



**HAL**  
open science

## Low Power In-Memory Implementation of Ternary Neural Networks with Resistive RAM-Based Synapse

Axel Laborieux, Marc Bocquet, Tifenn Hirtzlin, Jacques-Olivier Klein, L Herrera Diez, Etienne Nowak, Elisa Vianello, Jean-Michel Portal, Damien S Querlio

### ► To cite this version:

Axel Laborieux, Marc Bocquet, Tifenn Hirtzlin, Jacques-Olivier Klein, L Herrera Diez, et al.. Low Power In-Memory Implementation of Ternary Neural Networks with Resistive RAM-Based Synapse. 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2020, Genova (virtual), Italy. 10.1109/AICAS48895.2020.9073877 . hal-02403984

**HAL Id: hal-02403984**

**<https://hal.science/hal-02403984v1>**

Submitted on 6 May 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Low Power In-Memory Implementation of Ternary Neural Networks with Resistive RAM-Based Synapse

A. Laborieux\*, M. Bocquet<sup>†</sup>, T. Hirtzlin\*, J.-O. Klein\*, L. Herrera Diez\*, E. Nowak<sup>‡</sup>,  
E. Vianello<sup>‡</sup>, J.-M. Portal<sup>†</sup> and D. Querlioz\*

\*Université Paris-Saclay, CNRS, C2N, 91120 Palaiseau, France. Email: damien.querlioz@c2n.upsaclay.fr

<sup>†</sup> IM2NP, Univ. Aix-Marseille et Toulon, CNRS, France. <sup>‡</sup>CEA, LETI, Grenoble, France.

**Abstract**—The design of systems implementing low precision neural networks with emerging memories such as resistive random access memory (RRAM) is a major lead for reducing the energy consumption of artificial intelligence (AI). Multiple works have for example proposed in-memory architectures to implement low power binarized neural networks. These simple neural networks, where synaptic weights and neuronal activations assume binary values, can indeed approach state-of-the-art performance on vision tasks. In this work, we revisit one of these architectures where synapses are implemented in a differential fashion to reduce bit errors, and synaptic weights are read using precharge sense amplifiers. Based on experimental measurements on a hybrid 130 nm CMOS/RRAM chip and on circuit simulation, we show that the same memory array architecture can be used to implement ternary weights instead of binary weights, and that this technique is particularly appropriate if the sense amplifier is operated in near-threshold regime. We also show based on neural network simulation on the CIFAR-10 image recognition task that going from binary to ternary neural networks significantly increases neural network performance. These results highlight that AI circuits function may sometimes be revisited when operated in low power regimes.

## I. INTRODUCTION

Artificial Intelligence has made tremendous progress in recent years due to the development of deep neural networks. Its deployment at the edge, however, is currently limited by the high power consumption of the associated algorithms [1]. Low precision neural networks are currently emerging as a solution, as they allow the development of low power consumption hardware specialized on deep learning inference [2]. The most extreme case of low precision neural network, the Binarized Neural Network (BNN), also called XNOR-NET, is receiving special attention as it is particularly efficient for hardware implementation: both synaptic weights and neuronal activations assume only binary values [3], [4]. Remarkably, this type of neural network can approach near-state-of-the-art performance on vision tasks [5]. One particularly investigated lead is to fabricate hardware BNNs with emerging memories such as resistive RAM or memristors [6]–[13]. The low memory requirements of BNNs, as well as their reliance on simple arithmetic operations, make them indeed particularly adapted

This work was supported by the ERC Grant NANOINFER (715872) and the ANR grant NEURONIC (ANR-18-CE24-0009).

for “in-memory” or “near-memory” computing approaches, which achieve superior energy-efficiency by avoiding the von-Neumann bottleneck entirely.

In this work, we revisit one of this hardware developed for the energy-efficient in-memory implementation of BNNs [6], where the synaptic weights are implemented in a differential fashion. We show that it can be extended to a more complex form of low precision neural networks, ternary neural network [14] (TNN, also called Gated XNOR-NET, or GXNOR-NET [15]), where both synaptic weights and neuronal activations assume ternary values. The contribution of this work is as follows. After presenting the background of the work (section II):

- We demonstrate experimentally on a fabricated 130 nm RRAM/CMOS hybrid chip a strategy for implementing ternary weights using a precharge sense amplifier, which is particularly appropriate when the sense amplifier is operated in the near-threshold regime (sec. III).
- We carry simulations to show the superiority of TNNs over BNNs on the canonical CIFAR-10 vision task, and evidence the error resilience of hardware TNNs (sec. IV).

## II. BACKGROUND

The main equation in conventional neural networks is the computation of the neuronal activation  $A_j = f(\sum_i W_{ji} X_i)$ , where  $A_j$ , the synaptic weights  $W_{ji}$  and input neuronal activations  $X_i$  assume real values, and  $f$  is a non-linear activation function. BNNs are a considerable simplification of conventional neural networks, in which all neuronal activations ( $A_j, X_i$ ) and synaptic weights  $W_{ji}$  can only take binary values meaning  $+1$  and  $-1$ . Neuronal activation then becomes:

$$A_j = \text{sign} \left( \sum_i \text{XNOR}(W_{ji}, X_i) - T_j \right), \quad (1)$$

where  $\text{sign}$  is the sign function,  $T_j$  is a threshold associated with the neuron, and the  $\text{XNOR}$  operation is defined in Table I. Training BNNs is a relatively sophisticated operation, during which synapses need to be associated with a real value in addition to their binary value. Once training is finished, these real values can be discarded and the neural network is entirely binarized. Due to their reduced memory requirements,

TABLE I  
TRUTH TABLES OF THE XNOR AND GXNOR GATES

$W_{ji}$	$X_i$	XNOR	$W_{ji}$	$X_i$	GXNOR
-1	-1	1	-1	-1	1
-1	1	-1	-1	1	-1
1	-1	-1	1	-1	1
1	1	1	1	1	1
			0	X	0
			X	0	0

and reliance on simple arithmetic operations, BNNs are especially appropriate for in- or near- memory implementations. In particular, multiple groups investigate the implementation of BNN inference with resistive memory tightly integrated at the core of CMOS [6]–[13]. Usually, resistive memory stores the synaptic weights  $W_{ji}$ . However, this comes with a significant challenge: resistive memory is prone to bit errors, and in digital applications, is typically used with strong error correcting codes (ECC). ECC, which requires large decoding circuit [16], goes against the principles of in- or near- memory computing. For this reason, [6] proposes a two-transistor/two-resistor (2T2R) structure, which reduces resistive memory bit errors, without the need of ECC decoding circuit, by storing synaptic weights in a differential fashion. This allows for extremely efficient implementation of BNNs, and to use the resistive memory devices in very favorable programming conditions (low energy, high endurance).

In this work, we show that the same architecture may be used for a generalization of BNNs, TNNs<sup>1</sup>, where neuronal activations and synaptic weights  $A_j$ ,  $X_i$  and  $W_{ji}$  can now assume three values: +1, -1 and 0. Equation (1) now becomes:

$$A_j = \phi \left( \sum_i GXNOR(W_{ji}, X_i) - T_j \right); \quad (2)$$

*GXNOR* is the “gated” XNOR operation that realizes the product between numbers with values +1, -1 and 0 (Table I).  $\phi$  is an activation function that outputs +1 if its input is greater than a threshold  $\Delta$ , -1 if the input is lesser than  $-\Delta$  and 0 otherwise. We show experimentally and by circuit simulation in sec. III how the 2T2R BNN architecture can be extended to TNNs with practically no overhead, and in sec. IV the corresponding benefits in terms of neural network accuracy.

### III. THE OPERATION OF A PRECHARGE SENSE AMPLIFIER CAN PROVIDE TERNARY WEIGHTS

In this work, we use the architecture of [6], where synaptic weights are stored in a differential fashion. Each bit is implemented using two devices programmed either as low resistance state (LRS)/high resistance state (HRS) to mean weight +1 or HRS/LRS to mean weight -1. Fig. 1 presents the test chip used for the experiments. This chip cointegrates 130nm CMOS and resistive memory in the back-end-of-line,

<sup>1</sup>In the literature, the name “Ternary Neural Networks” is sometimes also used to refer to neural networks where the synaptic weights are ternarized, but the neuronal activations remains real or integer [17], [18].

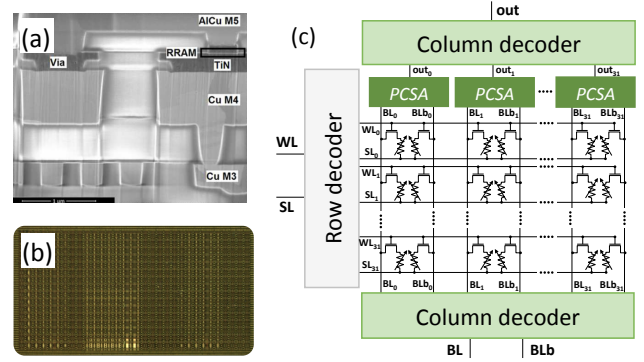


Fig. 1. (a) Electron microscopy image of a hafnium oxide resistive memory cell (RRAM) integrated in the backend-of-line of a 130nm CMOS process. (b) Photograph and (c) simplified schematic of the hybrid CMOS/RRAM test chip characterized in this work.

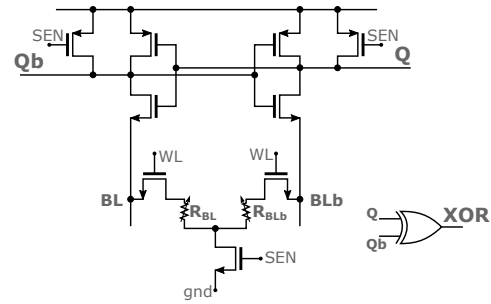


Fig. 2. Schematic of the precharge sense amplifier fabricated in the test chip.

between levels four and five of metal. The resistive memory cells are based on 10nm thick hafnium oxide (Fig. 1(a)). Our experiments are based on a kilobit array incorporating all sense and periphery circuitry (Fig. 1(b-c)). The synaptic weights are read using the onchip precharge sense amplifier (PCSA) presented in Fig. 2, initially proposed in [19]. Fig. 3(a) shows an electrical simulation of this circuit to explain its working principle. These simulations are presented in the commercial 130nm technology used in our test chip, with a near-threshold supply voltage of 0.6V [20] (the nominal voltage in this process is 1.2V). In a first phase ( $SEN=0$ ), the outputs  $Q$  and  $Qb$  are precharged to the supply voltage  $V_{DD}$ . In the second phase ( $SEN=V_{DD}$ ), each branch starts to discharge to the ground. The branch that has the resistive memory ( $BL$  or  $BLb$ ) with lowest electrical resistance discharges faster and causes its associated inverter to drive the output of the other inverter to the supply voltage. At the end of the process, the two outputs are therefore complementary, and can be used to tell which resistive memory has highest resistance and therefore the synaptic weight. We observed that the convergence speed of a PCSA depends heavily on the resistance state of the two resistive memories. This effect is particularly magnified when the PCSA is used in near-threshold operation, as presented here. Fig. 3(b) shows a simulation where the two devices  $BL$  and  $BLb$  were programmed in the HRS. We see that the two outputs converge to complementary values in 200ns, whereas

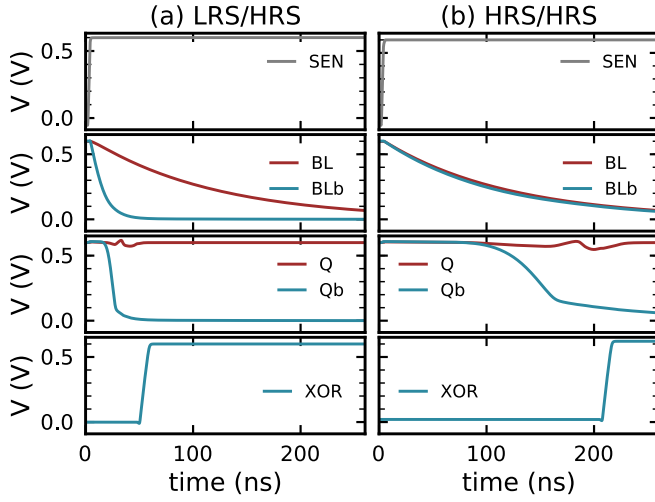


Fig. 3. Circuit simulation of the precharge sense amplifier of Fig. 2 with a near-threshold supply voltage of 0.6V, if the two devices are programmed in an (a) LRS / HRS (20k $\Omega$ /350k $\Omega$ ) or (b) HRS/HRS (320k $\Omega$ /350k $\Omega$ ) configuration.

only 50ns were necessary in Fig. 3(a), where the devices are programmed in complementary LRS/HRS states. These first simulations suggest a technique for implementing ternary weights using the memory array of our test chip. Similarly to when this array is used to implement BNN, we propose to program the devices in the LRS/HRS configuration to mean the synaptic weight 1, and HRS/LRS to mean the synaptic weight  $-1$ . Additionally, we use the HRS/HRS configuration to mean synaptic 0, while the LRS/LRS configuration is avoided. The sense operation is performed during a duration of 70ns. If at the end of this period, outputs Q and Qb have differentiated and the output of the XOR gate is therefore 1, output Q determines the synaptic weight (1 or  $-1$ ). Otherwise, output of the XOR gate is 0 and the weight is determined to be 0.

Experimental measurements on our test chip confirm that the PCSA can be used in this fashion. We first focus on one synapse of the memory array. We program one of the two devices (BLb) to a resistance of  $10^5\Omega$ . We then program its complementary device BL to several resistance values, and for each of them perform 100 read operations of duration 50ns, using an onchip PCSA operated in near-threshold. Fig. 4 plots the probability that the sense amplifier has converged during the read time. In 50ns, the read operation is only converged if the resistance of the BL device is significantly lower than 100k $\Omega$ . To evaluate this behavior in a wider range of programming conditions, we repeated the experiment on 109 devices and their complementary devices of the memory array programmed, each programmed 14 times, with various resistance values in the resistive memory, and performed a read operation in 50ns with an on-chip PCSA. Fig. 5 shows, for each couple of resistance value  $R_{BL}$  and  $R_{BLb}$  if the read operation was converged with  $Q = V_{DD}$  (blue), meaning a weight of 1, converged with  $Q = 0$  (red), meaning a weight of

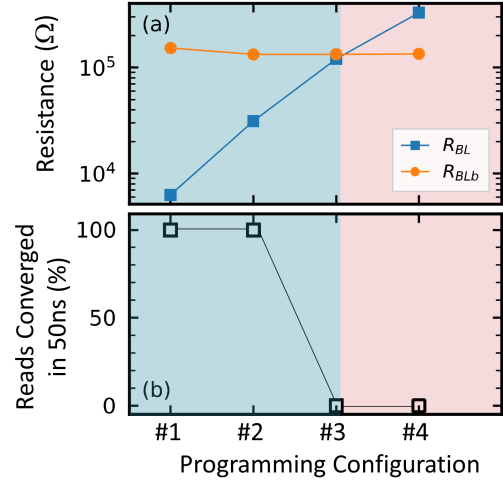


Fig. 4. Two devices have been programmed in four distinct programming conditions, presented in (a), and measured using an onchip sense amplifier. (b) Proportion of read operations that have converged in 50ns, over 100 trials.

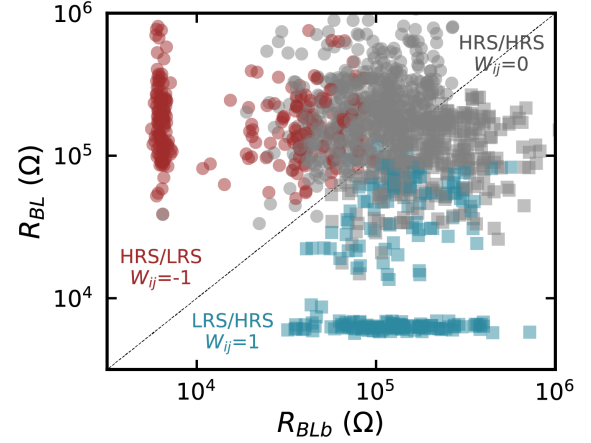


Fig. 5. For 109 device pairs programmed with multiple  $R_{BL}/R_{BLb}$  configuration, value of the synaptic weight measured by the onchip sense amplifier using the strategy described in body text and 50ns reading time.

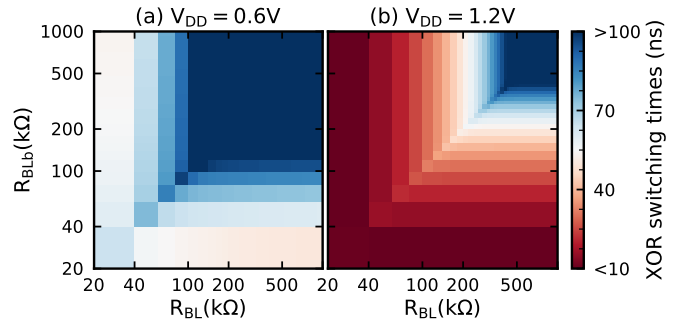


Fig. 6. Switching time of a precharge sense amplifier, extracted from circuit simulations using the design kit of a 130nm commercial technology, as a function of the resistance of the BL and BLb complementary resistive memories. Simulations performed using (a) near-threshold 0.6V supply voltage and (b) nominal 1.2V supply voltage.

-1, or not converged (grey) meaning a weight of 0. The results confirm that LRS/HRS or HRS/LRS configurations may be used to mean weights 1 and -1, and HRS/HRS for weight 0. Relatively high values of the HRS should be targeted: the separation between the 1 (or -1) and 0 regions is not strict, and for intermediate resistance values we see that the read operation may or may not converge in  $50ns$ .

Extensive circuit simulations in the 130nm technology of our test chip allow to evaluate this behavior in the general case. Fig. 6 shows the switching time of the PCSA as a function of the resistance of the two resistive memories BL and BLb, with nominal supply voltage (1.2V) and near-threshold supply voltage (0.6V). We see that for both supply voltages, the HRS/HRS configuration leads to longer switching times. In our technology, HRS states are typically characterized by resistances above  $100k\Omega$ . We see that the operation in near-threshold exhibits a larger area of HRS/HRS values with a switching time above  $70ns$ , corresponding to a 0 state. This implies a more robust detection of the 0 state in near threshold, compliant with the HRS variability.

#### IV. NETWORK LEVEL BENEFITS OF TERNARIZATION

We now investigate the accuracy gain when using ternarized instead of binarized networks. We trained BNN and TNN versions of networks with VGG-type architectures [21] on the CIFAR-10 task of image recognition, consisting in classifying color images among ten classes. The architecture of our networks consists of six convolutional layers with kernel size three. The number of filters at the first layer is called  $N$  and is multiplied by two every two layers. Maximum-value pooling with kernel size two is used every two layers and batch-normalization [22] every layer. The classifier consists of one hidden layer of 512 units. For the TNN, the activation function has a threshold  $\Delta = 5 \cdot 10^{-2}$  (as defined in section II). BNNs were trained following the methodology described in the Appendix of [23] and adapted from [3]. TNNs were trained with the methodology introduced in [2]. The training is performed using AdamW optimizer [24], [25] with minibatch size 128 and learning rate schedule used in [25], [26], resulting in a total of 700 epochs. Data is augmented using random horizontal flip, and random choice between cropping after padding and random small rotations.

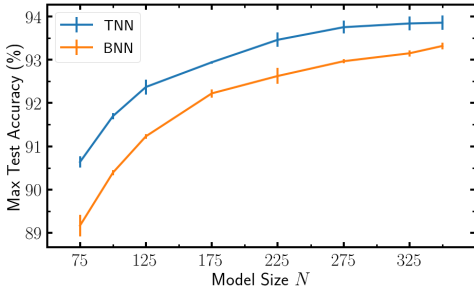


Fig. 7. Maximum Test Accuracy reached during one training, averaged over five runs, for BNNs and TNNs with various model sizes on the CIFAR-10 dataset. Error bar is one standard deviation.

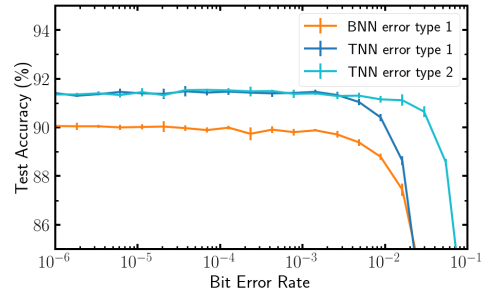


Fig. 8. Impact of Bit Error Rate on the Test Accuracy at inference time for model size  $N = 100$  TNN and BNN. Type 1 errors are sign switches (e.g. +1 mistaken for -1) and Type 2 errors are 0 mistaken for +1 or -1, or -1 and +1 mistaken for 0.

Fig. 7 shows the maximum test accuracies for different sizes of the model. TNNs always outperform BNNs with the same model size (and same number of synapses). The largest difference is seen for smaller model size, but a significant gap remains even for large models. Besides, the difference in the number of parameters required to reach a given accuracy for TNNs and BNNs increases with higher accuracies. There is therefore a clear advantage to use TNNs instead of BNNs.

We then investigate the impact of bit errors in BNNs and TNNs to see if the advantage provided by TNNs remains when errors are taken into account. Two types of errors are investigated: Type 1 errors are sign switches, while Type 2 errors are only defined for TNNs and correspond to 0 mistaken for +1 or -1, and +1 or -1 mistaken for 0. Type 1 errors are less likely than type 2 errors thanks to the differential reading scheme 2T2R, as seen in Fig. 5. Fig. 8 shows the impact of errors in the test accuracy for different values of the Bit error rate, averaged over five runs. Though Type 2 errors are more likely to occur with TNNs, their effect is not as serious as Type 1 errors as the drop in accuracy for Type 2 errors occurs for bit error rates one order of magnitude higher than for Type 1 errors. Therefore TNNs still outperform BNNs when device imperfections are included.

#### V. CONCLUSION

In this work, we revisited a differential compute in-memory architecture for BNNs. We showed experimentally on a hybrid CMOS/RRAM chip, and by circuit simulation that, its sense amplifier is able to differentiate not only the LRS/HRS and HRS/LRS states, but also the HRS/HRS states. This allows the architecture to store ternary weights, and to provide a building block for TNNs. We showed by neural network simulation on the CIFAR-10 tasks the benefits of using ternary instead of binary networks, and the high resilience of TNNs to weights errors. As this behavior is magnified in the slow but low power near-threshold operation regime, our approach specially targets extremely energy-conscious applications such as uses within wireless sensors or medical applications. This work opens the way for increasing the edge intelligence in such contexts, and also highlights that near-threshold operations of circuits may sometimes provide opportunities for new functionalities.

## REFERENCES

- [1] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi, "Scaling for edge inference of deep neural networks," *Nature Electronics*, vol. 1, no. 4, p. 216, 2018.
- [2] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.
- [3] 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," *arXiv preprint arXiv:1602.02830*, 2016.
- [4] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. ECCV*. Springer, 2016, pp. 525–542.
- [5] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Advances in Neural Information Processing Systems*, 2017, pp. 345–353.
- [6] M. Bocquet, T. Hirtzlin, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "In-memory and error-immune differential ram implementation of binarized deep neural networks," in *IEDM Tech. Dig. IEEE*, 2018, p. 20.6.1.
- [7] S. Yu, Z. Li, P.-Y. Chen, H. Wu, B. Gao, D. Wang, W. Wu, and H. Qian, "Binary neural network with 16 mb rram macro chip for classification and online training," in *IEDM Tech. Dig. IEEE*, 2016, pp. 16–2.
- [8] E. Giacomini, T. Greenberg-Toledo, S. Kvatinisky, and P.-E. Gaillardon, "A robust digital rram-based convolutional block for low-power image processing and learning applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 643–654, 2019.
- [9] X. Sun, S. Yin, X. Peng, R. Liu, J.-s. Seo, and S. Yu, "Xnor-rram: A scalable and parallel resistive synaptic architecture for binary neural networks," *algorithms*, vol. 2, p. 3, 2018.
- [10] Z. Zhou, P. Huang, Y. Xiang, W. Shen, Y. Zhao, Y. Feng, B. Gao, H. Wu, H. Qian, L. Liu *et al.*, "A new hardware implementation approach of bnns based on nonlinear 2t2r synaptic cell," in *2018 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2018, pp. 20–7.
- [11] M. Natsui, T. Chiba, and T. Hanyu, "Design of mtj-based nonvolatile logic gates for quantized neural networks," *Microelectronics journal*, vol. 82, pp. 13–21, 2018.
- [12] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on rram," in *Proc. ASP-DAC*. IEEE, 2017, pp. 782–787.
- [13] J. Lee, J. K. Eshraghian, K. Cho, and K. Eshraghian, "Adaptive precision cnn accelerator using radix-x parallel connected memristor crossbars," *arXiv preprint arXiv:1906.09395*, 2019.
- [14] H. Alemdar, V. Leroy, A. Prost-Boucle, and F. Pétrot, "Ternary neural networks for resource-efficient ai applications," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2547–2554.
- [15] L. Deng, P. Jiao, J. Pei, Z. Wu, and G. Li, "Gxnor-net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework," *Neural Networks*, vol. 100, pp. 49–58, 2018.
- [16] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli, "On-chip error correcting techniques for new-generation flash memories," *Proc. IEEE*, vol. 91, no. 4, pp. 602–616, 2003.
- [17] N. Mellempudi, A. Kundu, D. Mudigere, D. Das, B. Kaul, and P. Dubey, "Ternary neural networks with fine-grained quantization," *arXiv preprint arXiv:1705.01462*, 2017.
- [18] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. Ong Gee Hock, Y. T. Liew, K. Srivatsan, D. Moss, S. Subhaschandra *et al.*, "Can fpgas beat gpus in accelerating next-generation deep neural networks?" in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*. ACM, 2017, pp. 5–14.
- [19] W. Zhao, C. Chappert, V. Javerliac, and J.-P. Noziere, "High speed, high stability and low power sensing amplifier for mtj/cmos hybrid logic circuits," *IEEE Transactions on Magnetics*, vol. 45, no. 10, pp. 3784–3787, 2009.
- [20] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proc. IEEE*, vol. 98, no. 2, pp. 253–266, Feb 2010.
- [21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.
- [23] T. Hirtzlin, B. Penkovsky, M. Bocquet, J.-O. Klein, J.-M. Portal, and D. Querlioz, "Stochastic computing for hardware implementation of binarized neural networks," *IEEE Access*, vol. 7, p. 76394, 2019.
- [24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [25] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," *arXiv preprint arXiv:1711.05101*, 2017.
- [26] —, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.