In-Memory Computing for Machine Learning Using Pulse-Width Modulated Word Lines and Error Adaptive Classifier Boosting

Justin Doong, Kourosh Hakhamaneshi, and Alexis Moreno

***Abstract*—While machine learning has been a tremendously powerful tool in many applications, its data-centric nature has resulted in memory access dominating power consumption. Recently, there has been work on in-memory computing, a paradigm that can significantly reduce energy consumption but can also introduce reliability concerns due to device variations. This work proposes a simple topology for implementing in-memory computing for classification while mitigating the effects of voltage threshold variations and allowing for reduced memory size for equivalent accuracy.**

I. Introduction

In many cases, training a machine learning (ML) model can take several hours or even longer and thus is typically performed in the cloud. Inference, on the other hand, can be performed either in the cloud or at the edge (*e.g.*, IoT or mobile). Many applications require ML inference to be performed close to the sensors because of communication bandwidth limitations. For example, biomedical diagnostics often need to be performed right on the sensor rather than in the cloud.

The basic functions that need to be performed during inference are multiply and accumulate (MAC), $f(\Sigma_i w_i x_i)$, and a non-linear clipping function which determines the class (*e.g.*, $\mathrm{sgn}(x)$). Typically, fetching data in and out of memory is the bottleneck for energy consumption as ML models require a lot of data to be communicated between memory and the processor unit. For example, in $45\,\mathrm{nm}$ CMOS, a 16-b word demands 20-$100\,\mathrm{pJ}$ per access from $32\,\mathrm{kB}$ to $1\,\mathrm{MB}$ memory versus $1\,\mathrm{pJ}$ per multiply [1]. While there are a couple of approaches both in hardware and software to reduce this data movement [2], a lot of attention has been given to in-memory computing (IMC). In this approach, instead of fetching weights out of the memory, inputs are applied to the memory, and classification results are directly taken from the output.

The ML community has shown that if the weights are constrained to ± 1, there is minimal loss in accuracy, provided the model is trained properly [3]. Almost all of the recent work on IMC has leveraged this binary representation of weights.

The fundamental idea is that the model is trained outside of the chip/SRAM with co-optimization between hardware and algorithm in mind (*e.g.*, binary weighted networks, or BWN), and then the learned weights are loaded onto the SRAM to do the inference task. Then, digital input features (*i.e.*, sensory data) are fed into the SRAM and converted to an analog representation which cause a MAC operation (either in current or voltage domain) to take place, and then the sense amplifiers (SA) clip this accumulation to perform classification.

The main challenge is that since the computation is mainly in the analog domain, it is very susceptible to random variations (*e.g.*, threshold voltage V_{th} variation of bit cells). It is worth noting that some ML algorithms are immune to such variations if these variations are taken into account while training (*e.g.*, Adaboost [4]). There have been hardware approaches which have leveraged this immunity to perform accurate computations (*e.g.*, EACB [5])

The remainder of this paper is organized as follows. Section II provides an overview of the state of the art and the different approaches. Section III presents the approach in this work. Section IV describes the experimental setup. Finally, Section V concludes this paper.

II. State of the Art

There have been demonstrations of a few different approaches to IMC using mostly standard SRAM cells [6], [7], and [8]. The approaches all involve analog word-line driving to represent inputs, and all demonstrate significant energy savings over conventional architectures. A comparison between different works is presented in Table I

A. Amplitude Modulated Word Lines

In the system demonstrated in [6], computation is performed in-place by the bit cells of an SRAM, with word-line DACs (WLDACS) translating input sensory data to analog voltages driving the word lines (WL). This approach faces circuit non-idealities, especially high variability in bit cells, which degrades the quality of outputs. Fig. 1 illustrates this architecture.

This system operates in two modes: 1) SRAM mode, or standard read and write, and 2) classify mode, in which the WLs are driven by the analog representation of the input feature vector. In SRAM mode, only one WL is a driven at a time while multiple WLs are driven simultaneously in classify mode. Each column is a weak classifier, and EACB [5] is used to overcome the fitting errors of the weak classifiers as well as the non-idealities and variabilities of the circuit. The operation is as follows. First, the bit lines (BLs) (and BLBs) are pre-charged. Then all the WLs are driven at once to analog voltages corresponding to the elements of the inputs X (*i.e.*, the features); this is done via the peripheral WLDACs. Thus, each bit cell pulls a current modulated by its WL voltage from either BL or BLB, depending on its stored state. This approximates multiplication by ± 1. The bit-cell currents from the column then sum, as in an inner product, discharging BL or BLB. Finally, a comparator provides sign thresholding of the differential signal.

Fig. 2 shows the WLDAC circuit and simulated output (WL) waveform. Digital features are provided as five bits $X[4{:}0]$ to select binary-weighted PMOS current sources. An additional PMOS current source is included to enhance the linearity of the charge drawn by a bit cell. The current from the PMOS sources is used to bias a bit-cell replica to drive the WL. The replica consists of a switch $\mathrm{MD_R}$ and a diode-connected transistor $\mathrm{MA_R}$. Thus, all bit cells in the row provide a modulated current I_{bc} (scaled by the replica sizing ratio). Note that with all WLs driven this way, a potentially large number of bit cells drive a single BL/BLB at the same time (128 in the prototype). To not saturate discharge all of the BL/BLB capacitances, Q_{bc} of each is designed to be low ($< 10fC$), requiring WL voltages $< 0.4V$.

The BLs can have large swings, and since this affects the amount of current sourced or sinked by the bit cells as shown in Fig. 3, at lower common-mode voltages the difference between BL and BLB will get smaller. This makes it harder to design the comparator, as the offset of the comparator directly affects the decision made. This system utilizes a rail-to-rail comparator. A method of offset cancellation using the 6T SRAM array structure is also employed.

The demonstrated system achieved the ideal accuracy after 18 iterations, meaning that while using EACB, it needs 18 times to go over all weak classifiers to achieve the accuracy needed. This system achieved 113X lower energy than a discrete system with a standard training algorithm and 13X lower energy than a discrete system with the proposed training algorithm.

B. *Pulse-Width Modulated Word Lines*

The use of pulse-width modulated word line (PWM-WL) signals to read multiple rows simultaneously was proposed in [7], as shown in Fig. 4. There are key

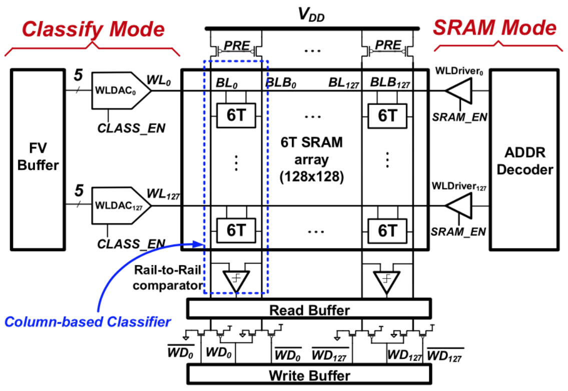

Fig. 1. Architecture of in-memory classifier in [6], showing periphery for two modes of operation.

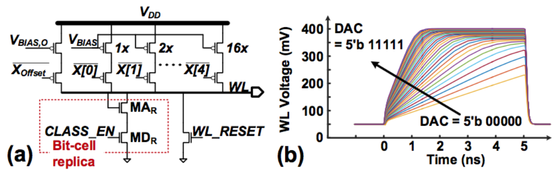

Fig. 2. (a) Current mode WLDAC in [6]. (b) Transient WL pulses with variable settling time for different input codes.

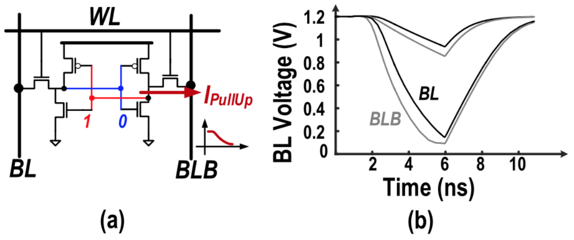

Fig. 3. (a) Nonlinearity arising from possible pull up condition in bit cells. (b) Simulation showing variable compression of nominally equivalent BL/BLB differential voltage due to different common mode levels.

similarities between this approach and the amplitude-modulated WL (AM-WL) scheme proposed in [6]; both perform multi-bit read accesses/operations by first pre-charging the BLs/BLBs and then driving WLs with pseudo-analog signals that correspond to inputs (*e.g.*, features for classification). In [6], the WL amplitude was used to encode information, while in [7], WL pulse-width was used. For a given column, driving the WLs with PWM signals results in a BL voltage drop proportional to the weighted sum of the bits (*e.g.*, weights) stored along the column. The longer the pulse-width, the more charge is delivered for the given current, which itself is just set by the bit stored in the SRAM cells and the voltage of the word-line. This contrasts with the

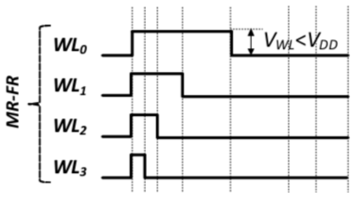

Fig. 4. PWM-WL approach to analog representation of inputs for IMC architecture [7].

mechanism in [6], in which the current itself varies. For example, to perform a dot product for an ML algorithm such as support vector machine (SVM), binary weights are stored in the SRAM cells while the pulse-width of the WLs are varied to represent the input data. A single column, with the various inputs from the various pulse-width modulated word lines, would produce a single dot product.

Relative to the AM-WL approach, the PWM-WL scheme is more immune to V_{th} variations because the WLs are driven at a fixed voltage. However, the PWM-WL still has significant limitations related to throughput, power efficiency, and robustness. The PWM-WL pulse width is limited by the BL RC time constant for linearity and read stability reasons. In [7], the pulse width was chosen to be less than 40% of the time constant. Moreover, just like in [6], the WL voltage V_{WL} cannot swing the full voltage range in order to prevent destructive read. To aid SRAM read given short pulse-width requirements, sub-ranged read was used in [7].

The proposed architecture in [7] also had additional features that enabled increased versatility. This included addition and subtraction with a replica bit-cell array as well as bit-line and cross bit-line processing to perform Manhattan distance as well as dot product operations. However, the decision to include reconfigurability may have resulted in non-optimal architectural choices given a specific application. When compared to a conventional architecture, the approach in [7] performed 16 times fewer read accesses and had 5.8 times greater throughput while still maintaining greater than 90% accuracy. This resulted in approximately an order of magnitude improvement in energy consumption over conventional architectures as shown in Fig. 5. However, the energy-delay product (EDP) and memory size were much greater than other IMC approaches [6].

C. Pulse-Width Modulated Bit Lines

Another approach using pulse-width modulated bit lines (PWM-BL) was presented in [8]. This system implemented a BWN convolutional neural network (CNN) using 10T SRAM, which can read and write separately. This approach was taken in order to mitigate the variance of a transistor's threshold voltage and reduce the required

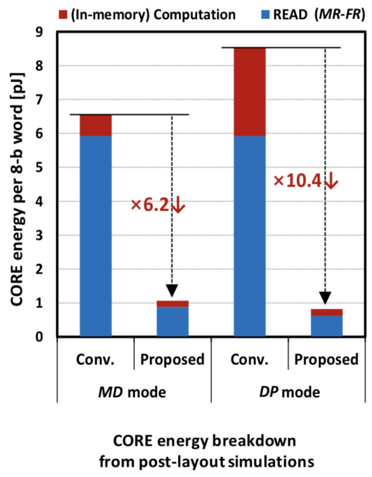

Fig. 5. Energy consumption comparison between the PWM-WL and conventional architectures [7].

memory size. The 256x64 SRAM architecture, Fig. 6, divides its rows into 16 local arrays each with 16 rows and 64 inputs.

The operation of this architecture is as follows:

1) DAC: The global bit lines are charged to V_{a} by converting the X_i 6-bit input to an analog voltage through the column-wise DAC.
2) Multiply and average (MAV): The read word line is set high and the V_{a} and W_{i} are multiplied. The first phase of averaging takes place concurrently on a local bit line. Adjacent columns perform the second phase of averaging within the local array when the ENp is high, which connects all the columns' BLs and BLBs with their respective type within a local array resulting in V_{pavg} and V_{navg} as shown in Fig 7.
3) ADC: Each local array's V_{pavg} and V_{navg} are fed into an ADC the produces a digital 7-bit output.

Drawbacks of this work are the required bigger 10T SRAM cells, the complexity introduced by the ADC's offset voltage, small memory capacity, and the additional stage to scale the output off chip. However, to the best of the authors' knoweledge this implementation has the highest energy efficiency published, $28.1\,\mathrm{TOPS/W}$.

III. Proposed Approach

In order to optimize the power, throughput, and area of IMC for ML using standard SRAM cells, this paper focuses on a particular application, namely classification, which employs dot product operations. This contrasts with the approach taken in [7], which emphasized versatility. This paper proposes an IMC architecture employing PWM-WL to deliver inputs while using cross bit-line weighting and summing to produce a strong classifier from multiple weak classifiers (*i.e.*, Adaboost).

TABLE I
COMPARISON OF STATE OF THE ART [8]

	Tech. (nm)	Algo., # Classes	Comp. Mode	Class. Accuracy	Input, Filter	SRAM Size (KB)	Throughput (GOPS)	Energy Efficiency (TOPS/W)
ISSCC '18 [8]	65	CNN, 10	Analog	96%	7b, 1b	2	10.7	28.1
ISSCC '16 [9]	65	CNN, 10	Digital	98.3%	16b	36	64	1.42
VLSI '16 [10]	40	CNN, 10	Digital	98%	6b, 4b	144	102	1.75
arXiv '16 [7]	65	kNN, 4	Analog	92%	8b	16	10.2	0.98

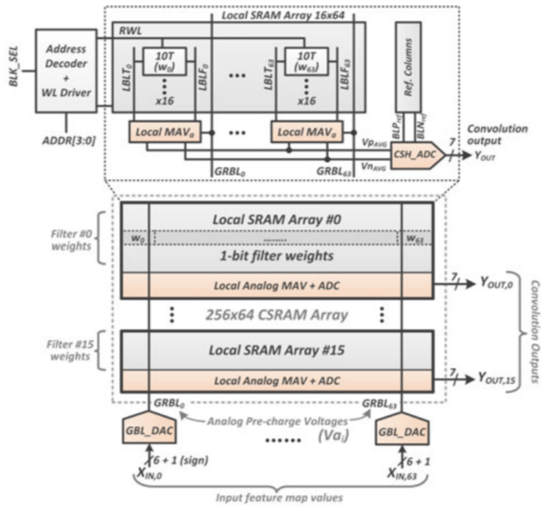

Fig. 6. Architecture of PWM-WL approach [8].

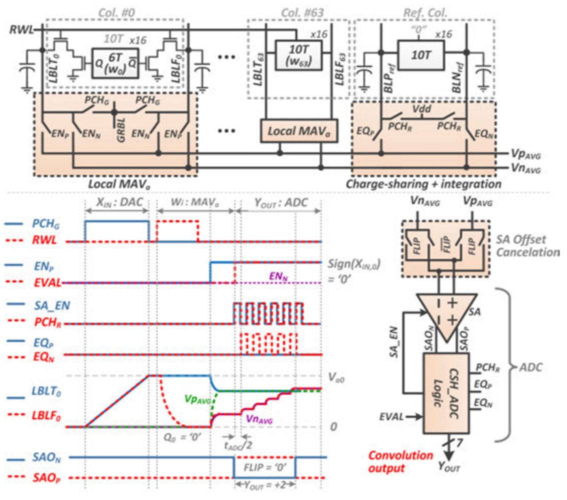

Fig. 7. Operation of architecture [8].

A. Pulse-Width Modulated Word Lines and Error Adaptive Classifier Boosting

PMW-WL is a promising analog representation of inputs/classification features for IMC because of its greater immunity to V_{th} variations. It is also a relatively simple approach that does not require extensive overhead to implement. However, as detailed above, it still suffers from voltage-swing limitations due to linearity, read-stability, and write disturb concerns. Nevertheless, this approach could potentially provide a significant improvement over amplitude-based WL driving approaches. This paper seeks to pair PWM-WL with the EACB approaches of [4] and [6] to maintain the accuracy of the IMC architecture despite bit-cell variations and the use of binary weights. The idea is that the decreased susceptibility to variations of the PWM-WL approach will increase the overall normalized accuracy of the system, in other words, reduce the required array size and thus also energy to achieve the same high accuracy.

IV. Experimental Setup

The primary focus of this work is to determine whether (and to what extent) using PWM-WL results in reduced susceptibility to variations. The approach using AM-WL will serve as the point of reference. The first step will be to prepare a standard SRAM array with custom analog WL driving via Cadence Virtuoso. Then the general behavior of the SRAM array across variations (generated by Monte Carlo simulations) will be examined for the PWM-WL and AM-WL approaches. The next step will be to evaluate and compare the performances of the two approaches for the Adaboost classification algorithm. The approaches will be evaluated using standard, system-level metrics such as accuracy, power, throughput, and area. The performance of each system will then be evaluated using a figure of merit introduced in [8].

V. Conclusion

Recent work on IMC has demonstrated significantly reduced power consumption for ML applications by reducing the frequency and number of memory accesses. However, a major hurdle has been susceptibility to variations. The proposed approach combines existing methods (PWM-WL and EACB) to improve the robustness and thus accuracy of IMC while minimizing complexity and overhead.

References

[1] M. Horowitz, "Computings energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2014, pp. 10–14.

[2] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," in *Proc. of the IEEE*, vol. 105.

[3] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *arXiv*, Nov. 2015. [Online]. Available: https://arxiv.org/abs/1511.00363

[4] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. Cambridge, MA, USA: MIT Press, 2012.

[5] Z. Wang, R. E. Schapire, and N. Verma, "Error adaptive classifier boosting (eacb): Leveraging data-driven training towards hardware resilience for signal inference," vol. 62.

[6] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6t sram array," vol. 52.

[7] M. Kang, S. Gonugondla, A. Patil, and N. Shanbhag, "A 481pj/decision 3.4m decision/s multifunctional deep in-memory inference processor using standard 6t sram array," *arXiv*, Oct. 2016. [Online]. Available: https://arxiv.org/pdf/1610.07501.pdf

[8] A. Biswas and A. P. Chandrakasan, "Conv-ram: An energy-efficient sram with embedded convolution computation for low-power cnn-based machine learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 488–489.

[9] e. a. J. Sim, "A 1.42tops/w deep convolutional neural network recognition processor for intelligent ioe systems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2016, pp. 264–265.

[10] e. a. B. Moonns, "A 0.32.6 tops/w precision-scalable processor for real-time large-scale convnets," in *IEEE Symp. VLSI Circuits*, 2016.