IMAGINE: An 8-to-1b 22nm FD-SOI Compute-In-Memory CNN Accelerator With an End-to-End Analog Charge-Based 0.15-8POPS/W Macro Featuring Distribution-Aware Data Reshaping

Adrian Kneip, *Member, IEEE*, Martin Lefebvre, *Member, IEEE*, Pol Maistriaux, *Graduate Student Member, IEEE*, and David Bol, *Senior Member, IEEE*

***Abstract*—Charge-domain compute-in-memory (CIM) SRAMs have recently become an enticing compromise between computing efficiency and accuracy to process sub-8b convolutional neural networks (CNNs) at the edge. Yet, they commonly make use of a fixed dot-product (DP) voltage swing, which leads to a loss in effective ADC bits due to data-dependent clipping or truncation effects that waste precious conversion energy and computing accuracy. To overcome this, we present IMAGINE, a workload-adaptive 1-to-8b CIM-CNN accelerator in 22nm FD-SOI. It introduces a 1152×256 end-to-end charge-based macro with a multi-bit DP based on an input-serial, weight-parallel accumulation that avoids power-hungry DACs. An adaptive swing is achieved by combining a channel-wise DP array split with a linear in-ADC implementation of analog batch-normalization (ABN), obtaining a distribution-aware data reshaping. Critical design constraints are relaxed by including the post-silicon equivalent noise within a CIM-aware CNN training framework. Measurement results showcase an 8b system-level energy efficiency of 40TOPS/W at 0.3/0.6V, with competitive accuracies on MNIST and CIFAR-10. Moreover, the peak energy and area efficiencies of the 187kB/mm^2 macro respectively reach up to 0.15-8POPS/W and 2.6-154TOPS/mm^2, scaling with the 8-to-1b computing precision. These results exceed previous charge-based designs by 3-to-5× while being the first work to provide linear in-memory rescaling.**

***Index Terms*—Computing in-memory (CIM), SRAM, charge-based dot-product (DP), convolutional neural networks (CNNs), analog batch-normalization (ABN), hardware-aware training, 22nm FD-SOI.**

I. Introduction

As of today,the fast-growing deployment of ever-more complex AI tasks in embedded systems has pushed conventional edge devices to their limit. Targeting a variety of biomedical, industrial or environmental applications, powerful AI algorithms such as convolutional neural networks (CNNs) require dedicated edge nodes that provide extreme levels of energy efficiency in order to save battery lifetime and avoid frequent replacement [1], [2]. In that regard, compute-in-memory (CIM) architectures [3], [4] have rapidly gained attention for their ability to efficiently perform massively-parallel dot-product (DP) operations while bypassing the von-Neumann bottleneck, making them suitable candidates to accelerate CNNs at the edge, as depicted in Fig. 1(a). Nowadays, the current landscape of CIM implementations undergoes different trade-offs between computing efficiency and accuracy. On the one hand, analog CIM-SRAMs based on current-domain DP operators yield outstanding computing efficiency [5], [6], but are also highly sensitive to analog impairments such as

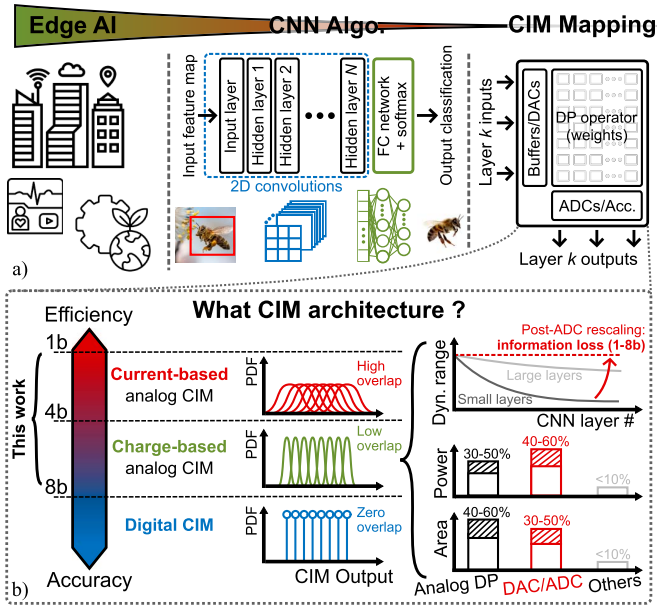

Fig. 1. a) Top-down overview of mapping edge AI applications onto compute-in-memory (CIM) hardware for high-efficiency edge CNN processing. b) Precision scope of existing CIM architectures and illustration of the challenges faced by charge-based ones.

Received 29 November 2024; revised 10 March 2025; accepted 29 May 2025. Date of publication 4 June 2025; date of current version 5 September 2025. This work was supported in part by the Fonds National de la Recherche Scientifique (FNRS), Belgium, under CDR Grant J.0014.20 and in part by Adrian Kneip's Research Fellowship. The review of this article was arranged by Associate Editor Mingoo Seok. *(Corresponding author: Adrian Kneip.)*

Adrian Kneip and Martin Lefebvre are with the EEMCS Department, Delft University of Technology, CD 2628 Delft, The Netherlands (e-mail: a.kneip@tudelft.nl).

Pol Maistriaux and David Bol are with the ICTEAM Institute, UCLouvain, 1348 Louvain-la-Neuve, Belgium.

Digital Object Identifier 10.1109/TCASAI.2025.3576323

nonlinearity, process variations and transistors mismatch, which hinder their computing accuracy [3], [7]. While part of these non-idealities can be compensated during the off-line training of the CNN [6], their actual computing precision usually remains below 4b, restricting their applicative scope. On the other hand, digital CIM macros can support high-precision targets with near-golden accuracy, but waste efficiency due to the overhead of distributed adder trees [8], [9] compared to their analog counterpart. In between, analog charge-based CIM-SRAMs offer an interesting compromise, relying on variation-insensitive metal-oxyde-metal (MoM) capacitances to perform analog DPs with a moderate area overhead [10], [11], while showcasing excellent reliability against process, voltage and temperature shifts [12]. With recent progress in quantization-aware training relaxing the precision requirements below 8b for many edge applications [13], [14], charge-based architectures emerge as appealing candidates for the deployment of versatile CNN workloads at the edge.

Nonetheless, charge-based architectures come with their own challenges, qualitatively illustrated in Fig. 1(b). First, previous works that rely on charge-injection [15], [16] utilize a fixed analog DP voltage swing, bound to the maximum array size. Yet, this approach reduces the effective number of available ADC bits when mapping small- or medium-sized CNN layers that do not require full macro utilization. Furthermore, combining a conventional full-scale ADC with such a fixed voltage swing introduces either clipping or truncation of the ADC output depending on the inbound DP data distribution. Altogether, these issues lead to wasted ADC area and energy, as well as to a loss of information that cannot be recovered by post-ADC rescaling. Finally, supporting a scalable 1-to-8b computing precision puts harsh constraints on DAC and ADC designs, consuming a significant fraction of the total energy and area of the macro and jeopardizing its flexibility towards small workloads. Although fully-serial implementations have been proposed to ease on that side [17], they suffer from a costly 8b ADC conversion per input precision bit.

In this work, we present IMAGINE, a massively-parallel CIM-CNN accelerator embedding a 1-to-8b **I**n-**M**emory-computing SRAM with end-to-end **A**nalog char**G**e-domain comput**I**ng and distributio**N**-aware data r**E**shaping. The featured macro combines a channel-wise swing-adaptive DP operator with an in-ADC multi-bit analog batch-normalization (ABN) function. Flexible bit-precision scaling is enabled by a novel input-serial, weight-parallel post-DP accumulation scheme. The CIM-SRAM macro is co-designed with hardware-aware CNN training to provide resilience against residual non-linearity and variability, enabling approximate LSB computing at the 8b precision to relax the ADC design constraints. IMAGINE has been implemented in 22nm FD-SOI within the CERBERUS chip, with measurement results showcasing peak 8b energy efficiencies of 150 and 40TOPS/W for the standalone macro and the entire accelerator, respectively. The CIM-SRAM also reaches competitive throughput with a high 187kB/mm^2 density, improving overall performance metrics over the state of the art, while being the first to propose a linear in-memory gain rescaling.

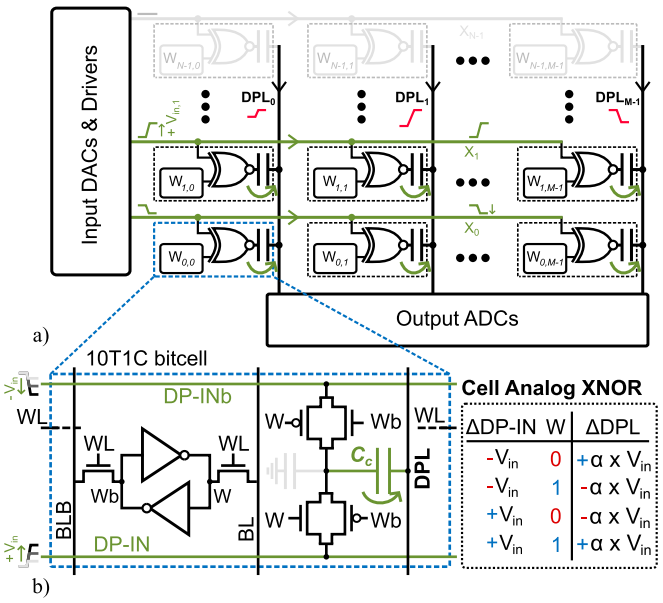

Fig. 2. a) Simplified view of charge-based cim-SRAM architectures, which accumulate the local results of analog XNORs from each b) 10T1C bitcell by means of charge injection through their computing capacitance C_c on the column-shared dot-product line (DPL).

The rest of this paper is outlined as follows. Section II zooms in on the working principle and related challenges of configurable charge-based CIM designs. Then, Section III presents the CIM-SRAM macro architecture, while Section IV covers the accelerator's digital dataflow. Finally, Section V discusses measurement results.

II. Basics and Challenges of Charge-Based CIM

In order to deal with the high inherent nonlinearity and variability of current-domain architectures, charge-based designs leverage variation-insensitive MoM capacitors to complete the analog DP operation with high accuracy. A typical architecture of charge-based CIM-SRAM relying on capacitive coupling is depicted in Fig. 2(a), focusing on the array of DP operators. After the DAC conversion, non-zero inputs X are propagated horizontally on the differential input lines DP-IN and DP-INb. Then, an analog XNOR operation takes place within each 10T1C bitcell, as represented in Fig. 2(b): depending on the value of the stored binary weight W, acting as a + 1/-1 factor, the analog XNOR outputs of each cell are accumulated on their shared dot-product line (DPL) by charge injection through the coupling MoM capacitance C_c, such that

$$V_{DPL,j} = V_{DD}/2 + \alpha \sum_{i=0}^{N_{on}-1} \Delta V_{in,i}\, \mathrm{W}_{i,j} \qquad (1)$$

where

$$\Delta V_{in,i} = \frac{1}{2^{r_{in}}} \sum_{k=0}^{r_{in}-1} (-1)^{1-\mathrm{X}_i[k]}\, 2^k\, V_{DD}. \qquad (2)$$

r_{in} is the bitwise input precision, N_{on} the number of activated rows and α the charge-injection attenuation factor given by $\alpha =$

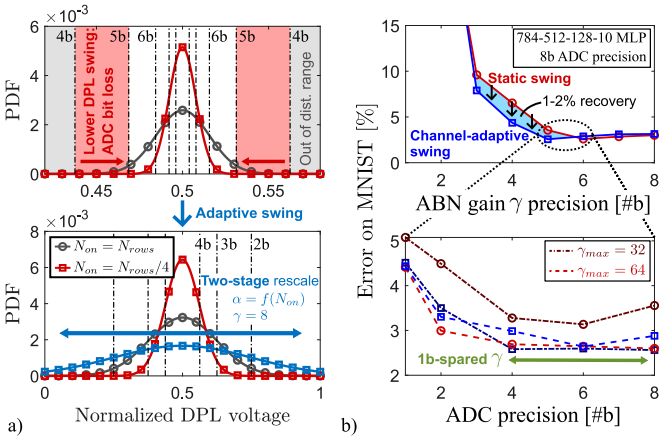

Fig. 3. a) Considering 8b ADCs, narrow normal distribution of DPL voltages in charge-based CIM-SRAMs lead to multiple wasted precision bits during the conversion. Voltage swing reduction with $n_{on} < N_{rows}$ further reduces this effective number of ADC bits. Providing (i) channel-adaptive DPL range compensation and (ii) pre-ADC ABN rescaling help solve these issues. b) Test error on the MNIST dataset for a 784-512-128-10 MLP with various ABN gain and ADC precisions. Providing a channel-adaptive swing on top of ABN rescaling trades accuracy recovery for γ precision.

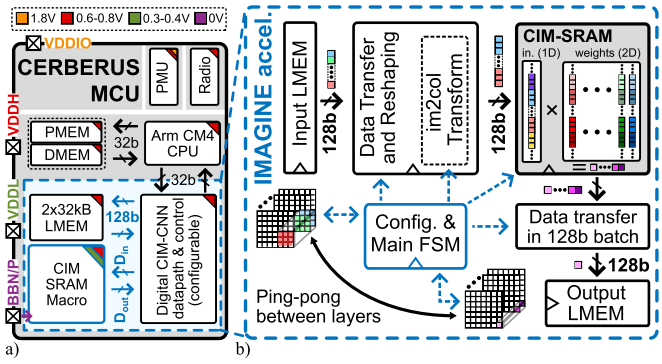

Fig. 4. a) Block diagram of the CERBERUS micro-controller, zooming on b) the IMAGINE mixed-signal CIM-CNN accelerator.

$1/N_{rows}$, ignoring parasitics and other DPL loads. While Eq. (1) demonstrates an excellent linearity, it also points out that covering the full V_{DD} voltage swing of the DPL is only possible at maximum array utilization, and decreases linearly with N_{on}.

The inability to rescale the DPL swing penalizes the mapping of layers with narrow DP distributions, as well as that of layers not utilizing the entire input span (e.g., early layers in CNNs), as they fail to fully utilize the available ADC dynamic range. Taking an example in Fig. 3(a), a CNN layer with a zero-centred DP distribution that uses all (resp. 1/4th) of the CIM-SRAM inputs sees an effective ADC precision reduced by 2b (resp. 3b). This can result in a significant accuracy loss due to the network's inability to learn proper scaling factors, as observed in [6] for ADCs below 6b. Moreover, this dynamic reduction in the effective number of ADC bits leads to wasted ADC power and area, reaching up to more than 75% at 8b, thereby jeopardizing the efficiency of the analog computation.

In light of these challenges, it becomes paramount to be able to adjust the DPL swing based on the input dimension and the DP distribution of the target layer. Such rescaling can be obtained in two steps. First, by configuring the analog DP array to match the target input dimension, making α a function of N_{on} so as to avoid wasting voltage swing and energy. Second, by providing data reshaping abilities prior to the ADC conversion, in order to linearly rescale and bias the DP distribution. This second part can be learned by the network, and thereby merged into a pre-ADC analog batch-normalization (ABN) with gain γ and offset β:

$$V_{ABN,j} = \gamma\, V_{DPL,j} + V_{\beta,j}. \tag{3}$$

By modeling both stages during the CNN training in Fig. 3(b), we demonstrate that the supporting a channel-wise adaptive swing on top of the ABN rescaling saves 1b of γ precision, simplifying the implementation of the ABN gain. Still, the design of these channel-adaptive and distribution-shaping stages has to be carefully carried out to minimize any downside effect on the linearity and variability of the CIM-SRAM operation. In particular, the linearity of the ABN implementation is key to map complex workloads, as opposed to the nonlinear ABN design in [6] which is limited to MNIST-type problems. We thus address these concerns in what follows.

III. Proposed CIM-SRAM Architecture

The proposed IMAGINE accelerator supports the 1-to-8b mapping of CNN layers with various dimensions, thereby offering flexibility on top of high efficiency. IMAGINE is embedded within the 22nm FD-SOI CERBERUS micro-controller unit (MCU), depicted in Fig. 4(a). Detailed in Fig. 4(b), the accelerator features a highly-parallel datapath inspired from [6] with 2×32kB local memory (LMEM) and *im2col* acceleration. IMAGINE's configurable datapath enables 128b end-to-end data transfers in a pipelined manner, providing precision- and size-dependent routing between the LMEMs and the charge-domain CIM-SRAM macro containing the model's weights. The dataflow of the entire accelerator is further discussed in Section IV. Beforehand, let us first focus on the CIM-SRAM architecture.

A. Overall Macro Architecture

Fig. 5(a) showcases the top-level block diagram of the proposed CIM-SRAM macro, supporting 1-to-8b I/O CIM data and a conventional SRAM read/write (R/W) interface to store weights and biases (not shown here for convenience). The macro uses low and high voltage levels V_{DDL} and V_{DDH}, respectively with nominal values of 0.4V and 0.8V. The analog core of the macro spans from the 1152×256 DP array to the distribution-shaping charge-injection (DSCI) ADCs that implement the ABN function. Interestingly, both units involve the same 10T1C bitcell structure shown in Fig. 2(b). Between both ends, the multi-bit input-and-weight (MBIW) units carry out a bitwise sequential input accumulation, before applying the weight bits by means of a summation across adjacent columns. Therefore, the analog core is split into 64 blocks of four columns each, mapping 1-to-4b weight bits as needed and plainly expendable to 8b. Importantly, Fig. 2(b) underlines that each column-wise analog operation occurs on the same

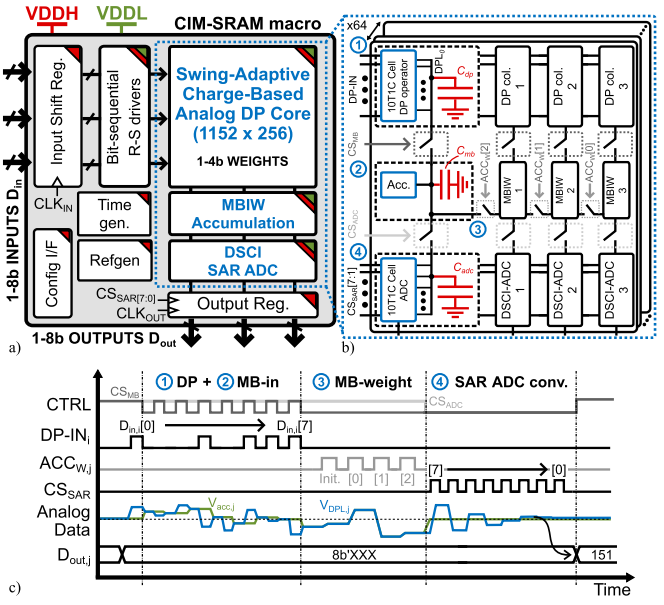

Fig. 5. a) Top-level view of the charge-domain 1152× 256 CIM-SRAM macro, highlighting blocks of main interest. b) Coarse architecture of the 64 DP-to-ADC analog cores, mapping up to 4b weights and carrying out charge-based operations on the same sub-divided DPL along the way. c) Qualitative depiction of the macro's main operations.

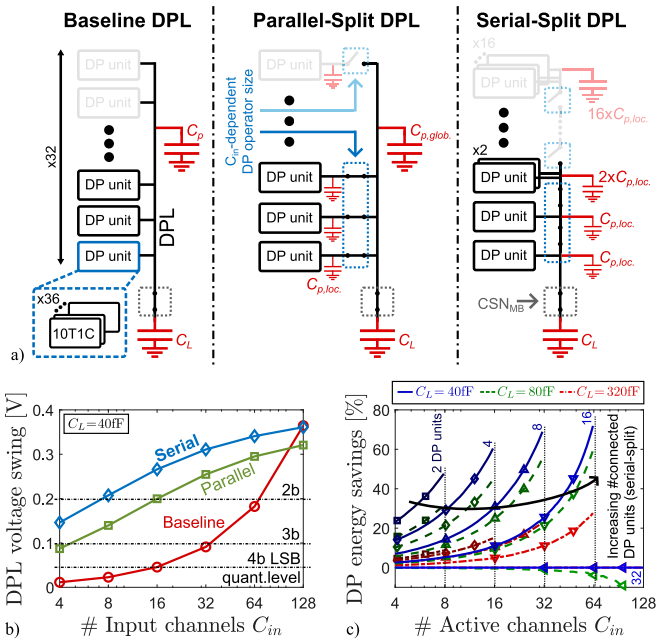

Fig. 6. a) Comparison between baseline and split-DPL architectures, dividing the 1152 DP rows array into 32 even-sized units. b) Improvement of the DPL voltage swing with split-DPL techniques. c) DP energy savings for various capacitive loads in the serial-split DPL case.

DPL without discontinuity, thereby always relying on process-robust charge-domain operations. Transmission gates progressively disconnect parts of the DPL as they become irrelevant, successively reducing the capacitive load seen during the DP, MBIW and ADC stages.

The macro's overall flow of operations is divided into four main steps, qualitatively represented in Fig. 5(c). Assuming an 8b input precision r_{in}, unsigned input data are first broadcast along the enabled DP-IN(b) lines in parallel, performing charge-based DP operations as per Eq. (1) one bit at a time. This process starts with LSB input bits and is repeated r_{in} times, with each subsequent DP operation separated by an accumulation of the DP result on node V_{acc} through a charge sharing with the DPL, as detailed in Section III-C. Once all input bits have been processed, the DP array is entirely disconnected from the DPL and charge sharing between adjacent columns leads to the spatial accumulation of the weight bits. Finally, ADC conversion takes place together with the ABN gain and offset stages, producing the final 1-to-8b rescaled output. The macro's flexible computing precision sustains a configurable trade-off between speed, energy and accuracy.

B. Swing-Adaptive Charge-Based DP Operator

Within each column of the DP array, the total 1152 bitcell rows can be divided into 32 DP units containing 36 cells each. In this way, each unit can map a filter of 2D-convolutional layers with a kernel size of 3× 3 and a minimum of four input channels C_{in}. However, compared to the ideal operator in Eq. (1), the presence of significant load capacitances on the DPL compresses the maximum DPL swing, replacing α by the *actual* attenuation factor

$$\alpha_{eff} = C_c/(N_{dp}\, C_c + C_p + C_L), \tag{4}$$

where $N_{dp} = N_{rows}$ in baseline designs, C_p models the parasitics due to metal routing, and C_L is the total load capacitance related to non-DP blocks connected to the DPL. While C_L is commonly dominated by the ADC input capacitance, its total value can be brought down to 40fF by adapting the ADC architecture, as later described in Section III-D. To further improve α_{eff}, one has to make N_{dp} and C_p functions of the input channel depth C_{in} so as to match the CNN layer size. While charge-sharing-based DP arrays genuinely support an adaptive α_{eff} [10], they dump charges at each compute cycle, preventing any potential reuse of the stored energy. Instead, we propose in Fig. 6(a) two split-DPL techniques for charge-injection-based designs, thereby improving energy reuse: either a *parallel-split* DPL with a C_{in}-dependent amount of local DPLs connected to a shared global DPL through switches, or a *serial-split solution that directly separates the sub-units with switches* on the main DPL. Fig. 6(b) demonstrates the DPL swing improvement using both techniques compared to the baseline case, vastly restoring the effective number of ADC bits at low C_{in}. In this situation, the maximum DPL swing is limited by the significant DPL load C_L. This point is considered during CNN training to compensate the swing reduction by increasing the ABN gain during the DSCI-ADC stage. Furthermore, the parallel-split DPL suffers from additional parasitics $C_{p,glob}$ associated with the global DPL routing, which results in a lower swing improvement compared to a serial-split DPL. Finally, Fig. 6(c) highlights the DP energy reduction with such a DPL division as a function of the number of activated 3× 3 channel rows, for various C_L loads and C_{in} configurations (i.e., number of connected DP units). The savings reach up to 72% with 64 channels and a 40fF load, but rapidly diminish with a higher

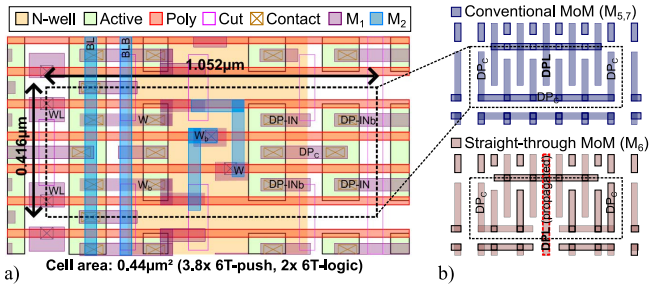

Fig. 7. Layout of the implemented 10T1C bitcell, respectively showing a) the transistors position and b) the custom MoM atop the cell. The metal-6 (M6) MoM layer propagates the DPL vertically.

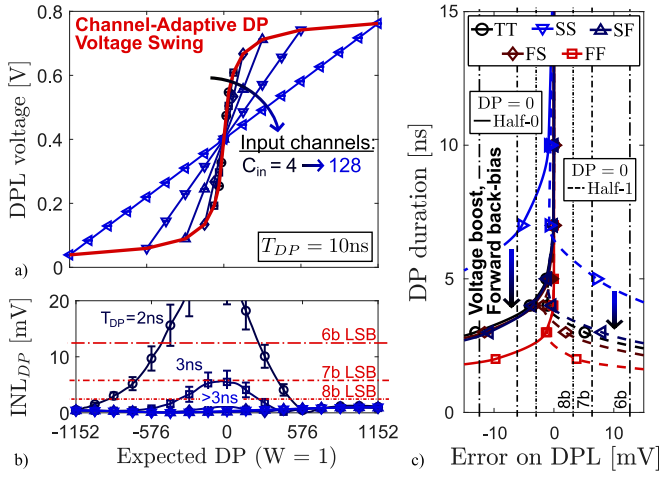

Fig. 8. a) Post-layout transfer function of the DP operator, highlighting the configurable voltage swing. b) Linearity error INL_{DP} observed for different DP durations T_{DP}. c) Worst-case error on the DP result for different process corners and with opposite worst-case weights distribution.

load as the total capacitance seen by the DP-IN driver rises. This further underlines the interest of optimizing the total DPL load.

The 10T1C bitcell's layout is drawn in Fig. 7 and achieves an area of $0.44\mu\mathrm{m}^2$, corresponding to four times that of a 6T-pushed rule bitcell in the same technology. With bottom metal layers M1-3 used for the bitcell data, control and power routing, a custom MoM capacitor is layouted atop the cell in M5-6-7 layers, with the M6 one propagated vertically. The M4 layer remains mostly unused to avoid any significant coupling between the MoM and the internal bitcell nodes. The resulting C_c capacitance reaches 0.7fF, generating a 2.4mV k_BT/C_c noise from the active bitcell transistors. This noise remains below the 8b LSB voltage (3.125mV) and is further attenuated by α_{eff}. Importantly, the metallization constraint does not allow to layout a parallel-split DPL without resorting on lower metal levels that would tightly couple with other signals. This point eventually motivates the use of the serial-split DPL solution.

The post-layout transfer function of the serial-split DP operator is reported in Fig. 8(a) with a 10ns DP duration T_{DP}, including device-to-device variability and highlighting the nonlinear dependence of the DPL swing on C_{in}. Moreover, we consider in Fig. 8(b) the linearity error on the DP result $\mathrm{INL}_{DP} = |V_{DPL} - V_{lin}|$ as a function of the DP duration across the full

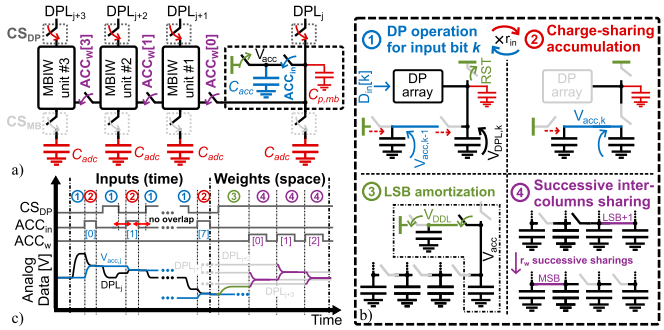

Fig. 9. a) Architecture of the MBIW unit, based on iterative charge-sharing both in time (input) and space (weights). b) Sequence of operations of the MBIW unit, with c) qualitative depiction of the sequential-input, spatial-weight accumulation.

array size. Here, V_{lin} is given by Eq. (1), with $\alpha = \alpha_{eff}$. Indeed, the transmission gates in the serial-split DPL architecture limit the charge-sharing speed, especially as the target DPL voltage nears $V_{DDH}/2$, weakening the driving strength of these gates. This increases the DP duration needed to satisfy the worst-case DPL settling time, or leads to a case-dependent error on the DPL voltage. The largest errors are obtained with the bottom half of the DP array injecting a positive voltage (*half-1*) onto the DPL, while the top half injects a negative one, or the opposite (*half-0*). This phenomenon is worsened when accounting for process corners as in Fig. 8(c), requiring margins on the DP duration. Here, we choose a duration of 5ns per single-bit DP, with a +/-1ns configurability range. These could be extended to also mitigate possible temperature variations, which are not considered in this work. As of now, we achieve an acceptable speed while limiting the linearity error below one LSB, and model it during CNN training. Note that the parallel-split DPL only needs a 1.5ns delay thanks to its lower series resistance.

C. Multi-Bit Input-and-Weight Accumulation

The overall architecture of one of the 64 MBIW units is depicted in Fig. 9(a) across a block of four adjacent columns. Each column possesses its own accumulation capacitance C_{acc}, layouted within the vertical 10T1C pitch and sized to equal the remaining DPL load. Hence, $C_{acc} = C_{mb} + C_{adc}$, with C_{mb} and C_{adc} respectively the total DPL loads associated with the MBIW and ADC blocks. The MBIW operates along four phases described in Figs. 9(b) and 9(c), all relying on capacitive charge-sharing. Phases 1 and 2 correspond to the accumulation of input bit precisions by repeating the DP operation r_{in} times. During the DP phase, the DPL voltage is sampled on the total MBIW and ADC load capacitance, while the accumulation capacitance is disconnected from it, storing the previous accumulation voltage $V_{acc,k-1}$. Then, the DP array is disconnected, while the accumulation capacitance is shared with the DPL load. Before performing the next DP computation, the MBIW accumulation node is disconnected from that load and the DPL is reset to V_{DDL}. Critically, the signals that respectively control connections to the DP array (CS_{DP}) and accumulation node (ACC_{in}) must not overlap to avoid any corruption of the stored V_{acc}. The

DPL voltage resulting from these r_{in} cycles is given by

$$V_{DPL} = V_{DDL}\left(\alpha_{mb}^{r_{in}-1} + \sum_{k=0}^{r_{in}-1} \alpha_{mb}^{r_{in}-k}\,(1 + \alpha_{eff} \sum_{i=0}^{N_{on}-1} \mathrm{X}_i[k]\,\mathrm{W}_{i,j})\right) \quad (5)$$

where $\alpha_{mb} = (C_{mb} + C_{adc})/(C_{acc} + C_{mb} + C_{adc}) \simeq 1/2$ is the multi-bit attenuation factor. In case of binary inputs, the accumulation phase is bypassed altogether, preserving the voltage swing seen at DP time. Once the input accumulation is finished, weight accumulation takes place during phases 3 and 4. The LSB weight is first self-weighted by sharing its DPL with the V_{DDL}-precharged accumulation node. Then, inter-column charge sharing takes place, successively sharing DPLs by pairs of two from LSB to MSB, obtaining the final MBIW result on the MSB's DPL

$$V_{MBIW} = \sum_{k=0}^{r_w-1} (1/2)^{r_w-k}\, V_{DPL,k} \quad (6)$$

with r_w the weights precision. Observe that performing inter-column charge-sharing by pairs of two rather than all at once mitigates the voltage range compression resulting from the sharing. Extending this scheme to 8b weights is straightforward by connecting more columns in a similar manner, but remains unsupported here to limit the configuration complexity. Moreover, while such charge-based weighting maximizes SNR [15] with a simple and regular hardware implementation, it does not inherently support the representation of zero-valued operands, thereby ignoring potential energy savings for sparse workloads through bit-level data gating [18]. Therefore, DP-IN drivers also support a sparse mode with sign-magnitude encoding of the activations and single DP-IN line toggling, based on the MSB's sign. As for weights, pruning techniques can be leveraged to improve both storage density and computing efficiency [19]. Going for an analog adder tree design with inverted MSB [20] is another option, but its lack of flexibility would restrict the CIM-SRAM support to a single weight precision.

Aside from the below-1% capacitance imbalance described by α_{mb}, the high linearity of Eqs. (5) and (6) is altered by leakage currents that (dis)charge the accumulation capacitance, and by non-zero charge injections from the MOS transmission gates serving as switches. Figs. 10(a) and 10(b) respectively consider the impact of both non-idealities on node V_{acc} during input accumulation, across all process corners (similar conclusions apply to temperature and voltage variations). Regarding leakage, the voltage error measured at the end of the 8b input accumulation phase T_{leak} remains negligible, except for unlikely extreme voltage values. Besides, while the impact of charge injection changes with the MBIW input voltage $V_{in,k}$ (corresponding to the k-th DP result), it stays below one LSB across all process corners. This dependence on voltage comes from the change in gate-to-source/drain capacitances of the transmission gates with their V_{gs} voltage, such that the error on $V_{acc,k}$ depends on both the input value $V_{in,k}$ and the accumulation voltage $V_{acc,k-1}$ stored before the charge-sharing step.

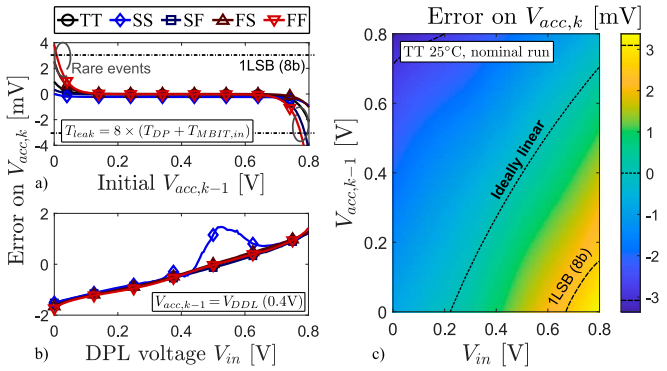

Fig. 10. Linearity error on the accumulation voltage across process corners due to a) leakage currents as a function of the initial $V_{acc,k-1}$, b) charge-injection, for different MBIW input voltages V_{in}. c) 2-D error map on the on the input accumulation voltage $V_{acc,k}$, depending on both the input voltage from the k-th DP operation and the previously stored voltage $V_{acc,k-1}$.

Fig. 10(c) showcases a 2D-map of this error in nominal conditions, highlighting the zero-error curve. The accumulation error reaches up to +/-1 LSB of an 8b ADC, acting as a deterministic input-dependent offset that we model during the CNN training. These results also apply for the weight-parallel accumulation, which reuses the same capacitors, but has a more relaxed 4b-LSB limit.

D. Distribution-Shaping Charge-Injection ADC

At the end of the bitwise accumulation, the MBIW unit is disconnected from the DPL and column-wise ADCs with a 1-to-8b precision r_{out} transform the analog DP result stored on the DPL into digital outputs D_{out}. As seen in Fig. 11(a), these outputs are stored within custom 8b asynchronous DFF registers. Relying on separated control signals for their master and slave latches allows to simultaneously write new ADC outputs to the first latch while preserving previous values in the second latch. Consequently, data available at the CIM output are only updated at the next positive clock edge by a CS_{out} pulse, which enables system-level pipelining as later discussed in Section IV. Finally, the necessary ADC inputs are generated by a gain-adaptive unit based on process- and variation-insensitive reference voltages $V_{BN,P}$, generated by an on-chip calibrated reference [21].

Detailed in Fig. 11(c), the DSCI ADCs perform data conversion in the charge domain, similar to the rest of the analog core. On top of providing in-situ storage ability, 10T1C bitcells form a binary-weighted capacitive array that mimics a charge-injection DAC, where the DPL holds the residual voltage in a SAR-like conversion. Moreover, the ADC performs distribution shaping by implementing ABN offset and gain stages. Hence, the overall ADC is divided in three sub-blocks: (i) a 5b ABN offset unit achieving a +/-30mV offset range on the DPL, (ii) a 7b calibration unit to counteract the sense amplifier (SA) offset, and (iii) an 8b SAR array embedding the ABN gain function. As such, the ADC's 10T1C bitcells not only hold the pre-computed offset and calibration data, but also duplicate the SAR output code storage in-situ. This redundancy avoids to face extreme routing congestion between the 8b ADC registers and the SAR capacitive bank within the tight column pitch.

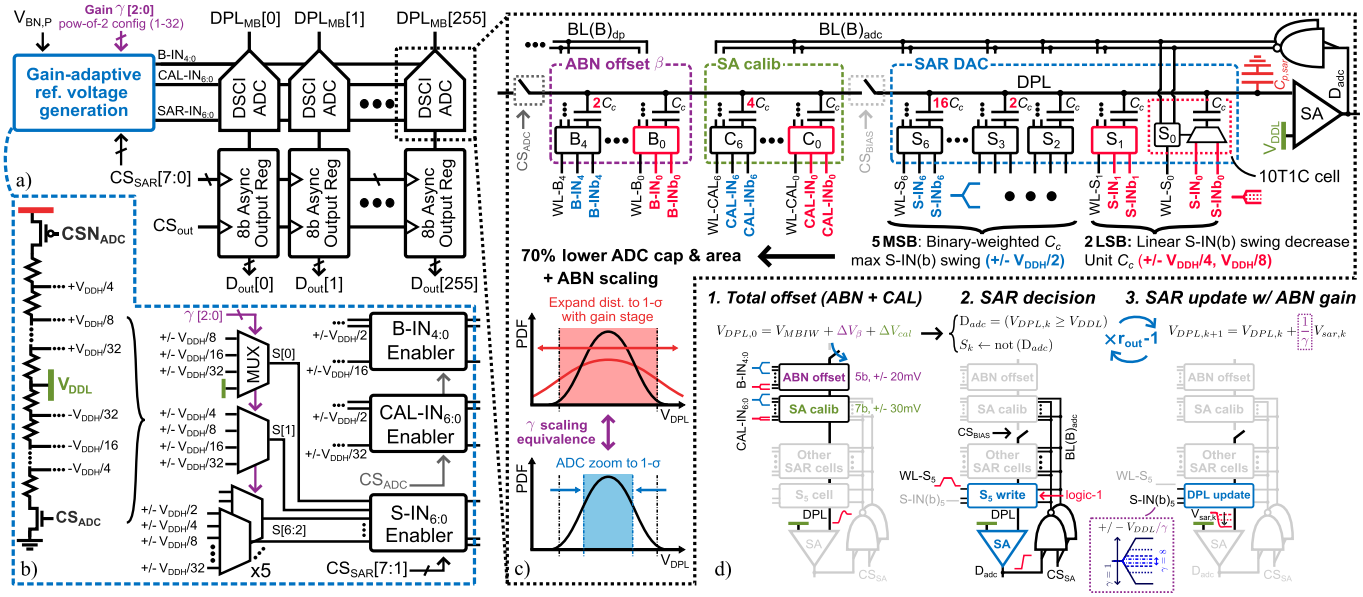

Fig. 11. a) Overall architecture of the distribution-shaping charge-injection (DSCI) ADCs, and of b) its shared gain-adaptive reference ladder implementing ADC thresholds zooming. c) Details of the 10T1C-based DSCI ADC architecture, with d) qualitative depiction of its operations during data conversion. From the data's point of view, ADC zooming taking place during the conversion is equivalent to a scaling effect.

On top of exploiting bitcells, the ADC introduces a voltage-split charge-injection DAC topology, seen in 11(c), that concurrently reduces the total ADC load and enables ABN scaling. Here, relying on a conventional capacitance-split DAC [22] to shrink the large 8b load $C_{adc} = 128 \times C_c$ is prohibited by the floating state of the DPL, which is sensitive to kickback coupling. Instead, the SAR array is split into an MSB part with binary-weighted capacitance values, similar to a conventional DAC, and an LSB part with unit capacitance but linearly-downscaled input swing on the S-IN(b) lines. In this design, the LSB array uses two bitcells to reduce the ADC load by more than 70%, while only requiring two additional input levels and bringing a negligible amount of additional nonlinearity. This technique is also used in the offset and calibration blocks, improving their maximum swing on the DPL by minimizing the load overhead. As a result, ADCs account for less than 5% of the total CIM-SRAM area, with a total load of 40fF per column, including parasitics, enabling the high energy savings predicted in Fig. 6(c). Furthermore, changing the maximum voltage swing of the SAR inputs also offers the opportunity to perform global ABN scaling without the need for an explicit gain stage. Indeed, applying the invert gain factor $1/\gamma$ to all S-IN(b) lines compresses the dynamic range of the ADC, working as a zoom effect equivalent to an explicit scaling of the DPL voltage distribution, as depicted in Fig. 11(c). To that end, the reference unit detailed in Fig. 11(b) adapts the S-IN(b) levels feeding the MSB and LSB split DAC arrays based on the γ configuration. These levels are directly selected from a double-sided resistive ladder, activated during the ADC operation and drawing a 1mA current to settle all S-IN(b) lines within 5ns. Due to mismatch and area constraints, the ladder affords a minimum voltage step of $V_{DDH}/32$, such that the MSB split DAC achieves a maximum gain of 16.

The post-layout operations of the resulting ADC are reported in Fig. 12, both in calibration and conversion modes. Focusing on the conversion mode first, happening at the end of each CIM cycle, ADC operations occur in three phases, depicted in Fig. 11(d). First, the ABN offset and SA calibration blocks add an offset to the initial DPL voltage, following their pre-stored bitcells data. These bitcells are then disconnected from the residual DPL, and SAR operations start. The k-th conversion cycle of the SAR begins with a binary SA decision on the DPL voltage to obtain the corresponding k-th output bit, storing it inside both its column register and the corresponding split-DAC bitcell. Next, the SAR residue is computed by updating the DPL based on the stored value, selecting either S-IN$_k$ or S-INb$_k$ to inject charges on the DPL. As described earlier, the ABN zoom effect is applied during this stage by downscaling the S-IN(b) voltage swing. The decision and update phases are repeated r_{out} times, increasing the SAR delay linearly and, to some extent, its conversion energy. Eventually, the resulting ADC output is given by

$$\mathrm{D}_{out} = \rho\left(2^{r_{out}-1} + \gamma\,\frac{\Delta V_{MBIW} + \Delta V_{\beta} + \Delta V_{cal}}{\alpha_{adc}\,(V_{DDH}/2^{\,r_{out}-1})}\right), \quad (7)$$

where $\rho(\,)$ is the integer floor function, ΔV_{β} and ΔV_{cal} are the ABN bias and calibration offsets, and $\alpha_{adc} = C_{sar}/(C_{sar} + C_{p,sar})$ accounts for the total capacitance of the SAR array, with $C_{sar} = 33\,C_c$. The nominal ADC transfer function extracted in Fig. 13 with zero offset and calibration confirms the high linearity of the ADC and demonstrates the impact of ABN gain scaling. Firstly, we observe that the input dynamic range is compressed due to the attenuation effect, which suitably brings the ADC range closer to the maximum DP voltage range, previously compressed during the DP and MBIW steps. More critically, both the nominal INL ($= |\mathrm{D}_{out} - \mathrm{D}_{lin}|$) and DNL ($= \mathrm{D}_{out,k} - \mathrm{D}_{out,k-1} \forall\, k \in [1, 2^{r_{out}}[$) of the ADC increase with a higher gain γ, as the sensitivity to small nonlinearities between subsequent charge injections increases. These nonlinearities are worsened when considering the parasitic resistances and the

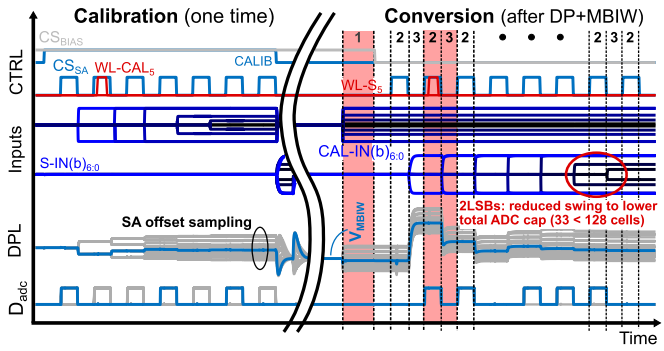

Fig. 12. Post-layout operations of the calibration and conversion phases of the ADC ($\gamma = 1$), showcasing 100 Monte-Carlo iterations.

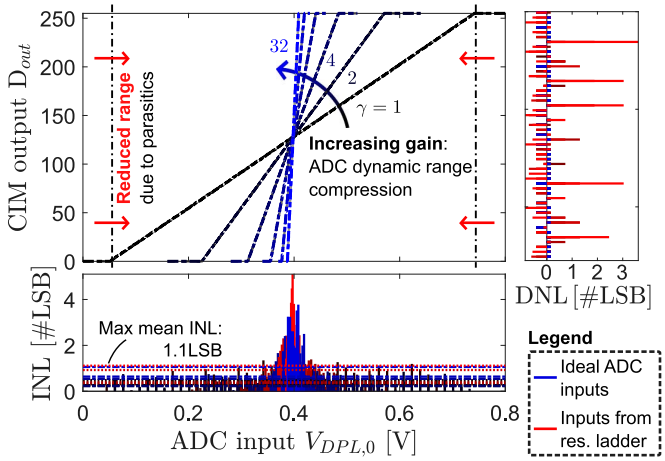

Fig. 13. Nominal post-layout transfer function of the ADC for increasing values of gain γ. The nominal DNL and INL of the ADC increase with γ due to exacerbated sensitivity to voltage errors.

mismatch of the resistive ladder, distorting the perfect linearity of the voltage steps and thereby leading to lost LSB information above $\gamma = 8$, as discussed before. In the end, the ADC reaches a mean INL error of 1.1 LSB, while its peak error rises up to 4.5 LSB when $\gamma = 32$, as the LSB voltage step shrinks. While these limited errors can be deterministically known at CNN train time, both the DNL and INL are further worsened by stochastic sources, namely the SA mismatch as well as the intrinsic noise affecting all DPL-connected blocks. Therefore, we purposefully added a calibration mechanism to the ADC.

E. Mismatch and Low-Frequency Noise Calibration

The ADC calibration occurs on a rare basis to refresh the combined compensation of the SA mismatch and the low-frequency noise affecting the DPL. During calibration, $V_{DPL,0}$ is precharged to V_{DDL} while decision and update phases happen similarly to the conversion phase, but applied to the calibration unit instead of the SAR. In this way, the calibration code converges towards matching the voltage offset seen at the input of each SA, compensating for it during CIM computations as seen in Fig. 13. This offset is namely dominated by transistor mismatch within the SA, which implements the low-kickback StrongArm topology with minimum-length input differential pair shown in Fig. 14(a). These techniques

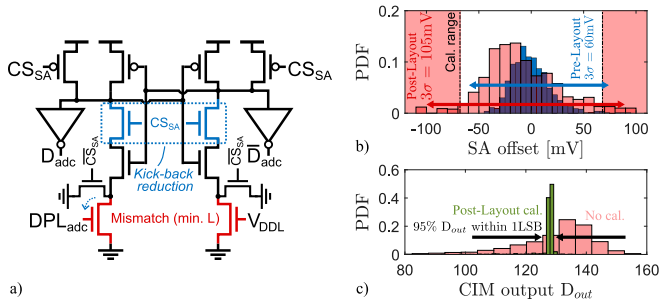

Fig. 14. a) Architecture of the StrongArm SA, with a kickback reduction structure and minimum transistor lengths. b) Distribution of the SA offset, worsened post-layout do to layout resizing constraints and proximity effects. c) Calibration brings 95% of the CIM outputs back within one LSB ($V_{DPL,0} = 0.4$V).

minimize the kickback level on the floating DPL undergone during each SA decision, bringing it below 0.03mV. However, the minimum-length devices further degrade the robustness to mismatch. Therefore, the 7b calibration unit adopts a $4 \times C_c$ MSB capacitance, covering the 3-σ 60mV pre-layout SA offset observed in Fig. 14(b). As such, the calibration process offers a resolution of 0.47mV, making the SAR resilient to any low-frequency SA offset for $\gamma < 8$. Yet, post-layout effects and resizing constraints to fit the column pitch lead to a steep 75% increase of the pre-layout deviation, such that only a 2-σ offset range remains fully handled by the calibration block. With no design-time compensation, out-of-bond offsets might lead to a few dysfunctional columns per macro, as seen in Fig. 14(c). When identified, extreme cases can be partly compensated by the ABN offset cells, at the cost of a reduced offset tuning range for that column. Moreover, the DSCI ADC remains sensitive to common-mode noise on power supplies. A differential SAR architecture as in [23] would help face it, at the cost of doubled area and energy.

IV. CIM-CNN Accelerator Dataflow

IMAGINE embeds the CIM-SRAM macro within a 1-to-8b highly parallel datapath to provide layer-by-layer execution of CNNs. The digital datapath surrounding the macro is based on [6] and represented in Fig. 15(a): it operates with 128b I/O transfers between the CIM-SRAM and LMEMs regardless of the configured bit precision and number of channels. The accelerator operates along four pipelined stages: (i) data fetching from the input LMEM, where data are encoded in a precision-first, channel-second and kernel-last format (ii) optional *im2col* transform for convolutional layers, rearranging the input data in a channel-last format suiting the CIM-SRAM's input shift-register, and applying zero-padding as requested, (iii) CIM computation as described in Section III, and finally (iv) CIM output storage to the output LMEM, in the same kernel-last format. Such format not only minimizes the number of LMEM accesses, but also enables direct data reuse for the next layer, after switching input and output LMEMs in a ping-pong manner. In that sense, LMEM data mapping is similar to [6], extending here the support to an 8b I/O precision. Besides, stage (ii) supports an optional signed-to-unsigned type conversion, and stage (iv) its reverse. While being optimized for 3× 3 kernels,

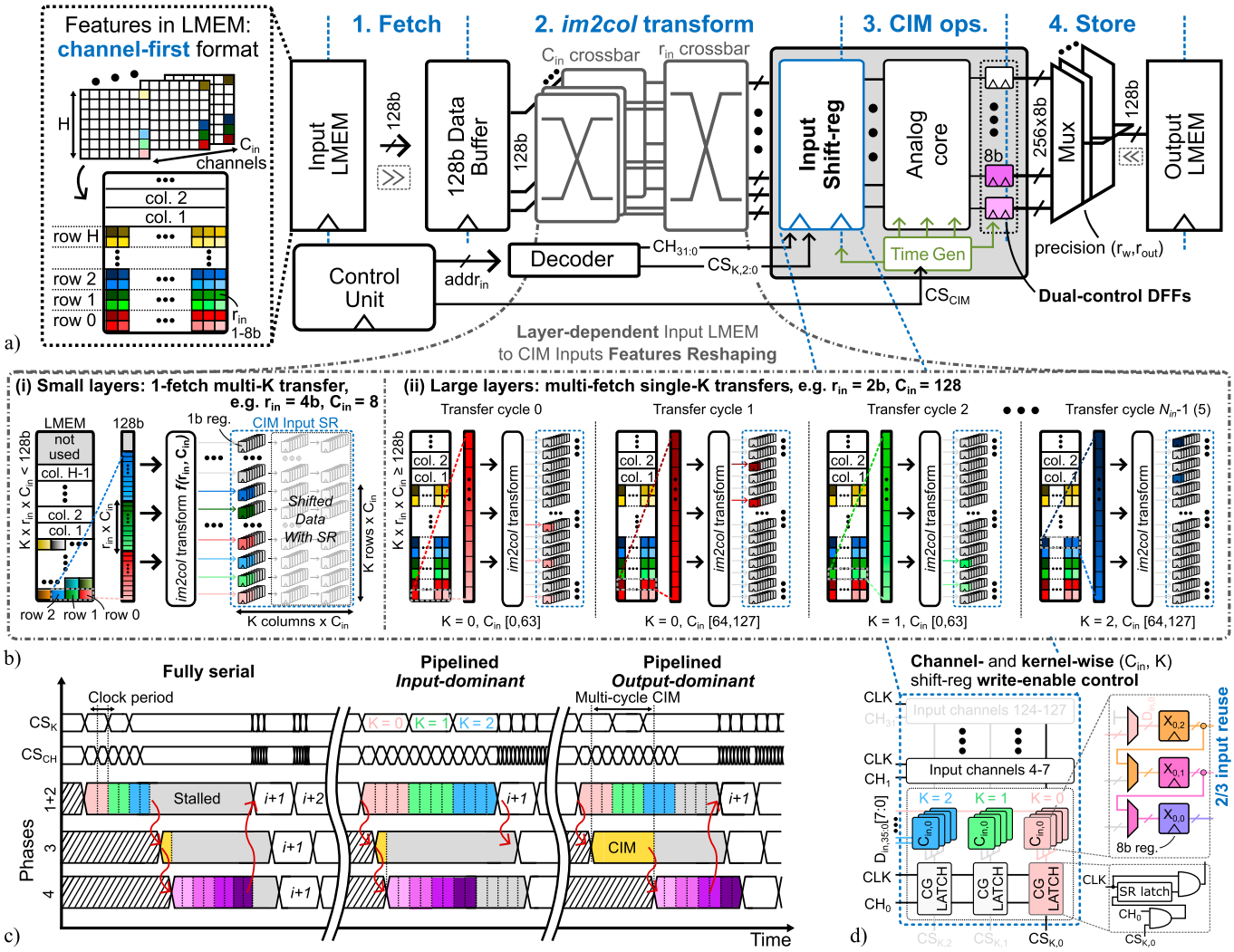

Fig. 15. a) Detailed architecture of the IMAGINE accelerator, with channel-first input LMEM data and a 128b four-stage pipelined datapath. (b) Illustration of data transfers and *im2col* reshaping between the input LMEM and CIM-SRAM input registers for (i) few-channel and (ii) many-channel convolutional layers, respectively fetching the data of multiple or a single kernel(s) per transfer. (c) Qualitative comparison of the accelerator's operations in serial and pipelined cases, with both input- and output-dominated LMEM transfers. (d) Detail of the conditionally-updated CIM-SRAM's input shift-register, which enables more than 60% digital area reduction by avoiding one-shot im2col and data pre-buffering.

larger ones (5× 5, etc.) can be executed on IMAGINE as a succession of 3× 3 kernels [24].

Compared to [6], the $im2col$ conversion is performed sequentially on batches of 128b data fetch from the input LMEM, rather than in a one-shot fashion. This change significantly reduces the required size of the pre-$im2col$ buffer, down to 128b instead of the CIM-SRAM's 1152×8b full input bandwidth, reducing the bit-normalized area overhead of the digital datapath by more than 60%. Nonetheless, this solution requires a more complex input shift-register architecture to support the conditional enabling of different input register subsets on the CIM-SRAM side, depending on the target layer configuration. To that end, the entire shift-register is split into 32 sub-block as detailed in Fig. 15(d), matching the DP array division described in Section III-A. Within each sub-block, local clock gating (CG) latches control the update of the block's registers, with 32 CH_i signals dictating the access to each block, while three $\mathrm{CS}_{K,j}$ signals decide which kernels within the blocks are to be updated. Therefore, the minimum configuration of the CIM-SRAM macro is four input channels in convolutional mode, and one full sub-block in fully-connected mode.

To illustrate the digital-to-macro interface functionality, two transfer situations between the input LMEM and the CIM-SRAM's shift-register are covered in Fig. 15(b), highlighting the *im2col* data reshaping in convolutional mode. On the one hand, CNN layers with few channels and low precision can transfer multiple kernels in a same image row within a single transfer. On the other hand, large CNN layers need to split the transfer of a single kernel over multiple cycles. At fixed C_{in}, the number of cycles is directly proportional to the input precision r_{in}, re-routing the shift-register inputs in the *im2col* as depicted.

Eventually, Fig. 15(d) presents the qualitative flow of IMAGINE's operation phases, considering various situations. Assuming a serial processing first, input transfers remain stalled until all CIM output data have been stored to the output LMEM,

leading to a cycle penalty of

$$N_{stall} = 1 + N_{cim} + \text{ceil}\left(\frac{r_{out} \times C_{out}}{\text{BW}}\right), \quad (8)$$

where $\text{BW} = 128\text{b}$ is the LMEM I/O bandwidth and N_{cim} is the number of clock cycles allowed for CIM-SRAM operations, usually set to one. This penalty severely hinders the overall CIM-CNN throughput, calling for pipelining IMAGINE's operating phases. Ideally, all four phases might then take place simultaneously: while the CIM-SRAM performs the i-th DP computation, previous outputs $i-1$ are stored to the output LMEM whereas new inputs $i+1$ are fetched from the input LMEM. However, this pipeline is subject to data-movement constraints between the different phases, depending on the CNN layer configuration, similar to [6]. On the one hand, *input-dominated* layers wait for input transfers to complete before issuing the next CIM-SRAM and store operations. For a convolutional layer, the number of cycles needed to process a single output-map value is then given by

$$N_{cycles} = N_{in} = (N_{cim} - 1) + \text{ceil}\left(\frac{K \times r_{in} \times C_{in}}{\text{BW}}\right). \quad (9)$$

Eq. (9) showcases that multi-cycle CIM-SRAM computations increase the number of cycles as the data stored within the input shift-register have to remain constant during the entire macro operation. Relying on split control signals for the master and slave latches, as for the macro's output registers, could however circumvent this bottleneck. Moreover, Eq. (9) only holds for data transfers within a same image row, dividing the number of transfers per K thanks to the input shift register. At the start of a new row, $K \times N_{in}$ cycles are necessary to fetch the whole new input data kernel. On the other hand, *output-dominated* layers stall new input data fetches and CIM-SRAM operations while outputs remain to be transferred. In that case, the number of cycles is given by

$$N_{cycles} = N_{out} = N_{cim} + \text{ceil}\left(\frac{r_{out} \times C_{out}}{\text{BW}}\right) - 1. \quad (10)$$

Now, multi-cycle CIM-SRAM operations delay the start of new output storages, which stresses the importance of minimizing the CIM-SRAM's operation time so as to avoid hindering the overall accelerator performance.

Finally, one may wonder how the map workloads that do not fit within the available LMEM or CIM-SRAM capacity. In such cases, the overall execution of a single CNN layer is split into CIM-CNN processing phases, as described above, and CIM-CNN read/write phases to move weights (resp. intermediate results) between an off-chip DRAM and the on-chip CIM-SRAM (resp. LMEMs). These read/write phases pose an end-to-end latency and energy bottleneck on the overall CNN execution. The additional latency depends on the ratio between the off-chip bandwidth and the CIM-CNN bandwidth. For instance, with a 32b off-chip BW, the latency for weight transfers is nearly equivalent to the number of cycles needed to process a single image, assuming a same clock frequency. Regarding energy, 32b DRAM accesses for weight fetching would amount to a total overhead below 10% over the whole image processing, which is acceptable. This overhead, however, may increase if the size of the intermediate layer maps scales down compared to that of the input image, due to pooling layers or striding. In such cases, providing more on-chip storage capacity is the go-to solution, at the expense of more area.

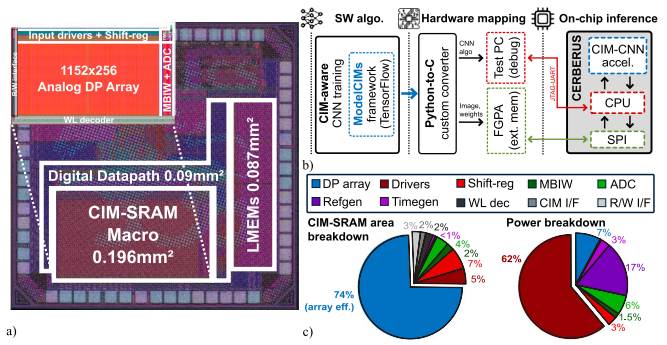

Fig. 16. a) Chip microphotograph with CIM-SRAM macro layout. b) Measurement setup for CNN use cases. c) CIM-SRAM area and power breakdown (1152 active rows, 8b/4b/8b configuration).

V. Measurement Results

The IMAGINE accelerator was embedded in the CERBERUS MCU, fabricated in 22nm FD-SOI. The chip microphotograph and the macro's layout are shown in Fig. 16(a). The CIM-SRAM area is dominated at 74% by the DP array, due to the use of large 10T1C bitcells while minimizing the ADC area as described in Section III-D. Overall, the macro occupies 53% of the 0.373mm^2 total accelerator's area. Both the individual macro and entire accelerator have been characterized using the setup shown in Fig. 16(b), allowing to decouple macro- and system-level performance.

A. CIM-SRAM Characterization

Firstly, standalone characterization of the CIM-SRAM macro is carried out using its fully-connected (FC) configuration to enable easy one-to-one mapping between input LMEM and CIM data. A dedicated test mode allows accurate power estimates by ensuring a 100% duty-cycled utilization of the macro, changing a single 128b patch of inputs per CIM cycle. Fig. 17 reports the macro's transfer function and INL at 8b and 0.6V measured across 256 columns, considering 16 channels (i.e., 128 rows in FC mode) and varying its gain γ. The expected DP is normalized to 1b for convenience, while we consider 8b inputs and 1b weights to simplify extraction. Note that 2b or 4b weight configurations showed close-to-linear trends without much deviation from Fig. 17's normalized response. Here, the transfer function is reported in worst-case mapping conditions, with inputs kept at zero while weights are progressively changed from all-0's to all-1's, from the bottom to the top of the DP array. In this way, we can detect a peak of mean INL around zero-valued DPs, resulting from a slightly short DP timing pulse in the measured slow chip corner, as predicted in Fig. 8(b). In these conditions, the 50-to-210 output dynamic range obtained under unity gain matches the $\sim 70\%$ DPL swing utilization expected from Fig. 6(b) for 16 input channels. Moreover, the aggregated

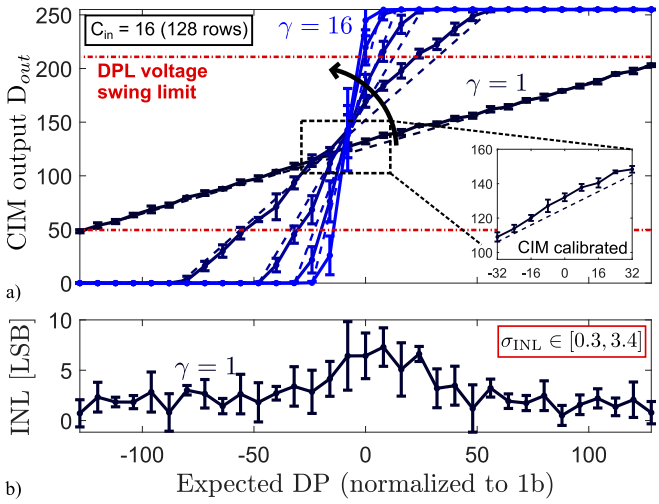

Fig. 17. a) CIM-SRAM 8b tranfer function at 0.6V, with 16 input channels in FC mode and increasing gain γ. b) Observed INL at unity gain (8b/1b inputs/weights).

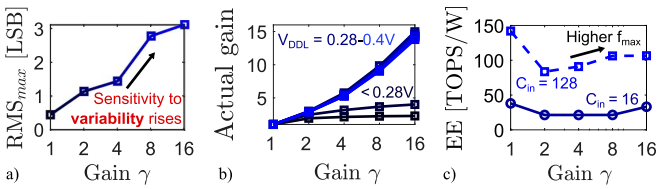

Fig. 18. Impact of gain γ scaling on a) the maximum output RMS error on the INL, b) the linearity of the gain for different supplies, c) the macro's 8b peak energy efficiency (EE).

output variability resulting from temporal noise (100 iterations) and residual column-to-column mismatch, after calibration, amounts to a maximum deviation of 3.5 LSB on the INL under unity gain. This yields a maximum RMS error of 0.52 LSB, which scales up with γ in Fig. 18(a) given the growing sensitivity to the residual noise floor. This error level is obtained after performing the calibration described in Section III-D, which brings the spatial deviation from 17 LSB down to just 2 LSB at the 8b precision, as seen in Fig. 19. Residual errors come from of mix of out-of-range SA offsets, insufficient voltage resolution during the calibration process, and noise induced by MoM caps. Reducing the error further down requires to use larger C_c capacitances and to upsize the SA to contain its mismatch. Nonetheless, both techniques would negatively impact the power and area of the macro. Instead, these low-noise statistics are included during the offline CNN training to be partly compensated. Furthermore, with regards to γ scaling, linearity slowly degrades as the V_{DDL} supply voltage is reduced from 0.4 down to 0.28V in Fig. 18(b), keeping $V_{DDL} = V_{DDH}/2$. Functionality is lost below 0.28V due to the insufficient configuration range of internal timings. Finally, the 8b peak energy efficiency of the CIM-SRAM macro indirectly depends on γ in Fig. 18(c), as the maximum operating frequency of the macro slightly improves between 2 and 16 thanks to the shorter transition time of the compressed V_{sar} levels. However, the efficiency remains better for unity gain as

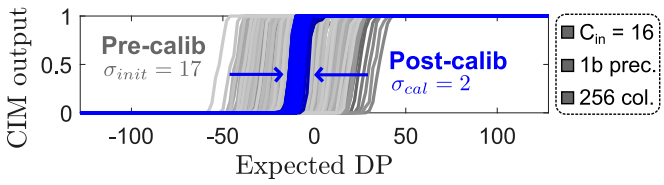

Fig. 19. CIM-SRAM 1b input-referred deviation prior to and after SA offset calibration across 256 columns (average over 100 different samples).

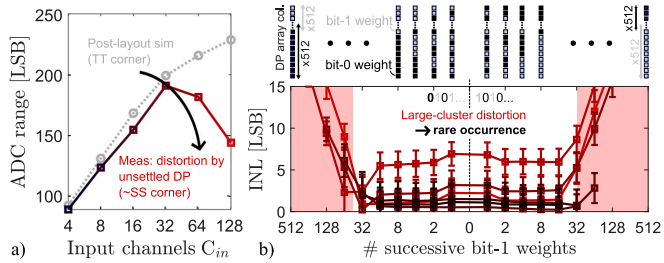

Fig. 20. a) Measured increase of the mean ADC output range with C_{in} ($\gamma = 1$). Unsettled DP in the slow chip corner results in distortion at high C_{in}. b) Measured distortion per C_{in} configuration for a zero-valued expected DP under incremental weight-value clustering (inputs fixed at zero). Mean INL strongly rises in rare highly-clustered cases.

the SAR MSBs are directly connected to ground and supply levels, alleviating the resistive ladder's total load.

Fig. 20(a) showcases the impact of changing the number of input channels. Overall, the output dynamic range improves at constant gain and supply when scaling C_{in} up, closely following post-layout estimates up to 32 channels. However, the maximum swing drops above this value as a result of distortions resulting from an unsettled DP result in the measured slow chip corner, as predicted in Fig. 8(c). Such distortion is estimated in Fig. 20(b) by measuring the output obtained when expecting a zero-valued DP, realized through different combinations of weight and input values. In particular, with inputs fixed at zero and half of the weights storing a bit-1, the other half a bit-0, we increment the number of row-wise consecutive bit-1 (resp. bit-0) weights, starting with a bit-1 (resp. bit-0) in the first CIM-SRAM row. A significant increase in INL appears above 32 consecutive values due to opposing charge injections in the different sub-parts of the split-DPL array, which lacks enough time to properly settle the ADC input node in a slow chip corner. This issue would be avoided with a larger DP timing configuration or a parallel-split DP structure, which could both not be implemented here due to tight area constraints and metallization limitations. Eventually, Fig. 21 points out that the maximum RMS error slightly increases with supply voltage. This phenomenon results from the combined effects of shortened timing pulses at higher voltage, overcoming the increase in transistor driving strength, and larger IR drops under high parallelism. As for the DP distortion, widening the timing pulses by 1.5-2× could help prevent such RMS worsening without compromising the retention of the leaky analog data.

Thanks to the precision-configurable MBIW and SAR units, the proposed CIM-SRAM is able to trade precision for throughput with a close-to-constant energy per computing bit. Notably, Fig. 22(a) reports the measured trade-off between the macro's

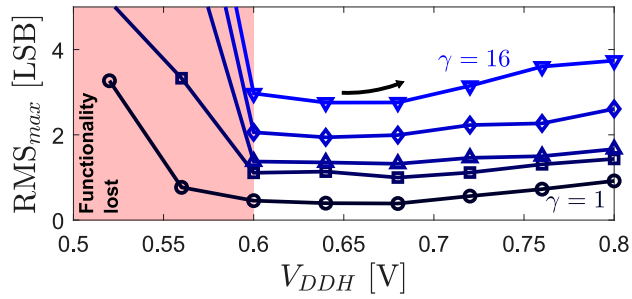

Fig. 21. Under unity gain γ, the 8b output RMS error increases with supply voltage following timing pulses shortening and higher IR drops ($C_{in} = 16$).

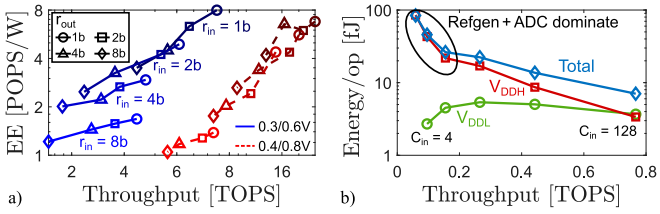

Fig. 22. Trade-off between throughput and a) raw peak energy efficiency for different I/O precisions ($r_w = 1$b) and supplies under unity gain ($C_{in} = 128$), b) 8b-normalized energy/op per supply source when scaling C_{in}.

peak energy efficiency and throughput, evaluated at two supply voltages for different combinations of input and output precisions. As for now, binary weights and 128 input channels (i.e., 1152 rows) with unity gain are considered, while I/O transfer cycles are excluded, and weights loading through the R/W interface are neglected. In such conditions, the highest efficiency per precision value is reached for $r_{in} = r_{out} = 8$b, yielding 1.2POPS/W (i.e., 0.15POPS/W with 8b weights norm.). At constant r_{out}, the macro's energy efficiency scales better than linearly with r_{in} as the fraction due to the single ADC conversion is amortized on more utilizations of the DP array. This is a key motivation for the pre-ADC sequential input accumulation, contrary to post-ADC version in prior works that require one conversion per input [17]. Besides, although $r_{in} < r_{out}$ is common in many previous works [11], [15] to support further digital post-processing, the DSIC ADC architecture makes the need for such configurations lesser in our design as motivated in Section II. Finally, the energy efficiency in the 2b and 4b cases can be directly derived from the linear scaling of Fig. 22(a)'s results, as the contribution of the parallel-weight accumulation process to the total macro's energy is not significant (Fig. 16(c)).

Now looking at the 8b input/output energy breakdown in Fig. 22(b), ADCs and the resistive ladder expectedly dominate the energy/operation at low C_{in}, as the ADC cost is merely amortized by the small-sized analog DP. As C_{in} increases, the disparity shrinks and both V_{DDL} and V_{DDH} supplies amount to a similar contribution to the total energy per operation at high C_{in}.

B. CIM-CNN Accelerator Breakdown

The overall throughput and efficiency of the whole CIM-CNN accelerator is altered by I/O data transfers to/from the macro. In convolutional mode, the standalone CIM-SRAM

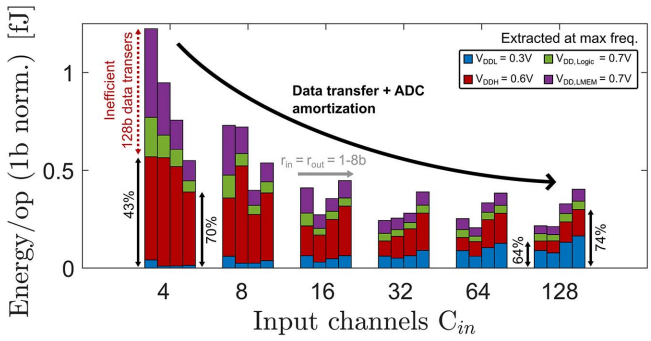

Fig. 23. Energy breakdown of the CIM-CNN accelerator in convolutional mode with increasing input channel and precision configurations (unity gain), at the 0.3/0.6V maximum frequency per point (unity gain γ).

throughput is divided by the number of transfer cycles from Eq. (9) or (10), where we consider the same clock frequency for the macro and its digital datapath. As such, Fig. 23 gives the channel- and precision-wise evolution of the CIM-CNN energy per operation, normalized to 1b and extracted at each configuration's maximum operating frequency for a 0.3/0.6V supply. Another dedicated test mode ensures accurate power estimates by looping on the convolution operation of a 32× 32 image, with randomly distributed inputs and weights. Now, the energy per operation decreases with C_{in} thanks to a better amortization of the ADC overhead (including the resistive ladder DC power) and data transfers. In particular, layer configurations using less than 128b have their total energy dominated by data transfers, which cannot fully make use of the available system bandwidth. On the contrary, the macro's energy accounts for 70-75% of the total accelerator energy in high precision and/or channel configuration. However, it becomes sensitive to leakage integrated over the high number of I/O transfers in the MHz-range. To prevent this while further boosting throughput, combining a higher clock frequency for pipelined transfers with multi-cycle CIM-SRAM operations, as described in Section IV, could be considered. Such solution was however not implemented here due to the higher inbound complexity of the CIM-SRAM's internal time generator as well as design-time constraints, but should be considered in future solutions.

C. Comparison to the State of the Art

The proposed macro and accelerator are compared to other state-of-the-art CIM architectures in Table I, including recent designs in the current, charge and digital domains. IMAGINE achieves the highest density thanks to its custom bitcell layout and area-optimized SAR topology, while reaching a 3-to-5× improved macro-level peak energy efficiency across charge-domain architectures. It also achieves a competitive 0.5TOPS throughput and a 40TOPS/W peak accelerator efficiency while being the first work to provide linear in-memory offset and gain rescaling abilities, therefore skipping the need for most inter-layer processing during CNN execution (whose overhead is usually not reported). Although the accelerator's throughput and energy efficiency are lower than current-based designs, these suffer from a much higher variability on their digitized result, which jeopardizes the mapping of applications requiring

TABLE I
COMPARISON TO STATE-OF-THE-ART CIM DESIGNS IN THE CURRENT, CHARGE AND DIGITAL DOMAINS

	Digital CIM		Analog current-based CIM			Analog charge-based CIM						
	[8]	[5]	[5]	[26]	[6]	[11]	[15]	[16]	[20]	[23]	[27]	**This work**
Year	2022	2023	2021	2022	2023	2021	2021	2022	2024	2024	2023	2023
Technology	28nm Bulk	28nm Bulk	7nm FinFET	22nm FD-SOI	22nm FD-SOI	16nm FinFET	28nm Bulk	22nm Bulk	28nm Bulk	28nm Bulk	12nm FinFET	22nm FD-SOI
Bitcell type	8T	8T-push	8T	12T	6T	8T1C	8T1C	8T1C	9T1C	10T3C	9T1C	10T1C
DP mechanism	Distributed adder tree	Approx. adder tree	Capacitive discharge	Capacitive discharge	Capacitive discharge	Charge-inj. + Bit-shifting	Charge-inj. + Bit-shifting	C-2C sharing	Charge-inj. + Analog adder	Differential Charge-inj.	Charge inj. + Analog adder	Charge inj. + MBIW acc.
On-chip CIM size	2kB	2kB	16×0.5kB	72kB	72kB	576kB	36kB	2×16kB	2kB	36kB	16kB	36kB
Density [kB/mm^2]	40.8	71.4	156.3	31.4	**595.1**	23	70.6	131	64.5	95.7	-	187
Supply voltage [V]	0.5-0.9	0.5-0.9	0.8-1	0.6-0.9	0.4/0.8	0.8	0.6	0.7-1	0.9	0.8	0.53-0.65	0.3/0.6-0.4/0.8
Max precision (in/w/out)	4/1/4b	8/8/8b	4/4/4b	7/1.5/6b	4/4/4b	8/8/8b	5/1/8b	8/8/8b	4/4/6b	4/2/10b	8/8/8b	8/4/8
Analog DP rescaling	-	-	No	No	Nonlinear	No	No	No	No	Fixed	No	Linear
CIM sub-system	No	No	No	Yes	Yes	Yes	No	No	No	No	No	Yes
Peak Throughput [TOPS][1]	0.01-0.325	N/A-0.12	1.5-1.8	2.2-4	0.08-0.4/0.7-4.5	0.25 (4)[2]	0.1	0.6-1	0.05	0.6	N/A-0.1	0.1-0.5
Peak Macro EE [TOPS/W][1]	16-8	38-102	152-109	-	650-1050	51	91	32.2-15.5	41	128-96	86.3-N/A	150-125
Peak AE [TOPS/mm^2][1]	0.16-6.3	4.2	23-36	1.6-2.1[3]	0.27-1.6[2]/ 6-36[3]	0.7	0.18	2.4-4	1.7	1.8	-	0.51-2.6
Peak System EE [TOPS/W][1]	-	-	-	49-29	112-204	30	-	-	-	-	-	40-35
Max 8b output RMS [LSB]	-	-	-	0.8-2	1.7-2.15	0.9	0.98	0.1	2	0.13	0.3	0.32-1.8
MNIST acc. [%]	-	-	99.6	-	98.1[5]	-	-	98.1	-	-	-	98.6[4] (98.8[6])
CIFAR-10 acc. [%]	90.4	91.6	88.5	89.3 (mix)	∼ 73[5]	91.5	91.1	-	91.7	92.3	-	90.8[5] (91.7[6])

[1]Normalized to 8b precision (inputs *and* weights), under maximum row parallelism (IMAGINE: 1152 rows). [2]Includes I/O transfer cycles. [3]Dependent on timings config.
[4]Measured on chip for a modified 4b/1b/4b LeNet-5. [5]Emulated post-silicon on a 4b/4b/4b VGG-16. [6]Ideal SW baseline for target precision and network.

a medium computing precision. Compared to previous charge-based designs, the serial-split DPL used in IMAGINE leads to a slightly smaller throughput and higher RMS following the insufficient timing configuration in the SS corner. However, its compact ADC leads to a higher energy efficiency. While [23] supports a single ADC rescaling option with lower supply injection similar to our DSCI, it provides no extensive DPL adaptivity to map an ABN function, and does not generate its additional reference supplies on chip. Finally, compared to the nonlinear charge-integrating ABN in [6], the linear approach in this work supports more realistic workloads than MNIST, e.g., CIFAR-10 and beyond. Altogether, this work extends the boundaries of the computing efficiency versus accuracy of moderate-precision CIM designs, achieving the best charge-based efficiency to date while supporting flexible workloads thanks to its adaptive dynamic range utilization.

VI. CONCLUSION

In this work, we presented IMAGINE, a 1-to-8b compute-in-memory (CIM) CNN accelerator embedding a charge-based CIM-SRAM with a multi-bit input-serial, weight-parallel (MBIW) end-to-end dot-product (DP) operation. Namely, a split DP line structure allows up to 20× higher voltage swing utilization, depending on the input channel configuration, while in-ADC 5b offset and rescaling effects enable linear data shaping, making full use of the selected ADC precision. Standalone measurement results of the dense 187kB/mm^2 macro showcase peak energy and area efficiencies of 0.15-to-8POPS/W and 2.6-154TOPS/mm^2, respectively, with quasi-linear scaling from 8 to 1b computing, exceeding previous charge-based CIM-SRAM designs by up to 5×. Moreover, the mean measured 8b RMS error under unity gain lays below one LSB, similar to previous works. Still, in-ADC scaling also upscales the RMS error, bringing computing LSBs below the macro's noise floor for 8b computations. Although noise-aware CNN training partly deals with this loss on medium complexity tasks, further mismatch and noise reduction would be required for 8b applications targeting gain values above 16, at the cost of additional energy and area. At the system level, the entire CIM-CNN accelerator reaches a 40TOPS/W peak energy efficiency and an overall throughput comparable to previous works. Nonetheless, it could be further improved by enabling multi-cycle macro operation or, possibly, a precision-last dataflow based on intertwined analog operations and digital transfers. Such approach could bring the intrinsically slower charge-based designs closer to the throughput and system-level efficiency of current-based CIM-SRAM approaches, while taking advantage of their much lower variability.

ACKNOWLEDGMENT

The authors would like to thank ECS team members for their help with the CERBERUS chip design and proofreading of this work, in particular M. Gonzalez and S. Favresse.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[2] M. Lauridsen, R. Krigslund, M. Rohr, and G. Madueno, "An empirical NB-IoT power consumption model for battery lifetime estimation," in *Proc. IEEE VTC Spring*, 2018, pp. 1–5.
[3] J. Zhang, Z. Wang, and N. Verma, "A machine-learning classifier implemented in a standard 6T SRAM array," in *Proc. IEEE VLSI*, 2016, pp. 1–2.
[4] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, Feb. 2018.
[5] M. E. Sinangil et al., "A 7-nm compute-in-memory SRAM macro supporting multi-bit input, weight and output and achieving 351 TOPS/W and 372.4 GOPS," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 188–198, Jan. 2021.
[6] A. Kneip, M. Lefebvre, J. Verecken, and D. Bol, "IMPACT: A 1-to-4b 813-TOPS/W 22-nm FD-SOI compute-in-memory CNN accelerator featuring a 4.2-POPS/W 146-TOPS/mm2 CIM-SRAM with multi-bit

analog batch-normalization," *IEEE J. Solid-State Circuits*, vol. 58, no. 7, pp. 1871–1884, Jul. 2023.
[7] A. Kneip and D. Bol, "Impact of analog non-idealities on the design space of 6T-SRAM current-domain dot-product operators for in-memory computing," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 68, no. 5, pp. 1931–1944, May 2021.
[8] H. Fujiwara et al., "A 5-nm 254-TOPS/W 221-TOPS/mm^2 Fully-digital computing in-memory macro supporting wide-range dynamic-voltage-frequency scaling and simultaneous MAC and Write Operations," in *Proc. IEEE ISSCC*, 2022, pp. 186–188,
[9] D. Wang, C.-T. Lin, G. K. Chen, P. Knag, R. K. Krishnamurthy, and M. Seok, "DIMC: 2219TOPS/W 2569F^2/b digital in-memory computing macro in 28nm based on approximate arithmetic hardware," in *Proc. IEEE ISSCC*, 2022., pp. 266–268,
[10] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-Tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, Jun. 2019.
[11] H. Jia et al., "A programmable neural-network inference accelerator based on scalable in-memory computing," in *Proc. IEEE ISSCC*, 2021, pp. 236–238.
[12] Z. Chen et al., "PICO-RAM: A PVT-insensitive analog compute-in-memory SRAM macro with in situ multi-bit charge computing and 6T thin-cell-compatible layout," *IEEE J. Solid-State Circuits*, vol. 60, no. 1, pp. 308–320, Jan. 2025.
[13] S. Han, H. Mao, and W.-J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2016, *arXiv:1510.00149*.
[14] C. Baskin et al., "UNIQ: Uniform noise injection for non-uniform quantization of neural networks," *ACM Trans. Comput. Syst.*, vol. 37, no. 4, pp. 1–15, 2021.
[15] J. Lee, H. Valavi, Y. Tang, and N. Verma, "Fully row/column-parallel in-memory computing SRAM macro employing capacitor-based mixed-signal computation with 5-b inputs," in *Proc. IEEE VLSI*, 2021, pp. 1–2.
[16] H. Wang et al., "A 32.2 TOPS/W SRAM compute-in-memory macro employing a linear 8-bit C-2C ladder for charge domain computation in 22nm for edge inference," in *Proc. IEEE VLSI*, 2022, pp. 36–37.
[17] H. Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma, "A programmable heterogeneous microprocessor based on bit-scalable in-memory computing," *IEEE J. Solid-State Circuits*, vol. 55, no. 9, pp. 2609–2621, Sep. 2020.
[18] M. Shi, V. Jain, A. Joseph, M. Meijer, and M. Verhelst, "BitWave: Exploiting column-based bit-level sparsity for deep learning acceleration," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA),* 2024, pp. 732–746.
[19] C. Chu et al., "PIM-Prune: Fine-grain DCNN pruning for crossbar-based process-in-memory architecture," in *Proc. 57th ACM/IEEE Des. Autom. Conf. (DAC)*, 2020, pp. 1–6.
[20] B. Zhang et al., "MACC-SRAM: A multistep accumulation capacitor-coupling in-memory computing SRAM macro for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 59, no. 6, pp. 1938–1949, Jun. 2024.
[21] M. Lefebvre and D. Bol, "A family of current references based on 2T voltage references: Demonstration in 0.18-μm With 0.1-nA PTAT and 1.1-μA CWT 38-ppm/°C designs," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 69, no. 8, pp. 3237–3250, Aug. 2022.
[22] K. Lee, S. Cheon, J. Jo, W. Choi, and J. Park, "A charge-sharing based 8T SRAM in-memory computing for edge DNN acceleration," in *Proc. IEEE DAC*, 2021, pp. 739–744.
[23] J. Lee, B. Zhang, and N. Verma, "A switched-capacitor SRAM in-memory computing macro with high-precision, high-efficiency differential architecture," in *Proc. IEEE Eur. Solid-State Electron. Res. Conf. (ESSERC)*, 2024, pp. 357–360.
[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[25] Y. He et al., "A 28nm 38-to-102-TOPS/W 8b multiply-less approximate digital SRAM compute-in-memory macro for neural-network inference," in *Proc. IEEE ISSCC*, 2023, pp. 130–131.
[26] P. Houshmand et al., "DIANA: An end-to-end hybrid DIgital and ANAlog neural network SoC for the edge," *IEEE J. Solid-State Circuits*, vol. 58, no. 1, pp. 203–215, Jan. 2023.
[27] S.-E. Hsieh et al., "A 70.85–86.27TOPS/W PVT-insensitive 8b word-wise ACIM with post-processing relaxation," in *Proc. IEEE ISSCC*, 2023, pp. 136–137.

Adrian Kneip (Member, IEEE) received the M.Sc. degree (*summa cum laude*) in electrical engineering and the Ph.D. degree in engineering sciences from the Université catholique de Louvain (UCLouvain), Louvain-la-Neuve, in 2019 and 2024, respectively. Currently, he is a Postdoctoral Researcher in the CogSys Group led by Dr. C. Frenkel with Delft University of Technology, and a Visiting Researcher in Prof. Marian Verhelst's Team with KU Leuven, Belgium. His main research interest is the design of ultra-low-power digital and mixed-signal ICs for edge-AI chips, with particular interest for HW/SW co-design aspects, novel accelerator architectures and dataflows, as well as SRAM-based in-memory computing. Recently, he has also shown a rising interest for event-based computing techniques. He is the author or co-author of several research papers in IEEE conferences and journals, receiving the Best Student Paper Award for the 2022's ESSCIRC-ESSDERC conference. He also serves as a Reviewer for various IEEE SSCS and CAS journals, wherein IEEE ESSCERC, ISCAS, IEEE JOURNAL OF SOLID-STATE CIRCUITS, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I AND II. He was also a Student Representative to the IEEE Benelux Section for 2020–2023.

Martin Lefebvre (Member, IEEE) received the M.Sc. and Ph.D. degrees in engineering sciences from the Université catholique de Louvain (UCLouvain), Belgium, in 2017 and 2024, respectively. Currently, he is a Postdoctoral Researcher with the cognitive sensor nodes and systems (CogSys) laboratory led by Prof. C. Frenkel with Delft University of Technology (TU Delft), The Netherlands, working on neuromorphic hardware/software co-design for efficient on-chip learning. His research interests include hardware-aware machine learning algorithms, mixed-signal vision chips for embedded image processing, and low-power current reference architectures. He serves as a Reviewer for various IEEE journals and conferences including IEEE JOURNAL OF SOLID-STATE CIRCUITS and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I AND II.

Pol Maistriaux (Graduate Student Member, IEEE) received the M.Sc. degree in electrical engineering from the Université catholique de Louvain (UCLouvain), Louvain-la-Neuve, Belgium, in 2021, where he is currently working toward the Ph.D. degree with the Electronics Circuits and Systems Group, under the supervision of Prof. D. Bol. His current research interests include digital IC design and ultra-low-power receivers to enable large-scale mesh sensor networks for environmental monitoring applications.

David Bol (Senior Member, IEEE) received the M.Sc. degree in electromechanical engineering and the Ph.D. degree in engineering science from the Université catholique de Louvain (UCLouvain), Louvain-la-Neuve, Belgium, in 2004 and 2008, respectively. In 2005, he was a visiting Ph.D. student at the CNM National Centre for Microelectronics, Sevilla, Spain, in advanced logic design. In 2009, he was a Postdoctoral Researcher with intoPIX, Louvain-la-Neuve, Belgium, in low-power design for JPEG2000 image processing. In 2010, he was a Visiting Postdoctoral Researcher with the UC Berkeley Laboratory for Manufacturing and Sustainability, Berkeley, CA, in life-cycle assessment of the semiconductor environmental impact. Currently, he is an Associate Professor with UCLouvain. In 2015, he participated to the creation of e-peas semiconductors, Louvain-la-Neuve, Belgium. He leads the Electronic Circuits and Systems (ECS) Group focused on ultra-low-power design of integrated circuits for the IoT and biomedical applications including computing, power management, sensing, and wireless communications. He is engaged in a social-ecological transition in the field of ICT research. He co-teaches four M.S. courses in electrical engineering with UCLouvain on digital, analog and mixed-signal integrated circuits and systems as well as sensors, with two B.S. courses including the course on sustainable development and transition.