Designing Circuits for AiMC Based On Non-Volatile Memories: A Tutorial Brief On Trade-Offs and Strategies for ADCs and DACs Co-Design

# Designing Circuits for AiMC Based on Non-Volatile Memories: a Tutorial Brief on Trade-offs and Strategies for ADCs and DACs Co-design

| | |
|---|---|
| Journal: | *IEEE Transactions on Circuits and Systems II: Express Briefs* |
| Manuscript ID | TCAS-II-18912-2023.R1 |
| Manuscript Type: | Special Issue on Tutorial Briefs 2023 |
| Date Submitted by the Author: | 29-Nov-2023 |
| Complete List of Authors: | Vignali, R.; Universita degli Studi di Pavia Facolta di Ingegneria<br>Zurla, R.; STMicroelectronics SRL<br>Pasotti, Marco; STMicroelectronics SRL<br>Rolandi, P. L.; STMicroelectronics SRL<br>Singh, A.; IBM Research Europe - Zurich<br>Le Gallo, Manuel; IBM Research Europe - Zurich<br>Sebastian, Abu; IBM Research Europe - Zurich<br>Jang, Taekwang; ETH Zurich Foundation<br>Antolini, Alessio; Universita degli Studi di Bologna Scuola di Ingegneria<br>FRANCHI SCARSELLI, ELEONORA; Universita degli Studi di Bologna Scuola di Ingegneria<br>Cabrini, Alessandro; Universita degli Studi di Pavia Facolta di Ingegneria |
| EDICS: | ACS - Analog and Mixed Mode Circuits and Systems, ACS320 - Digital to analog converters < ACS - Analog and Mixed Mode Circuits and Systems, ACS330 - Analog to digital converters < ACS - Analog and Mixed Mode Circuits and Systems |
| TCAS-II Subject Category<br>Please select the subject category that most closely fits with the scope of your manuscript: | Analog and Mixed Mode Circuits and Systems |
| | |

# Designing Circuits for AiMC Based on Non-Volatile Memories: a Tutorial Brief on Trade-offs and Strategies for ADCs and DACs Co-design

R. Vignali, R. Zurla, M. Pasotti, P. L. Rolandi, A. Singh, M. Le Gallo, A. Sebastian, T. Jang, A. Antolini, E. Franchi Scarselli, and A. Cabrini

***Abstract*—Analog In-Memory Computing (AiMC) based on Non-Volatile Memories (NVM) is a promising candidate to reduce latency and power consumption of neural network (NN) inference in edge-computing applications. This kind of computational accelerators allows both storing weights and performing in-situ analog computation inside the array. This tutorial explores trade-offs and strategies in the design of DACs and ADCs for this kind of systems, highlighting the strong interdependence between the two converters. Starting from an analysis of input and weights encoding techniques this tutorial will then propose a discussion aiming at exploring critical aspects that constrain the design of D-A and A-D converters drawing some co-design considerations.**

***Index Terms*—AiMC, Non-Volatile Memories, NVM, Artificial Intelligence, DAC, ADC, MAC operation.**

## I. Introduction

In the last few years, applications of Artificial Intelligence, AI, spread and surged at an exponential rate. For this reason, the availability of highly efficient hardware-accelerators for on-silicon implementation of AI algorithms is becoming one of the key challenges for the next years especially considering the endless growth of edge computing applications.

In this respect, the feasibility of AiMC was demonstrated based on different memory arrays including both volatile (as SRAMs [1]–[3], and DRAMs [4], [5]) and non-volatile memories (NVMs) [6]–[13]. Among NVMs, resistive switching (RS) devices such as phase-change memory (PCM) and metal-oxide-based resistive random access memory (RRAM) have been considered as primary candidates for next AiMC.

The key advantages of AiMC implemented on NVM devices are: the convenient exploitation of basic Ohm's law together with Kirchhoff's laws (i.e., current and voltage laws, KCL and KVL, respectively) to efficiently implement the analog Matrix-Vector-Multiplication, MVM, operation as $Y = AX$, where $A$ is the NVM matrix storing the weights of the neural network, while $X$ and $Y$ are the input and the output vector, respectively; the non-volatility which allows to maintain weights through power cycling; the excellent scalability of the unit cell that could ideally lead, in a crossbar array implementation, to an impressively low $4F^2$ cell footprint ($F$ being the technology half pitch [14]); the possibility of storing a given number of different discrete weights in a single cell by modulating the resistance of matrix $A$ elements with suitable multilevel, ML, programming algorithms [10], [15]–[17].

In this frame, the analog elements $a_{i,j}$ of matrix A (which is organized as an array with a given number of word-lines, WLs, and bit-lines, BLs, so that $1 < i < \#WL$ while $1 < j < \#BL$) are mapped onto RS matrices as conductance values, $g_{i,j}$, while the elements, $x_{D,i}$, of a (digital) input vector, $X_D$, (coming form higher AI hierarchy levels) must be encoded to get the elements $x_{i,j}$ of the analog vector, $X$, using suitable digital-to-analog-converters (WL-DACs) (see Fig. 1).

Analogously, vector $Y$, which includes the analog signals, $y_i$, generated as outputs of the MVM (generally currents or voltages, as implied by Ohm's Law, KCL and KVL) must be sensed and digitized by means of suitable analog-to-digital converters (BL-ADCs) to get their digital equivalent $Y_D$ (to be sent to higher AI hierarchy levels).

It is well recognized that WL-DACs and BL-ADCs represent intrinsic bottlenecks of the entire system (representing more than 60% contribution in term of area and power [18]) and, for this reason, their design is one of the most critical and challenging issues of AiMC systems. A key aspect in this respect is represented by the strong interdependence of the two elements. Indeed, the architectures of WL-DACs and BL-ADCs cannot be chosen independently. The optimization

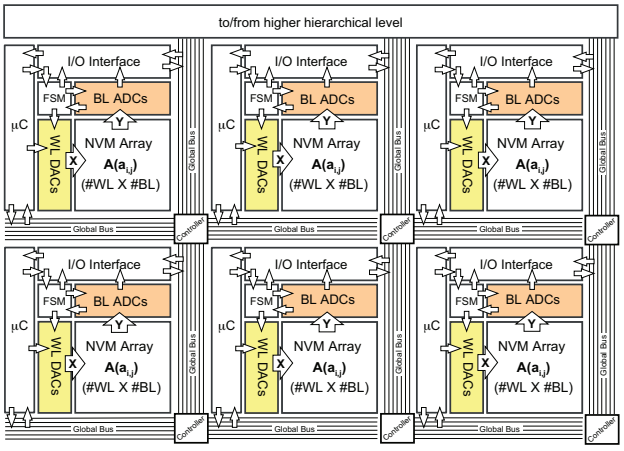


Fig. 1. Simplified architecture of an AiMC core. The figure shows a system including 6 NVM arrays $A$ each one being the core of a single *tile* which includes BL-ADCs, WL-DACs, FSM and microcontroller, and all the circuits required to manage the array A. The number of tiles of the system are normally chosen considering the specifications of AI application, efficiency in terms of power and speed and silicon area occupation.

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received April 19, 2021; revised August 16, 2021.

of the system performance requires co-design strategies that are needed to meet the stringent targets, generally expressed as TOPs/W, under severe constraints in terms of silicon area, computational efficiency, accuracy and speed. Indeed, by following for example an input-based design strategy, the first aspect to be addressed is the encoding of inputs. Once the encoding has been defined, the design space for possible DAC architectures gets limited to some specific solutions. Among these possible structures, which strongly impacts on the generation of the signals that will be provided to the cascaded ADCs, depending on design constraints on power consumption, speed, and area, the most convenient ADC architecture may be identified. Similarly, alternative solutions could be conceived by adopting, for example, a complementary design approach (output-based).

The purpose of this tutorial brief is to address most of the aspects that designers of circuits for AiMC hardware accelerators based on NVMs have to consider for an optimal design of the system. In particular, the tutorial will review, analyze and summarize the trade-offs and the inter-dependencies that exist and arise between encoding of variables (e.g., input/output vectors $X$ and $Y$), array architecture, DAC and ADC architectures, performance of MVM operation (in terms of speed and accuracy), power consumption and silicon area occupation. The focus will be on large NVM arrays (i.e., for $\#WL \times \#BL > 1M$) that allow to store large kernels minimizing the impact of peripheral circuit on the system area. The final goal is to provide the reader useful considerations to obtain design guidelines and co-design strategies to address the complex procedures that are involved in the definition of the architecture of an AiMC system regarding, in particular, WL-DACs and BL-ADCs.

## II. Weights & Input Vector Encoding

Before discussing the implementation of WL-DACs and BL-ADCs it is necessary to address the way weights and inputs can be encoded into a physical analog quantity.

### A. Weights: Cell and Array Biasing

As previously mentioned, the weights $a_{i,j}$ of matrix $A$, are mapped as conductance value $g_{i,j}$. From the simple application of the Ohm's law, a conductive element $g_{i,j}$ could be used to drain a current $i_{i,j}$ by imposing a fixed voltage $V$ (i.e., $i_{i,j} = g_{i,j}V$) or, vice versa, to generate a voltage $v_{i,j}$ by biasing the element with a fixed current $I$ (i.e., $v_{i,j} = r_{i,j}I$ where $r_{i,j} = 1/g_{i,j}$). To better take advantage of the array structure, the preferred way for implementing analog MVM is to impose a fixed voltage, $V_{bl}$, to each memory cell belonging to the same BL (thus connecting in parallel all the cells of the BL) obtaining the analog products $x_i \cdot a_{i,j}$ as currents that are then summed over the BL (KCL approach) to get $I_{bl}$ (as shown in Fig. 2a). The complementary approach, which requires biasing with a fixed current $I_{bl}$ all the cells, connected in series, of a single BL (KVL approach), is commonly discarded due to the increase in terms of complexity and area of the array, cell non-linearity, unpractical range of the generated output voltage, $V_{bl}$, especially when dealing with large arrays. Another important aspect, that will be discussed in the next section, is about the need of encoding signed weights $a_{i,j}$ that must be implemented despite the inherently non-negative values of conductances $g_{i,j}$.

### B. Multi-bit Input Encoding

The $b_X$-bit elements $x_{D,i}$ of (digital) input vector $X_D$ can be encoded to get the analog $x_i$ of $X$ as either voltages $v_i(x_i)$ or voltage pulses with fixed amplitude $v_{WL}$ and variable time duration $t_i(x_i)$. As can be seen in Fig. 2b, the elementary cell topology allows providing inputs $x_i$ (required to complete the biasing of the $g_{i,j}$) equivalently through the WLs (gates of $T_{i,j}$) or the SLs (sources of $T_{i,j}$) biasing the other terminal with a suitable constant voltage (thus providing a $v_{gs,i}$ voltage to transistors $T_{i,j}$). The problem is that the I-V characteristic of the cell suffers from non-linearity due to the presence of the resistive elements [19] and select transistors $T_{i,j}$. As discussed above, if we bias the BL with a fixed $V_{bl}$ (KCL approach), it is possible to express $I_{bl}$ as:

$$I_{bl} = V_{bl} \sum_{i=1}^{\#WL} \frac{1}{1/g_{i,j} + r_{on_{i,j}}} \tag{1}$$

where

$$g_{i,j} = f_{NL}(V_{g_{i,j}}) \text{ and } V_{g_{i,j}} = \frac{V_{bl}}{1 + g_{i,j} \cdot r_{on_{i,j}}} \tag{2}$$

$r_{on_{i,j}}$ being the equivalent on-resistance of $T_{i,j}$ (which is a function of $v_{gs,i}$, and, hence, of $x_i$). From (1), it is evident that a variation in the WL and/or SL voltage, determined by the different values of $v_i(x_i)$, affects the I-V response of the cell that will thus depend also on the value of $x_i$. For this reason encoding the input vector as a voltage amplitude $v_i(x_i)$ is not convenient in this kind of application and the voltage-pulse approach is preferred. By following this approach there will be a time-varying BL current $I_{bl}(t)$ that will depend on how many cells are conducting at a given time instant $t$. Without losing generality, in the following we will assume the WLs as the terminals used to provide the different inputs $x_i$ as variable-length voltage pulses. It is worth mentioning that, by providing the voltage pulses through the WLs, problems related to IR drops are largely limited thus simplifying the design of the WL-DAC. Nonetheless driving the SLs could also be feasible depending on the memory array organization.

From above considerations, it is more convenient to encode the $x_{D,i}$ as voltage pulses with variable lengths $t_i(x_{D,i})$. A straightforward way of encoding inputs $x_{D,i}$ is to modulate the pulse width as being proportional to $x_{D,i}$, with a maximum duration $t_{max}$ of $2^{b_X} - 1$ unitary time-pulses $t_u$. We can identify this method as PWM (Pulse Width Modulation, as shown in Fig. 2c). Alternatively, the inputs $x_{D,i}$ could be separated into $b_X$ pulses, with fixed duration and amplitude equal to 0 or $V_{WL}$ depending on the value of the $b_X$ bits of $x_{D,i}$. We can refer to this method as BS (i.e., bit-serial or, alternatively, bit slicing, see Fig. 2d). Without entering in the details, it is important to note that, in a PWM scheme, the MAC (Multiply-and-Accumulate) operation of each MVM must be performed by integrating $I_{bl}$ over $t_{max}$ to get a charge

that is then converted by the BL-ADC while, in the BS case, the MAC operation can be performed by properly handling the BL-ADC in each time slot of the sequence of pulses taking into account the significance of the bits.

Choosing, among the two approaches, which is the most convenient is not straightforward and requires many considerations, especially on the effects that the encoding scheme has on time, accuracy, and energy required for the A-D conversion. For instance, in a PWM approach the conversion time, often called integration time, is given by:

$$t_{int,PWM} = t_{ch} + (2^{b_X} - 1) \cdot t_u \tag{3}$$

where $t_{ch}$ is a time interval required to charge WL, BL and SL at their correct voltage level and $t_u$ is the minimum duration of the pulse (e.g., the duration of the input LSB). In practice, $t_u$ is limited by the propagation delay of enable signals of the computation inside the memory tile and, hence, $t_u \propto \#WL^2$. In contrast, in the BS technique the conversion time is:

$$t_{int,BS} = b_X \cdot (t_{ch} + t_u). \tag{4}$$

Since the value of each bit of the input vector could be different, enough time $t_{ch}$ has to be guaranteed at each step to correctly set the voltages at input terminal. It can be easily shown that $t_{int,BS} > t_{int,PWM}$ when:

$$\alpha = \frac{t_{ch}}{t_u} > \frac{2^{b_X} - b_X - 1}{b_X - 1} \tag{5}$$

which shows that the BS approach becomes more advantageous when $b_X$ increases. As an example, assuming $\alpha = 4$, the BS approach becomes faster than PWM when $b_x \geq 5$.

The physical dimension of the array also plays a major role in determining which one of the two methods is more appropriate. In a AiMC architecture large arrays are desired for two main reasons: increasing $\#BL$ allows increasing parallelism (i.e., the number of computations that can be performed at the same time); increasing $\#WL$ allows to store larger kernels along a single BL and to diminish the area overhead of a single ADC. However, increasing $\#BL$ and/or $\#WL$ leads to longer routing and larger parasitics (resistances and capacitances) associated to WLs, BLs, and SLs which adversely affect the rising and falling times for driving these nodes and, hence, both $t_{ch}$ and $t_u$. It is also worth to point that $t_u$ could differ in the PWM and BS case (in BS the topology of the ADC, and the corresponding constraints, could indeed play an important role on the minimum time required to operate the DAC). While discussing the accuracy related to the input encoding, contrasting results emerged from previous analysis [20], [21]. However, the advantage in terms of precision of one over the other is not so remarkable and, for this reason, we can consider the two technique at least comparable [21].

### C. Signed Weights and Inputs and Array Organization

It is worth to mention that the propagated results of the hidden layers of a DNNs are usually non-negative (thanks to non-linear functions, i.e. ReLu [22]), whereas the use of positive and negative kernel coefficients is often exploited to perform inferences. For this reason, an interesting aspect to address, is the management of signed inputs and weights. Since it is not possible to implement negative values of $g_{i,j}$, different strategies have been proposed to counteract this physical limit. We are going to briefly review two approaches:

*a)* obtaining a signed quantity as the difference between two memory cells, one for the plus and one for the minus sign. This is achieved by separating positive and negative MAC integrations in two different space (Fig. 3a) [23] or time (Fig. 3b) locations. In other words this means storing positive and negative weights in two separate BLs (Fig. 3a) or WLs (Fig. 3b) and then performing two successive integrations. Alternatively, we can integrate the two BLs at the same time while subtracting the two contributions. The output difference can be performed directly in current in the analog domain or, after the A-D conversion, in the digital domain. In terms of energy efficiency the two approaches are equivalent. The time separation approach is expected to have better accuracy while the space-separation allows to save time by a factor of two or more but could suffer more from mismatches in the readout path (from BL to ADC input).

*b)* introducing an offset in the weight programming (Fig. 3c)

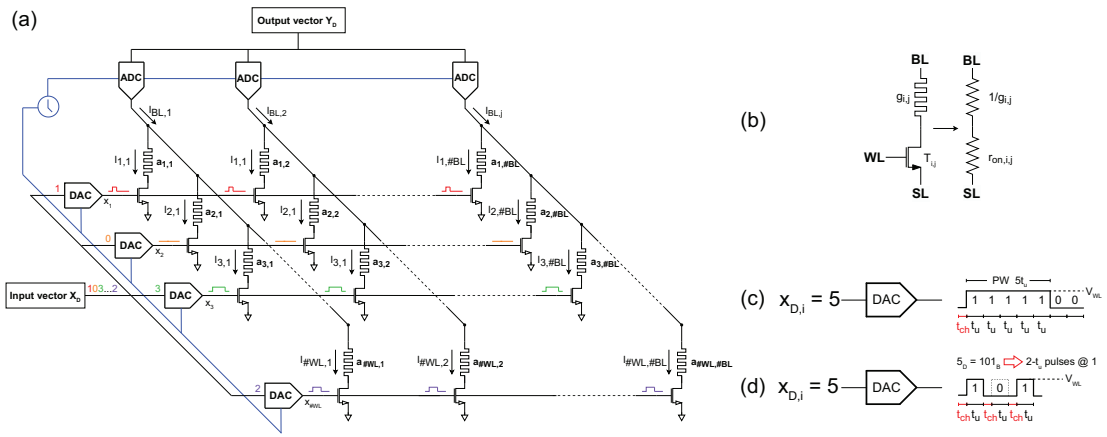


Fig. 2. An example of AiMC array architecture (a). Inputs are provided through WLs while MVM outputs are read on BLs. In this example DACs convert the digital inputs $x_{D,i}$ into voltage pulses with variable length. BLs voltage bias is provided by a circuit (not shown here) placed close to each ADC; elementary PCM cell in terms of resistances (b); (c) and (d) depict the analog input obtained for $x_{D,i} = 5$ using a PWM and BS time encoding, respectively.

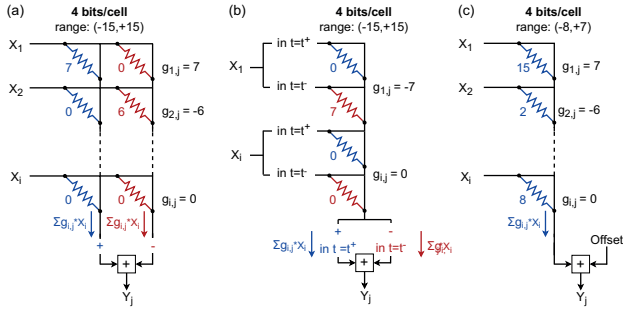


Fig. 3. Differential weights encoding in space (a) time (b) and offset (c).

[24], i.e. rescaling the minimum (for negative weights) conductance value in the RESET state of the PCM cell. This push the entire output dynamic in the positive range with an offset that has to be accounted after performing the A-D conversion. Analogous considerations can be applied while dealing with signed inputs. Additional problems arise when both signed input and weights must be handled at the same time: in this case a combination of the two is necessary.

A concept introduced in [20] highlights the fact that the computation of the inference of DNNs in AiMC architecture is more accurate when the conductance of each PCM cell is directly proportional to the weights of the kernels of the NN and the analog result of the MAC operation is directly proportional to its numerical calculation. This condition is guaranteed only by the differential technique since the offset violates the direct proportionality. It is worth also to note that most weights in a DNN are equal to zero. Having no current when the cell has to store a '0' could represent a significant practical advantage. Thus, this could be another possible downside when implementing offset encoding of negative quantities.

## III. Input $X$ D-A Conversion: WL-DACs

As we have seen, a convenient way to provide the analog input $X$ to $A$ is through time encoding of $x_i$ by following both a PWM or a BS approach depending on the array topology. This means that the WL-DAC is, in practice, a digital-to-time converter, DTC. Together with the usual constraints on area and power, the design of this block comes with additional constraints to guarantee synchronicity between input signals provided to different WLs. Due to the significant size of the array, a propagation delay arises between the top and bottom WLs. Thus, along a BL, it could happen that some cells start (or stop) conducting with different time alignment to others or, even worse, signals supposed to be synchronous are not properly aligned at given $y$. This, combined with delay in the $x$-direction, could lead to inaccuracies in the computation (thus practically limiting the minimum value of $t_u$). Normally, the procedure used to meet all the timing constraints is to synthesize the RTL through a digital design tool.

Even if a practical implementation of the DTC is not going to be presented, some general considerations can be provided. Indeed, the main purpose of the DTC is to provide a suitable enable-like signal controlling when a specific WL has to be activated (depending on input vector values). Even if, ideally, these signals could be propagated directly from global buses to the WL terminals (see in Fig. 1), this does not represent a convenient approach due to different propagation delays associated to the non-identical physical characteristics of paths (e.g. from global bus to I/O interface and, finally, to WL DACs) that adversely affects synchronism determining significant timing errors. In this respect, a way for addressing this issue, consists in (locally) storing the digital values for each WL in a (local) register close to the WL and, then, through (local) logic, generate the actual WL "analog" enable. This way, only one global synchronization signal has to be propagated through the array to provide the required correct timing.

## IV. Output $Y$ A-D Conversion: BL-ADCs

The A-D converter represents probably the most critical circuit of an AiMC system. To compare different topologies, it is important to consider the following parameters:
*Area* - ADC has to be directly interfaced with the tile. To fully take advantage of AiMC having an ADC for each BL would be the best option. This is not going to be the case in practice since the minimum distance between adjacent BLs, that we are going to call BL pitch, can be consistently smaller than the size required to build an ADC. This mostly affects the ADC aspect ratio since one dimension (in our case the one running along the WLs) is going to be considerably smaller than the one along the BL. To overcome this issue it is necessary to share an ADC for a given number of BLs, this number depends on the pitch size. Fortunately this aspect, that is a problem, could conveniently be exploited to store kernels belonging to different DNNs in different BLs connected to a single ADC. In general, based on the BL pitch and the parallelism required by the AiMC application it is first possible to determine the width of the converter and, then, based on the total area that we are willing to occupy (a good compromise could be around 30% of the memory array for the entire BL-ADCs block) to calculate the ADC size in the other direction;
*Time overhead* - as we mentioned before in (3) and (4), depending of the input encoding, the minimum time required to perform a MAC operation is limited by physical constraints in propagating control signals over the memory array. We would like the A-D conversion to not introduce additional delay or to be at least smaller than the time necessary to prepare the next integration ($t_{ch}$);
*Resolution* - the full precision guarantee criterion [24], predicts that, within a single conversion step, the number of bits $b_y$ of $Y_D$ elements can be calculated as:

$$b_Y = \begin{cases} b_X + b_{a_{i,j}} + log_2(\#WL) \text{ if } b_X, b_{a_{i,j}} > 1, \\ b_X + b_{a_{i,j}} + log_2(\#WL) - 1 \text{ otherwise} \end{cases} \quad (6)$$

where $b_{a_{i,j}}$ is the equivalent number of bits of the weights (ML programming). This way, the maximum A-D relative error is:

$$\epsilon_{ADC} = \frac{2^{(b_Y - b_{ADC})} - 1}{2^{b_Y} - 1} \quad (7)$$

If A-D quantization error is similar to the error coming from analog inaccuracies it doesn't represent a considerable decrement to performance. Moreover, DNNs typically exhibit

a precision of 8 bits [22] that could be taken as a realistic resolution of an ADC for AiMC. However, the ADC range must be carefully calibrated for this to work. Indeed, ADC range cannot be too large (strong quantization noise) nor too small (strong clipping). An additional constraint is that ADC range must be the same for all cores, and therefore layers of the DNN, so no fine tuning of ADC range per layer is possible;
*Power overhead* - reasonably the power consumption of the BL-ADCs must be adequately smaller than power consumption required to drive the NVM array [18];
*Flexibility* - possibility of integrating non-linear functions directly in the ADC: this could lead to an increased throughput of the system by removing the activation layer computations in the digital domain between a MAC and the following one.

A key aspect to consider is that different kinds of input encodings generally call for different BL-ADC topologies. Let us consider for instance the design of a BL-ADC for PWM encoding. In this case, the maximum charge integrated on the BL (as mentioned before, the charge is the final result of the BL MAC operation) $Q_{int,PWM}$ is roughly equal to $I_{max} \cdot t_{int,PWM}$ where $I_{max}$ is equal to $\#WL \cdot g_{max} V_{bl}$. The minimum capacitance $C_{S\&H,min,PWM}$ needed to store $Q_{int,PWM}$ is about $Q_{int,PWM}/V_{dd}$. If we suppose to have $I_{max} = 100uA$, $t_{int,PWM} = 100ns$ and a supply voltage of $1V$, the resulting minimum S&H capacitance is $10pF$. Clearly even in the most scaled technology, this capacitance is going to exceed, by itself, the minimum area constraint that we previously discussed to guarantee an adequate level of parallelism. This inhibits the use of S&H based ADCs such as SARs and pipeline that exhibits state of the art performance in the required output resolution [25].

### A. BL-ADCs: Attractive Schemes

From above discussion it is evident that the design ADCs for AiMC requires innovative approaches. In this respect, a suitable solution used for example in PWM-based encodings is offered by the so called current based ADCs using Current Controlled Oscillators (CCOs). The topology of this converter can be summarized in 2 blocks: a CCO that has the goal to generate, inside the converter, a clock frequency proportional to $I_{bl}$ (and, hence, somehow related to the result of the BL MAC operation, the relation being not necessarily straightforward); an asynchronous ripple counter (ARC), driven by the clock generated by the CCO, that provides the output code.

This topology offers great advantages such as: reduced area occupation (only 2 capacitors of few tens of fF are required, followed by a cascade of basic logic gates); a run-time conversion that happens during the integration: as soon as the current stop flowing the result is ready on the output of the ARC. This allow to eliminate any time overhead in the conversion; a small additional power contribution is required by the digital logic to operate. However, this represents a marginal contribution with respect to the current flow in the bit-line, as expressed in (1) that is used to charge and discharge the input capacitor; ease of implementing a ReLu function directly on the ADC by resetting the result if the MSB is negative. The main drawback of this solution is represented

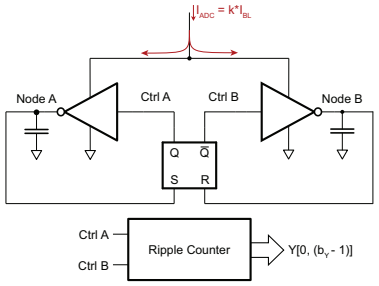


Fig. 4. Basic scheme of a CCO-based BL-ADC. The current $I_{ADC}$ is not necessarily equal to $I_{BL}$ but can be a scaled replica. This current is alternatively switched between capacitors A and B to activate the oscillation on nodes A and B.

by intrinsic non-linearity of the CCO (mainly due to the fact that oscillator loop delay $t_d$ is almost independent from $I_{bl}$). This leads to a total oscillator period that has the shape of:

$$T_{CCO}(t) = \frac{C_{int} * V_{th}}{\alpha * I_{BL}(t)} + t_d. \tag{8}$$

A possible workaround to address this issue is to make the delay $t_d$ proportional to the current $I_{bl}$ [9].

Another potentially viable current-mode ADC alternative is represented by a current starved ring oscillator (RO) [26], [27]. Here the output is partially provided as the value of each stage of the RO, enabling a direct Time-to-Digital conversion. However, the application to this topology to large arrays, hence potentially large currents, is still unproven.

On the other hand, a bit-serial input encoding decreases significantly the integration time when the number of input bit increase ($b_X \cdot t_u$ vs $(2^{b_X} - 1) \cdot t_u$ in PWM). This consequently decrease the amount of charge (hence signal) that has to be converted. In this framework a sample & hold approach could be applied without too much area penalty.

In BS the final result is obtained in successive steps each requiring an A-D conversion. Here no accuracy is lost if:

$$b_Y = \begin{cases} b_{a_{i,j}} + log_2(\#WL) \text{ if } b_X > 1, b_{a_{i,j}} > 1, \\ b_{a_{i,j}} + log_2(\#WL) - 1 \text{ otherwise} \end{cases} \tag{9}$$

This may reduce the resolution required by the converter for the same input parameters with respect to the PWM case. If the precision required is in order of 2 to 4 bits, flash ADCs can be used without introducing too much area penalty [8].

## V. Conclusion

Analog In-Memory Computing (AiMC) based on Non-Volatile Memories (NVM) is a promising candidate to reduce latency and power consumption of neural network inference in edge-computing applications. In this tutorial constraint and critical aspects in the design of D-A and A-D converters have been reviewed, analyzing different topologies and their correlation with the requirements coming from various input encoding mechanisms. More challenges have to be faced in the next few years. Indeed, next generation AiMC systems will require larger arrays to map larger NN and enhance parallelism. This will ask for dedicated co-design strategies to counteract the increasing power, latency and area that growing computational complexity is going to require.

## REFERENCES

[1] A. Guo, X. Si, X. Chen, F. Dong, X. Pu, D. Li, Y. Zhou, L. Ren, Y. Xue, X. Dong, H. Gao, Y. Zhang, J. Zhang, Y. Kong, T. Xiong, B. Wang, H. Cai, W. Shan, and J. Yang, "A 28nm 64-kb 31.6-TFLOPS/W digital-domain floating-point-computing-unit and double-bit 6T-SRAM computing-in-memory macro for floating-point CNNs," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 128–130.

[2] P.-C. Wu, J.-W. Su, L.-Y. Hong, J.-S. Ren, C.-H. Chien, H.-Y. Chen, C.-E. Ke, H.-M. Hsiao, S.-H. Li, S.-S. Sheu, W.-C. Lo, S.-C. Chang, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 22nm 832kb hybrid-domain floating-point sram in-memory-compute macro with 16.2-70.2TFLOPS/W for high-accuracy AI-edge devices," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 126–128.

[3] G. Desoli, N. Chawla, T. Boesch, M. Avodhyawasi, H. Rawat, H. Chawla, V. Abhijith, P. Zambotti, A. Sharma, C. Cappetta, M. Rossi, A. De Vita, and F. Girardi, "16.7 a 40-310TOPS/W SRAM-based all-digital up to 4b in-memory computing multi-tiled nn accelerator in FD-SOI 18nm for deep-learning edge applications," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 260–262.

[4] S. Kim, Z. Li, S. Um, W. Jo, S. Ha, J. Lee, S. Kim, D. Han, and H.-J. Yoo, "16.5 dynaplasia: An eDRAM in-memory-computing-based reconfigurable spatial accelerator with triple-mode cell for dynamic resource switching," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 256–258.

[5] S. Xie, C. Ni, A. Sayal, P. Jain, F. Hamzaoglu, and J. P. Kulkarni, "16.2 eDRAM-CIM: Compute-in-memory design with reconfigurable embedded-dynamic-memory array realizing adaptive data converters and charge-domain computing," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 248–250.

[6] 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, 2022, pp. 1–3.

[7] W.-H. Huang, T.-H. Wen, J.-M. Hung, W.-S. Khwa, Y.-C. Lo, C.-J. Jhang, H.-H. Hsu, Y.-H. Chin, Y.-C. Chen, C.-C. Lo, R.-S. Liu, K.-T. Tang, C.-C. Hsieh, Y.-D. Chih, T.-Y. Chang, and M.-F. Chang, "A nonvolatile al-edge processor with 4mb SLC-MLC hybrid-mode ReRAM compute-in-memory macro and 51.4-251TOPS/W," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 15–17.

[8] W.-S. Khwa, Y.-C. Chiu, C.-J. Jhang, S.-P. Huang, C.-Y. Lee, T.-H. Wen, F.-C. Chang, S.-M. Yu, T.-Y. Lee, and M.-F. Chang, "A 40-nm, 2M-cell, 8b-precision, hybrid SLC-MLC PCM computing-in-memory macro with 20.5 - 65.0TOPS/W for tiny-AL edge devices," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[9] R. Khaddam-Aljameh, M. Stanisavljevic, J. Fornt Mas, G. Karunaratne, M. Brändli, F. Liu, A. Singh, S. M. Müller, U. Egger, A. Petropoulos, T. Antonakopoulos, K. Brew, S. Choi, I. Ok, F. L. Lie, N. Saulnier, V. Chan, I. Ahsan, V. Narayanan, S. R. Nandakumar, M. Le Gallo, P. A. Francese, A. Sebastian, and E. Eleftheriou, "HERMES-core—a 1.59-TOPS/mm$^2$ pcm on 14-nm CMOS in-memory compute core using 300-ps/LSB linearized CCO-based ADCs," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 4, pp. 1027–1038, 2022.

[10] P. Narayanan, S. Ambrogio, A. Okazaki, K. Hosokawa, H. Tsai, A. Nomura, T. Yasuda, C. Mackin, S. C. Lewis, A. Friz, M. Ishii, Y. Kohda, H. Mori, K. Spoon, R. Khaddam-Aljameh, N. Saulnier, M. Bergendahl, J. Demarest, K. W. Brew, V. Chan, S. Choi, I. Ok, I. Ahsan, F. L. Lie, W. Haensch, V. Narayanan, and G. W. Burr, "Fully on-chip MAC at 14 nm enabled by accurate row-wise programming of pcm-based weights and parallel vector-transport in duration-format," *IEEE Transactions on Electron Devices*, vol. 68, no. 12, pp. 6629–6636, 2021.

[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.

[12] 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, 2021, pp. 404–406.

[13] A. Antolini, A. Lico, E. F. Scarselli, A. Gnudi, L. Perilli, M. L. Torres, M. Carissimi, M. Pasotti, and R. A. Canegallo, "An embedded PCM peripheral unit adding analog MAC in-memory computing feature addressing non-linearity and time drift compensation," in *ESSCIRC 2022- IEEE 48th European Solid State Circuits Conference (ESSCIRC)*, 2022, pp. 109–112.

[14] S. Yu and P.-Y. Chen, "Emerging memory technologies: Recent trends and prospects," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, pp. 43–56, 2016.

[15] A. Antolini, C. Paolino, F. Zavalloni, A. Lico, E. F. Scarselli, M. Mangia, F. Pareschi, G. Setti, R. Rovatti, M. L. Torres, M. Carissimi, and M. Pasotti, "Combined HW/SW drift and variability mitigation for PCM-based analog in-memory computing for neural network applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 1, pp. 395–407, 2023.

[16] N. Papandreou, H. Pozidis, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, and E. Eleftheriou, "Programming algorithms for multilevel phase-change memory," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, 2011, pp. 329–332.

[17] A. Cabrini, S. Braga, A. Manetto, and G. Torelli, "Voltage-driven multilevel programming in phase change memories," in *2009 IEEE International Workshop on Memory Technology, Design, and Testing*, 2009, pp. 3–6.

[18] 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)*, 2020, pp. 500–502.

[19] M. L. Gallo and A. Sebastian, "An overview of phase-change memory device physics," *Journal of Physics D: Applied Physics*, vol. 53, no. 21, p. 213002, mar 2020. [Online]. Available: https://dx.doi.org/10.1088/1361-6463/ab7794

[20] T. P. Xiao, B. Feinberg, C. H. Bennett, V. Prabhakar, P. Saxena, V. Agrawal, S. Agarwal, and M. J. Marinella, "On the accuracy of analog neural network inference accelerators," *IEEE Circuits and Systems Magazine*, vol. 22, no. 4, pp. 26–48, 2022.

[21] M. L. Gallo, S. R. Nandakumar, L. Ciric, I. Boybat, R. Khaddam-Aljameh, C. Mackin, and A. Sebastian, "Precision of bit slicing with in-memory computing based on analog phase-change memory crossbars," *Neuromorphic Computing and Engineering*, vol. 2, no. 1, p. 014009, feb 2022. [Online]. Available: https://dx.doi.org/10.1088/2634-4386/ac4fb7

[22] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[23] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, Jan 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-1942-4

[24] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 14–26.

[25] B. Murmann, "ADC Performance Survey 1997-2023," [Online]. Available: https://github.com/bmurmann/ADC-survey.

[26] X. Zhang, J. Acharya, and A. Basu, "A 0.11–0.38 pJ/cycle differential ring oscillator in 65 nm CMOS for robust neurocomputing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 2, pp. 617–630, 2021.

[27] A. Singh, R. Bishnoi, A. Kaichouhi, S. Diware, R. V. Joshi, and S. Hamdioui, "A 115.1 tops/w, 12.1 TOPS/mm$^2$ computation-in-memory using ring-oscillator based ADC for edge AI," in *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2023, pp. 1–5.