 most significant bits (MSB) with minimal impact on network accuracy.
- The proposal of an original solution based on successive charge sharing to implement on-chip, weighted accumulation of partial feature map activations.

The rest of the paper is organized as follows: Section II presents the evaluation of the proposed hardware-friendly first layer in terms of accuracy compared to the classical baseline on the CIFAR-10 dataset with a Visual Geometry Group (VGG) [20] like network structure, Section III details the circuit architecture focusing on the weighted accumulation with successive charge sharing, and finally, Section IV concludes the paper.

II. BNN FIRST LAYER BINARIZATION

A. First layer binarization principle

The architecture of the first layer of a classical CNN is shown in Fig.1, where a colored input image typically has three channels, representing the red, green, and blue (RGB) components of the pixels, coded each on 8 bits. A filter, usually 3×3 in size and also with three channels, is applied to the input image and moved across it. At each position of the filter, the input pixels under it are multiplied by their corresponding weight in the filter. The results of these multiplications for the three channels are then added together to create one activation for the next feature map. By moving the filter across the input image, all the activations for the next feature map are generated. To detect more features in the input image, multiple filters can be applied. This results in the feature map having multiple channels, each channel being generated by a different

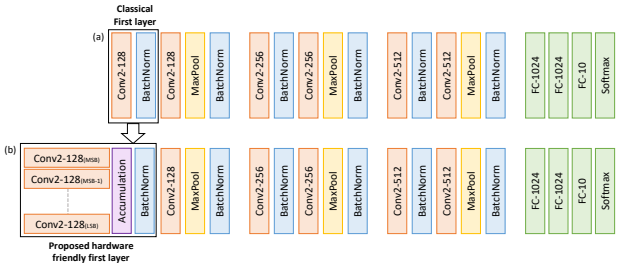

Fig. 3. Illustration of the VGG like network with identification of all layers with (a) the classical architecture and (b) the proposed hardware friendly first layer.

filter. After the convolutional layer, batch-normalization and sign layers are applied to compare the activations to threshold values, fully binarizing the feature maps. The convolution of an input image with a given filter at a certain position can also be thought of as a fully connected neuron with 27 inputs (3×3×3) and one output activation a_i.

In classical BNNs, the input image is not converted to a binary format in order to maintain a high level of accuracy during the inference process. This means that when building a hardware accelerator for a BNN, the design of the first layer must be different from the rest of the fully binary layers and requires a more complex structure.

We propose a method to overcome this challenge by first breaking down a k-bit input image into k binary input images, as illustrated in Fig.2. Then, the convolution process is performed using only binary operations and generates k different feature maps. Doing so, the standard BNN multiplication replacement with XNOR can be used, opening in-memory computation capability. To combine these feature maps into a single one, we use a power of two weighting. The resulting single feature map

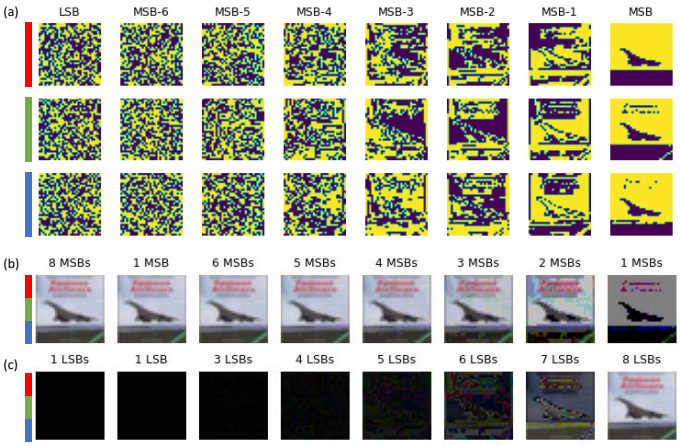

Fig. 4. Analysis of the CIFAR10 dataset bits contributions. (a) Airplane image bitmap with its three RGB channels and 8 bits separated (from LSB to the MSB), (b)-(c) same image with its three channels but with only a portion of its 8 bits, with (b) starting from the 8 MSB to only 1 MSB, and (b) from 1 LSB to 8 LSB.

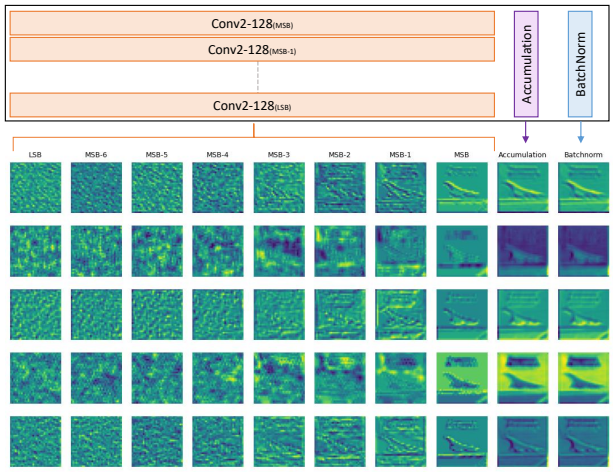

Fig. 5. Analysis of intermediate activations inside our custom first layer, for the same image as in 4, and for the first five channels (over 128).

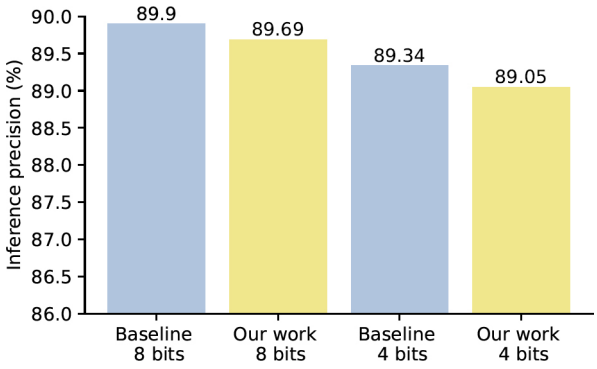

Fig. 6. The VGG inference accuracy is represented versus the different options (baseline on 8 bits coded image and with the code of the input image pixel truncated to the 4 MSB bits as references and the same network with our hardware friendly first layer here also on 8 bits and truncated to the 4 MSB bits)

is then binarized using batch-normalization and sign layers.

This first layer architecture is similar to having k fully connected layers with 27 binary inputs and one output activation $a_i^{<j>}$, with j going from 1 to k, or from the Most Significant Bit (MSB) to the Least Significant Bit (LSB). Finally, the accumulation step corresponds to a classical fully connected layer with fixed weights equal to $2^{-1}, ..., 2^{-k}$.

We want to benchmark the performance of the proposed first layer architecture versus the classical one, using Tensorflow [21] on the CIFAR10 dataset. We use a VGG-like network as a baseline, with 8-bit activations for the input image. We also use the same network, but with our proposed first layer, which is designed to be hardware friendly. The figure Fig.3 illustrates the two different network architectures. Our proposed architecture replaces the traditional multi-bit convolution in the first layer with 8 convolutional layers (with shared weights) and an accumulation layer. Each convolutional layer j processes input images of a specific bit of rank j, generating a partial output feature map $a^{<j>}$. These partial output feature maps are then combined, using our specific accumulation layer, to create the final output feature map.

The baseline accuracy during inference is equal to 89,9%, whereas the same network with our hardware friendly first layer achieved an accuracy of 89.69%. This result fully validates our approach since, we face a drop of only 0.21%.

B. Binarized first layer optimization

Our solution aims to improve its hardware friendliness by reducing the amount of computation and memory required, which in turn reduces power consumption. We propose to use the principle of approximate computing. By reconstructing images starting with the least significant bits (LSB) and progressively adding more bits to the pixel code, we can get a hint of the image content after 6 bits (see Fig.4b). On the other hand, if we reconstruct images by accumulating bits starting with the most significant bits (MSB), we can clearly see the contours of the image with just the MSB value. Additionally, by analyzing the bit ranks j of images pixel from the CIFAR10 dataset, we can visually see that most of the useful information is captured by the four most significant bits, as shown in Fig.4a.

Our proposed first layer approach allows for the same type of analysis to be performed on intermediate values within the network. Fig.5 illustrates the intermediate activations after the eight convolutional layers, each corresponding to a different bit rank, for the same image as in Fig.4. It also includes the activations obtained after accumulation, batchnorm, and sign layers. Similarly to the input image analysis, we can see that the first four most significant bits (MSB) of the intermediate activations contain more visually meaningful information, while the last four least significant bits (LSB) contain more noisy information. Based on these observations, the main idea is to use an approximate computing approach and only rely on the four MSB bits of each pixel of the image.

We conduct simulations using Tensorflow to compare the performance of our baseline and our custom first layer ap-

proach, where the input pixels are truncated to the four most significant bits (MSB). The results of the inference precision are shown in Fig.6. Compared to the 8-bit approach, the baseline accuracy dropped by only 0.56%, resulting in a very good inference accuracy of 89.34%. This confirms the results of our visual analysis on the dataset images. Our proposed first layer approach shows a similar trend, with a good inference accuracy of 89.05%, which corresponds to a drop in accuracy of only 0.29% compared to the 4-bit baseline. This results clearly validate that an approximation can be performed on the input pixels' code, keeping only the 4 MSB bits.

III. Hardware Implementation

Our first layer architecture allows for the use of any in-memory computing solutions that perform XNOR operations and popcount to compute the convolution part of the layer. This is one of the main goals of the first layer modification. However, to obtain a complete circuit, supporting the proposed first layer architecture, we have now to develop a custom solution for the weighted accumulation. To achieve this, we propose an original solution that is based on successive charge sharing of the partial feature map activations $a^{<j>}$.

As an initial hardware solution for the modified first layer architecture, we use the in/near memory computing circuit proposed in [16]. Fig.7 illustrates the initial circuit architecture. The circuit uses a ReRAM array to store binary weights in a complementary fashion using 2 Transistor - 2 ReRAM (2T2R) cells. The binary input activations are also applied in a complementary fashion on the Bit Line (BL) and complementary BL (BL_b). Each 2T2R cell is in a resistive divider configuration, with the Source Line (SL) being the middle point, whose value corresponds to $XOR(x_m, w_{i,m})$. A simple inverter is then added at the bottom of the SL, which converts the XOR analog output to a clean binary $XNOR(x_m, w_{i,m})$ value. The popcount operation is then performed using a capacitive divider, taking inspiration of an original work proposed in [17], [18]. The batchnorm and sign operations are done by comparing the capacitive divider value to a trained threshold voltage. The main advantages of this circuit are its high robustness against intrinsic ReRAM variability and its low energy consumption. This circuit is already able to compute all the BNN operations

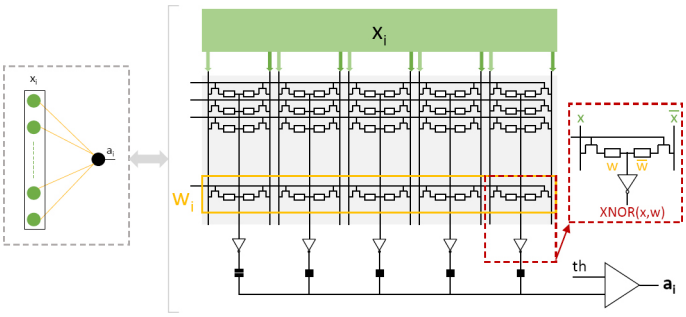

Fig. 7. ReRAM based BNN accelerator, with XNOR operation performed in memory and accumulation performed through a capacitive neuron. Comparison to the threshold value and sign function are performed with a comparator proposed in [16]

and we focus here on adapting it to the weighted accumulation of partial feature maps activations.

A. Charge sharing accumulation

The proposed weighted accumulation technique is based on successive charge sharing operations. This means that the partial feature map activation $a_i^{<j>}$ has to be stored as a voltage $V_{a_i^{<j>}}$ across a capacitor.

As shown in Fig.8, the voltage $V_{a_i^{<j>}}$ across the capacitor is directly proportional to the partial feature map activation $a_i^{<j>}$. Additionally, $a_i^{<j>}$ is in the range of $[\![0, 27]\!]$, which corresponds to a voltage $V_{a_i^{<j>}}$ between 0 and the supply voltage V_{DD}. Therefore, we can express the relationship between $V_{a_i^{<j>}}$ and $a_i^{<j>}$ as:

$$V_{a_i^{<j>}} = \frac{V_{DD}}{27}.a_i^{<j>} \quad (1)$$

Where V_{DD} is the supply voltage and the voltage across the capacitor is linearly proportional to the activation value.

The proposed technique for weighted accumulation uses a series of charge sharing operations. This process begins by storing the activation value of the partial feature map corresponding to the least significant bit (LSB) in an activation capacitor (C_{acti}). This value is then shared with an accumulation capacitor (C_{accu}). This process is repeated for each successive partial feature map activation $a_i^{<j>}$, up to the activation corresponding to the most significant bit (MSB). By the end of this process, the voltage across the accumulation capacitor ($V(C_{accu})$) represents the total weighted accumulation of all the partial activations.

The sharing process can be described step by step. Following eq.1, the $a_i^{<1>}$ activation value is stored as a voltage level $V_{a_i^{<1>}}$ in a activation capacitor C_{acti}. A second capacitor, C_{accu}, of the same size as C_{acti}, is initially grounded, so that $V(C_{accu}{}^{<0>}) = 0$ volts. The two capacitors are then connected

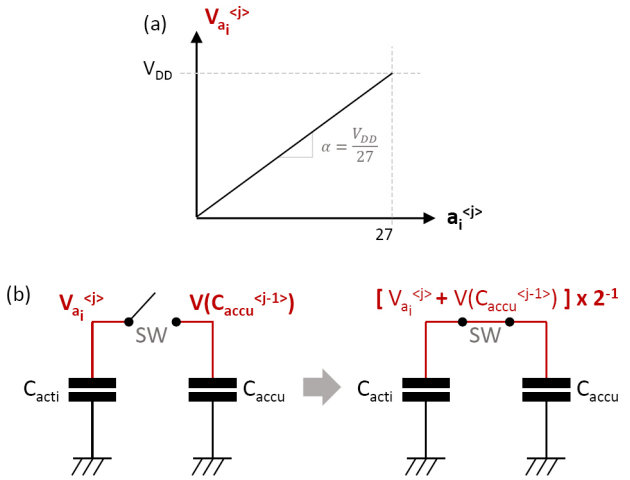

Fig. 8. (a) $a_i^{<j>}$ analog voltage equivalence, (b) weighted $a_i^{<j>}$ accumulation using capacitive sharing

and following the charge conservation law, the resulting voltage is given by:

$$V(C_{\text{accu}}{}^{<1>}) = \frac{V(C_{acti}^{<0>}) + V(C_{accu}^{<0>})}{2} = \frac{V_{a_i^{<1>}}}{2} \quad (2)$$

The process is then repeated with the next activations $a_i^{<2>}$,...,$a_i^{<k>}$, without clearing the accumulation capacitor C_{accu}. We can show recursively that, after the kth accumulation, the voltage accross C_{accu} is given by:

$$\begin{aligned} V(C_{accu}^{<k>}) &= 2^{-1}.V_{a_i^{<k>}} + 2^{-2}.V_{a_i^{<k-1>}} + ... + 2^{-k}.V_{a_i^{<1>}} \\ &= \sum_{j=1}^{k} 2^{-j}.V_{a_i^{<k-j+1>}} \\ &= \frac{V_{DD}}{27} \sum_{j=1}^{k} 2^{-j}.a_i^{<k-j+1>} \end{aligned} \quad (3)$$

with, the $V(C_{accu}^{<k>})$ voltage level corresponding to the a_i element of the output feature map.

In the circuit shown in Figure 7, an additional analog switch and accumulation capacitor are needed. They are added between the capacitive bridge and the comparator. It is important to note that the capacitive bridge acts as the activation capacitor in this case. The new circuit design is shown in Figure 9.

B. Simulation results

To validate our approach, we carried electrical simulations using an industrial 130nm Design Kit. We used capacitors of size 2 fF for the capacitive divider, and of size 27×2 = 54 fF for C_{accu}. The simulated chronogram of Fig.10 illustrates the circuit operation.

To produce the output activation a_i, the row of weights corresponding to the neuron is activated. Next, the capacitive divider and the accumulation capacitor (C_{accu}) are reset. For each time step, j, where j is between 1 and 4, the binary neuron input activations $x_m^{<j>}$ of rank j are applied to the BL/BL$_b$ circuit. This performs a MAC (multiply-accumulate) between the inputs and the weights $w_{i,m}$, resulting in a voltage $V_{a_i^{<j>}}$ on

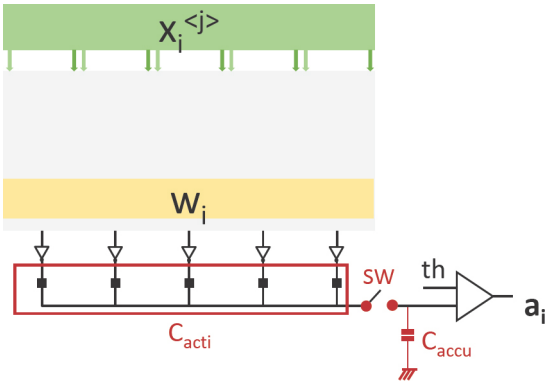

Fig. 9. ReRAM-based BNN first layer implementation : compared to the initial scheme the capacitive bridge used for the accumulation operation acts as C_{acti} and an analog switch SW together with an extra capacitor C_{accu} are added to implement the weighted accumulation.

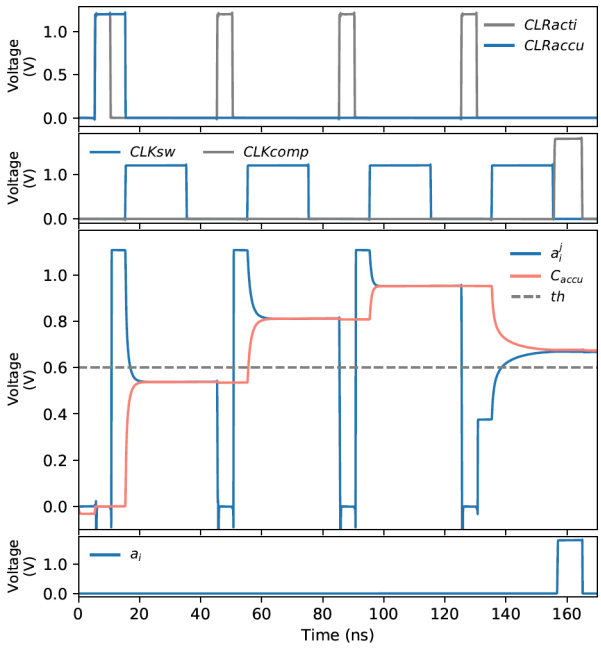

Fig. 10. The simulated chronogram presents the different steps of the weighted accumulation through successive charge sharing operation between C_{acti} and C_{accu}

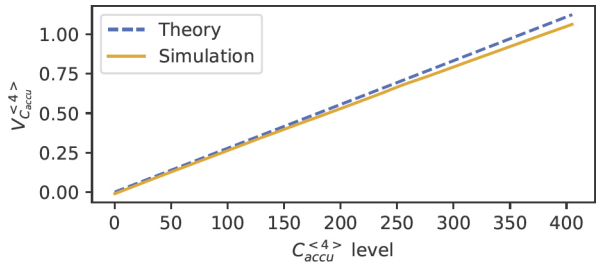

Fig. 11. The voltage $V(C_{accu}^{<k=4>})$ on the accumulation capacitor C_{accu} is plotted versus the number of possible level on the capacitor C_{accu} using the theoritical equation and the simulated results. A good agreement is reported

the capacitive divider. The switch SW is then closed to perform a partial accumulation on C_{accu}. After the partial accumulation step, SW is opened again, the capacitive divider is reset, and the next binary neuron input activations $x_m^{<j+1>}$ can be applied. Once the four partial accumulation steps are completed, the comparator is triggered, and the final binary output activation is generated at the comparator's output.

Fig.11.a shows all the possible simulated accumulation levels $V(C_{accu}^{<k=4>})$, for a clock period of 40ns. We can see that the accumulated values follow closely the theoretical values, but that the maximum $V(C_{accu}^{<k=4>})$ voltage is 61 mV lower than the theoretical value, which is mainly due to the size of the comparator's input capacitor with regards to C_{accu} capacitance. The minimum $\Delta V(C_{accu}^{<k=4>})$ is equal to 2.65 mV, which enables a sufficient voltage margin at the comparator's inputs.

IV. Conclusion

In this paper, we propose a modification to the first layer of a Binary Neural Network (BNN) that allows for fully binary in-memory computation. Currently, BNNs are limited by the fact that the first layer is fed with non-binary activation values, as image pixels are typically coded using 8-bits per RGB channel. Our solution addresses this issue by breaking down a k-bit input image into k binary input images with their associated fully binary convolution layers and adding an accumulation process that mimics a fully connected layer with fixed weights (equal to $2^{-1}, ..., 2^{-k}$). Additionally, we limit the number of accumulation by using approximate values and truncating the 8-bit image values to 4 most significant bits. Our solution leads to a very small decrease in accuracy of only 0.28% compared to a 4-bit baseline. As a result, the first layer becomes fully compatible with XNOR and popcount operations that can be performed in-memory. We also propose a cost-effective modification to a ReRAM based in-memory computation solution by adding only an analog switch and a capacitor to implement the weighted accumulation. This solution has been validated through functional electrical simulation. It's worth noting that the proposed modification of the first layer is compatible with any BNN hardware implementation that presents popcount values in an analog way, typically in the form of voltage.

References

[1] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[2] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016.

[3] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 10–14, ISSN: 2376-8606.

[4] W. Wan, R. Kubendran, C. Schaefer, S. B. Eryilmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, B. Gao, S. Joshi, H. Wu, H.-S. P. Wong, and G. Cauwenberghs, "A compute-in-memory chip based on resistive random-access memory," vol. 608, no. 7923, pp. 504–512. [Online]. Available: https://www.nature.com/articles/s41586-022-04992-8

[5] R. Mochida, K. Kouno, Y. Hayata, M. Nakayama, T. Ono, H. Suwa, R. Yasuhara, K. Katayama, T. Mikawa, and Y. Gohou, "A 4m synapses integrated analog ReRAM based 66.5 TOPS/w neural-network processor with cell current controlled writing and flexible network architecture," in *2018 IEEE Symposium on VLSI Technology*, 2018, pp. 175–176, ISSN: 2158-9682.

[6] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C.-X. Xue, W.-H. Chen, J. Tang, Y. Wang, M.-F. Chang, H. Qian, and H. Wu, "33.2 a fully integrated analog ReRAM based 78.4tops/w compute-in-memory chip with fully parallel MAC computing," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pp. 500–502, ISSN: 2376-8606.

[7] W.-H. Chen, K.-X. Li, W.-Y. Lin, K.-H. Hsu, P.-Y. Li, C.-H. Yang, C.-X. Xue, E.-Y. Yang, Y.-K. Chen, Y.-S. Chang, T.-H. Hsu, Y.-C. King, C.-J. Lin, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 65nm 1mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 494–496, ISSN: 2376-8606.

[8] C.-X. Xue, W.-H. Chen, J.-S. Liu, J.-F. Li, W.-Y. Lin, W.-E. Lin, J.-H. Wang, W.-C. Wei, T.-W. Chang, T.-C. Chang, T.-Y. Huang, H.-Y. Kao, S.-Y. Wei, Y.-C. Chiu, C.-Y. Lee, C.-C. Lo, Y.-C. King, C.-J. Lin, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "24.1 a 1mb multibit ReRAM computing-in-memory macro with 14.6ns parallel MAC computing time for CNN based AI edge processors," in *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*, pp. 388–390, ISSN: 2376-8606.

[9] J.-H. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "29.1 a 40nm 64kb 56.67tops/w read-disturb-tolerant compute-in-memory/digital RRAM macro with active-feedback-based read and in-situ write verification," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, pp. 404–406, ISSN: 2376-8606.

[10] C.-X. Xue, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, T.-W. Liu, S.-Y. Wei, S.-P. Huang, W.-C. Wei, Y.-R. Chen, T.-H. Hsu, Y.-K. Chen, Y.-C. Lo, T.-H. Wen, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "15.4 a 22nm 2mb ReRAM compute-in-memory macro with 121-28tops/w for multibit MAC computing for tiny AI edge devices," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. IEEE, 2020, pp. 244–246.

[11] C.-X. Xue, J.-M. Hung, H.-Y. Kao, Y.-H. Huang, S.-P. Huang, F.-C. Chang, P. Chen, T.-W. Liu, C.-J. Jhang, C.-I. Su, W.-S. Khwa, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Y.-D. Chih, T.-Y. J. Chang, and M.-F. Chang, "16.1 a 22nm 4mb 8b-precision ReRAM computing-in-memory macro with 11.91 to 195.7tops/w for tiny AI edge devices," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 245–247, ISSN: 2376-8606.

[12] Z. Wei, T. Takagi, Y. Kanzawa, Y. Katoh, T. Ninomiya, K. Kawai, S. Muraoka, S. Mitani, K. Katayama, S. Fujii, R. Miyanaga, Y. Kawashima, T. Mikawa, K. Shimakawa, and K. Aono, "Demonstration of high-density reram ensuring 10-year retention at 85°c based on a newly developed reliability model," in *2011 International Electron Devices Meeting*, 2011, pp. 31.4.1–31.4.4.

[13] T. Hirtzlin, M. Bocquet, B. Penkovsky, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays," *Frontiers in neuroscience*, vol. 13, p. 1383, 2020.

[14] L. Wang, W. Ye, C. Dou, X. Si, X. Xu, J. Liu, D. Shang, J. Gao, F. Zhang, Y. Liu, M.-F. Chang, and Q. Liu, "Efficient and robust nonvolatile computing-in-memory based on voltage division in 2t2r RRAM with input-dependent sensing control," vol. 68, no. 5, pp. 1640–1644, 2021.

[15] J.-M. Hung, Y.-H. Huang, S.-P. Huang, F.-C. Chang, T.-H. Wen, C.-I. Su, W.-S. Khwa, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Y.-D. Chih, T.-Y. J. Chang, and M.-F. Chang, "An 8-mb DC-current-free binary-to-8b precision ReRAM nonvolatile computing-in-memory macro using time-space-readout with 1286.4-21.6tops/w for edge-AI devices," in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, pp. 1–3, ISSN: 2376-8606.

[16] M. Ezzadeen, A. Majumdar, M. Bocquet, B. Giraud, J.-P. Noël, F. Andrieu, D. Querlioz, and J.-M. Portal, "Low-overhead implementation of binarized neural networks employing robust 2t2r resistive RAM bridges," in *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*, 2021, pp. 83–86.

[17] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28nm CMOS," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 222–224, ISSN: 2376-8606.

[18] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 141–142.

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," p. 60.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[21] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *Osdi*, vol. 16, no. 16. Savannah, GA, USA, 2016, pp. 265–283.