An SRAM-Based Multibit In-Memory Matrix-Vector Multiplier With a Precision That Scales Linearly in Area, Time, and Power

Riduan Khaddam-Aljameh, *Graduate Student Member, IEEE*, Pier-Andrea Francese, *Senior Member, IEEE*, Luca Benini, *Fellow, IEEE*, and Evangelos Eleftheriou, *Life Fellow, IEEE*

***Abstract*— A novel interleaved switched-capacitor and SRAM-based multibit matrix-vector multiply-accumulate engine for in-memory computing is presented. Its operation principle is based on first converting an SRAM-stored n-bit weight into a proportional voltage using a pipeline D/A converter built from $n + 1$ equally sized stages. A switched-capacitor stage then multiplies these voltages with an m-bit digital input activation. Finally, the output voltages that correspond to the different multiplication results are accumulated along one column by means of charge-sharing. With our proposed architecture, the required circuit area, computation time, and power consumption scale linearly versus the bit resolution of both the inputs and the weights. Analytical formulas are presented for the energy consumption in both capacitors and switches. Moreover, the impact of fabrication mismatch on analog computation accuracy is examined. The full system architecture is described, and the feasibility is demonstrated, via a full macroimplementation study in 14 nm, detailing area and energy consumption, as well as the overall latency. Finally, a specific design of a 128 × 2048 6-bit weight and 6-bit input signed matrix-vector multiplication accelerator system in 14 nm is presented, which runs at 2.43 TOP/s at an efficiency of 16.94 TOP/s/W, while using the nominal supply voltage of 0.8 V. If the operands' precision is considered in the metric, then the efficiency becomes 609.7 TOP/s/W.**

***Index Terms*— Analog computation, hardware accelerator, in-memory computation, multibit weights, SRAM.**

I. Introduction

PERFORMING computations on the conventional von Neumann computing systems results in a significant amount of data being moved back and forth between the physically separated memory and processing units. This costs time and energy and constitutes an inherent performance bottleneck. Overcoming the restrictions of the classical von-Neumann-based architectures, which enforce a dogmatic separation of the processing unit and the memory subsystem, requires reevaluating the well-established charge-based memory technologies, such as SRAM, DRAM, and Flash [1]–[3], as well as the emerging resistance-based nonvolatile memory technologies [4], [5].

It is becoming increasingly clear that, for application areas, such as artificial intelligence (AI), we need to transition to computing architectures in which logic and memory are colocated [6]. In-memory computing (IMC) is a novel non-von Neumann computing paradigm where certain computational tasks are performed in the memory itself by exploiting the physical attributes and state dynamics of the charge- or resistance-based memory devices [6]. Several computational tasks, such as logical operations, arithmetic operations, and even certain machine learning tasks, can be implemented in such an IMC-based system. As a result, the execution time, energy consumption, and silicon area of an IMC-based architecture are reduced compared to von-Neumann architectures, yielding a compact and highly efficient system [7].

Generally, IMC-approaches perform computations with relatively low numerical precision. Hence, IMC does not aim to replace digital floating-point arithmetic units and, instead, targets applications, such as deep neural network inference, which are resilient to low precision. To reach the numerical accuracy typically required for data analytics and scientific computing, the limitations arising from device variability and nonideal device characteristics need to be addressed. Thus, the concept of mixed-precision in-memory computing, which combines a conventional high-precision von Neumann machine with IMC, was introduced in [8]. Its application to deep neural network training was proposed in [7] and [9].

In IMC, the physics of the nanoscale memory devices and the organization of such devices in crossbar arrays are exploited to perform certain computational tasks within the memory unit. For instance, crossbar arrays of phase change memory (PCM) and resistive random access memory (ReRAM) devices can be used to store a matrix and perform analog matrix-vector multiplications at constant $\mathcal{O}(1)$ time complexity without intermediate movements of data.

Manuscript received April 20, 2020; revised August 25, 2020 and October 19, 2020; accepted November 9, 2020. Date of publication December 9, 2020; date of current version January 28, 2021. The work of Riduan Khaddam-Aljameh was supported by the European Research Council (ERC) through the European Union's Horizon 2020 Research and Innovation Programme under Grant 682675. *(Corresponding author: Riduan Khaddam-Aljameh.)*

Riduan Khaddam-Aljameh is with the IBM Zurich Research Laboratory, 8803 Rüschlikon, Switzerland, and also with the Integrated Systems Laboratory, ETH Zurich, 8092 Zürich, Switzerland (e-mail: rid@zurich.ibm.com).

Pier-Andrea Francese and Evangelos Eleftheriou are with the IBM Zurich Research Laboratory, 8803 Rüschlikon, Switzerland (e-mail: pfr@zurich.ibm.com; ele@zurich.ibm.com).

Luca Benini is with the Integrated Systems Laboratory, ETH Zurich, 8092 Zürich, Switzerland, and also with the Department of Electrical, Electronic and Information Engineering, University of Bologna, 40136 Bologna, Italy.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2020.3037871.

Digital Object Identifier 10.1109/TVLSI.2020.3037871

Specifically, modulation of the read voltage amplitude or duration that is applied to a PCM cell leads to a proportional change in the current that flows through it. This property can be used to perform an analog-domain multiplication with the value stored in the PCM cell in the form of a conductance. If many cells are activated on the same line, their respective currents are summed up in accordance with Kirchhoff's law. Thus, by combining the multilevel storage capability of PCM with Ohm's and Kirchhoff's laws, an analog IMC multiply-accumulate (MAC) engine can be constructed [10].

Besides the classical analog-domain processing issues, such as thermal- and A/D converter (ADC) noise, memristor-based computation is, furthermore, subject to drift and $1/f$ noise [11], which are issues that can only be addressed through material and device innovations [12]. In addition, the large-scale current summations require wide and highly conductive bit-lines, which leads to a reduced area efficiency [13].

Static random access memory (SRAM) can in a similar way to PCM and ReRAM be enhanced with IMC capabilities. A native approach consists of using the ON-resistance $R_{\mathrm{DS,ON}}$ of the pull-down transistors to convert the digital bits stored in the SRAM cells into a proportional current value [14]–[16]. However, serious practical limitations of $R_{\mathrm{DS,ON}}$-based IMC in SRAM arise from cell-to-cell variations, as well as from the risk of unintended overwriting of stored bits. To address these issues, charge-based analog computation has been explored as an alternative to those that are current-based, due to the much better matching of back-end-of-the-line (BEOL) capacitors. This feature is also expected to improve with CMOS scaling.

For example, the 8T SRAM cell in [17] is enhanced by a transmission-gate (TG) and a single capacitor giving rise to a 10T + C unit cell structure. Thus, by introducing the highly matched metal capacitors, the IMC design eliminates mismatch-ridden minimum-size transistors. However, the elements of the input vector and synaptic matrix of this XNOR-based IMC engine are limited to binary values only.

A solution for supporting at least multibit inputs is presented in [18]. Nevertheless, the weights are still kept in binary format. Moreover, since a partially pulsewidth-modulation (PWM)-based D/A converter (DAC) is used, the relationship between the number of input bits and the computation cycle time becomes inherently exponential, significantly impacting the latency. An example of a near-memory computing architecture that supports multibit inputs and synaptic weights is described in [19]. However, bandwidth limitations also arise here due to the restriction to row-by-row processing and the PWM-based digital-to-analog (D/A) conversion procedure for mapping the matrix values to voltages. Finally, another multibit near-memory computing architecture, based on a passive switched-capacitor approach, is described in [20]. Here, the exponential area requirements for the flying capacitors used in the implementation of the input DAC prevent an efficient extension of this approach to large-scale in-memory computing (IMC) arrays.

As can be seen from the aforementioned recent work, severe limitations in performance, precision, scalability, and circuit complexity have arisen in all CMOS-based IMC implementations, as soon as support for more than binary quantization for both the input signals and the synaptic weights was required. This article proposes a novel SRAM-based architecture supporting in-memory MAC-operations, where both input signals and matrix elements can assume multibit values while still exhibiting a linear dependence between cycle time, silicon area, power consumption, and the number of bits of inputs and weights.

The remainder of this article is organized as follows. In Section II, the in-memory computation circuit used for the analog MAC operation is presented, and an analytical model describing the functionality is given. Moreover, the energy consumption of an analog multiply operation is analyzed in detail, and the effect of mismatch is examined. The full memory architecture is described in Section III. An examination of the computational memory system's area and the energy consumption is presented in Section IV based on an implementation study. In addition, the performance is compared with the state of the art. Finally, Section VI concludes this article.

II. Multibit In-Memory Compute Unit

In essence, the energy and bandwidth gains achieved through IMC stem from moving digital arithmetic and logic operations into the analog domain. As implied by the term "in-memory," all computational primitives are executed within the memory subsystem, as opposed to near-memory computing architectures that keep the processing units separated at close proximity.

Matrix-vector multiplication and transpose-matrix-vector multiplication are among the most important computational primitives for machine learning and deep learning applications. Thus, the goal of this article is the acceleration of MAC operations, which can be written in the form of

$$\vec{x}^{\mathsf{T}} \times \boldsymbol{A} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}^{\mathsf{T}} \times \begin{pmatrix} w_{1,1} & \cdots & w_{1,M} \\ \vdots & \ddots & \vdots \\ w_{N,1} & \cdots & w_{N,M} \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_M \end{pmatrix}^{\mathsf{T}} = \vec{y}^{\mathsf{T}}$$

where $^{\mathsf{T}}$ denotes the transpose vector or matrix operator. Each element y_m of the vector $\vec{y}$ can be written as a sum of N products

$$y_m = \sum_{n=1}^{N} w_{n,m} \cdot x_n, \quad m = 1, \ldots, M. \tag{1}$$

Fig. 1 contrasts the conventional von-Neumann architecture, where the memory and the processing units are separated, to the in-memory computing architecture. In the latter, the matrix elements that are stored in SRAM cells remain stationary, whereas processing occurs in the so-called in-memory compute units (IMCUs), which are collocated with the SRAM. Specifically, the stationary matrix elements $w_{n,m}$ (called weights) are stored in the SRAM array, and the inputs x_n are fed from outside to the IMCU. Both inputs and weights are assumed to be signed quantities. For a given fixed-point arithmetic precision, the elements $w_{n,m}$ are stored in binary form wordwise, and in column-major format, into the SRAM array, as shown in Fig. 1(b). Thus, the SRAM cells, which store the binary representation of weight, are collocated with one IMCU in a crossbar configuration.

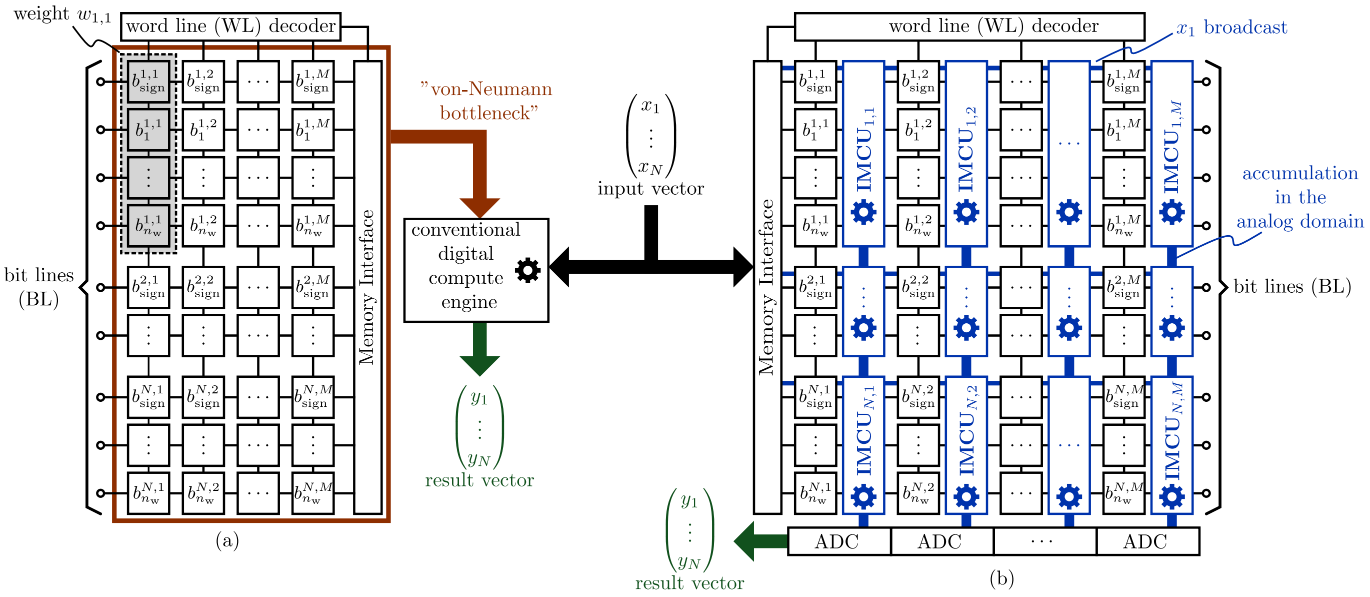


Fig. 1. (a) Conventional von-Neumann architecture with distinct memory and processing units. To perform MAC operations, the matrix elements, which are stored in the memory unit, need to be transferred to the physically separated digital compute unit. This costs time and energy, creating an inherent performance bottleneck. (b) In IMC systems, memory and processing units are collocated. Thus, the matrix elements that are stored in SRAM cells remain stationary, whereas processing occurs in the so-called IMCUs, which are collocated with the SRAM, in a crossbar configuration.

Using signed magnitude representation (SMR), the following correspondence between a weight value $w_{n,m}$ and the stored bits $b_k^{n,m}$ is established

$$w_{n,m} = \left(\sum_{k=1}^{n_\text{w}} b_k^{n,m} \cdot 2^{k-n_\text{w}-1}\right) \cdot \begin{cases} +1, & \text{if } b_{\text{sign}}^{n,m} = 0 \\ -1, & \text{otherwise.} \end{cases} \tag{2}$$

In the following sections, for simplicity, the column index m will be dropped from $w_{n,m}$. Hence, without any loss of generality, the MAC-operation will be illustrated for a single column only. Similarly, the corresponding weight bits will be denoted by b_k^n and the input bits by i_p^n. Moreover, the magnitude of a weight w_n will be represented by n_w bits and the magnitude of the input x_n by n_x bits. The binary variables b_{sign}^n and i_{sign}^n denote the respective sign bits. Finally, 8T SRAM cells will be used to store the weight bits.

The two main blocks of the IMCU, namely, the DAC for transforming the binary weight into a voltage value and the analog multiplier, will be detailed in the subsequent sections. The combination of both circuits will be referred to as an interleaved switched-capacitor-based multiplier.

A. Matrix Value D/A Conversion

The initial step in the analog computation procedure consists of performing a D/A conversion, which turns the stored weight bits b_k^n into a proportional voltage $V_{\text{w},n}$. This can be achieved by connecting the digitally stored weight bits from the SRAM cells to the inputs of a DAC. Fig. 2(a) demonstrates an implementation built from equally sized unit capacitors C_k, interconnected through switches. The total amount of required unit capacitors in the pipeline DAC scales linearly with the number of stored bits n_w in the weight magnitude

$$n_{\text{unitcap,w}} = n_\text{w} + 1. \tag{3}$$

The circuit is built around the quasi-passive charge-sharing DAC design presented in [21] and [22] to which two key elements have been added to realize the proposed IMCU. First, a dynamic precharge voltage selection has been integrated to support SMR. Second, a single additional switched-capacitor stage unit is connected in cascade to perform the multiplication of the D/A converted digital weight with the input data.

To conduct the D/A conversion, a set of three nonoverlapping digital pulse signals ϕ_0, ϕ_1, and ϕ_2 is used. Each SRAM cell, containing one bit b_k^n of the weight, controls one stage in the pipeline DAC. Based on this weight bit b_k^n, the corresponding capacitor C_k is initially precharged to either a voltage V_{pre} or to the common-mode V_{CM}. Next, the top plates of capacitor C_k and its predecessor C_{k-1} are shorted, effectively averaging their voltages

$$V_{\text{C},k} = 0.5 \cdot \left(b_k^n \cdot V_{\text{pre}} + V_{\text{C},k-1}\right). \tag{4}$$

The first capacitor C_0 is always precharged to V_{CM} such that the significance of the various bits is respected during the subsequent voltage-sharing operations. This procedure can be continued until all magnitude bits of the weight are processed, finally yielding the weight-proportional voltage $V_{\text{w},n}$ on the last capacitor C_{n_w} after $n_{\text{cyc,w}}$ cycles

$$V_{C_{n_\text{w}}}[n_{\text{cyc,w}}] = V_{\text{w},n} = V_{\text{pre}} \cdot \sum_{k=1}^{n_\text{w}} b_k^n \cdot 2^{k-n_\text{w}-1}. \tag{5}$$

The voltage V_{pre} is selected based on the desired sign of $V_{\text{w},n}$ so that both positive and negative weights can be represented in the analog domain. In total, a number of $n_{\text{cyc,w}}$ cycles is

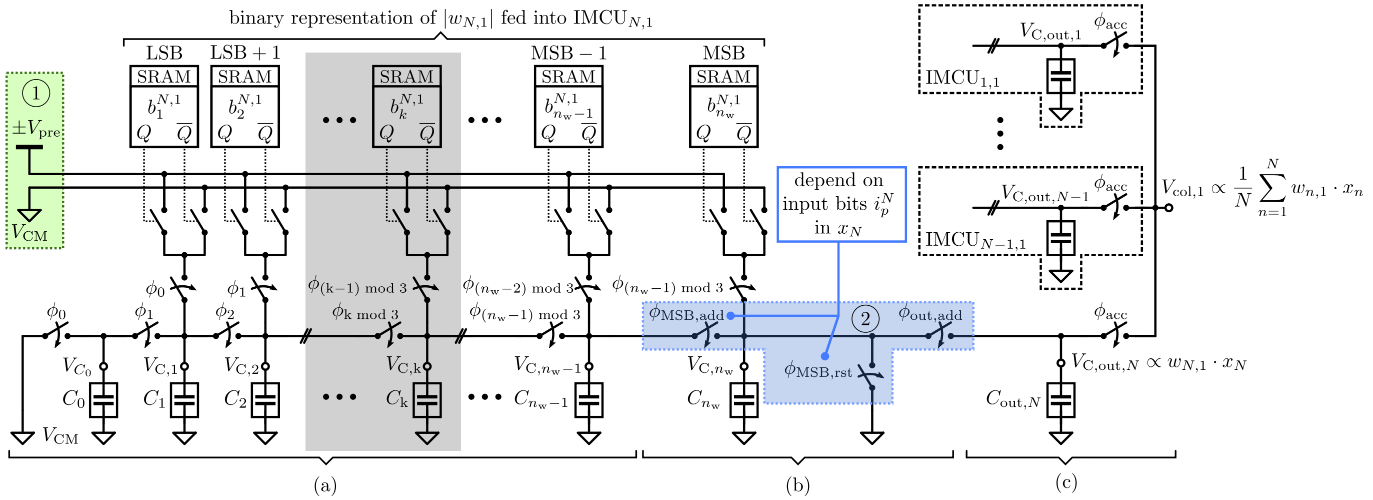


Fig. 2. Schematic of the interleaved switched-capacitor-based multibit in-memory compute unit (IMCU) that allows the execution of matrix-vector multiplications. The pipelined D/A converter in (a) generates a voltage, which is proportional to the stored weight bits $\{b_1^{N,1}, b_2^{N,1}, \ldots, b_{n_w}^{N,1}\}$, representing the unsigned weight $|w_{N,1}|$. Note that the sign of the precharge voltage in ① is selected based on the sign of both input and weight. (b) Analog multiplier performs a multibit multiplication as a series of binary multiplication steps, by controlling the switches in ②. Based on each input bit, either zero or a weight proportional amount of charge is added to the output capacitor $C_{\mathrm{out,N}}$. (c) Analog accumulator performs the summation of all multiplication results of the IMCUs along the first column by means of charge-sharing. All capacitors employed throughout the process are of equal size.

required until the voltage $V_{\mathrm{w},n}$ is generated, whereby $n_{\mathrm{cyc,w}}$ depends linearly on the number of bits in the weight

$$n_{\mathrm{cyc,w}} = n_{\mathrm{w}} + 1. \tag{6}$$

Once the first valid $V_{\mathrm{w},n}$ voltage is obtained on the last capacitor, pipelining allows consecutive replication of $V_{\mathrm{w},n}$ every three clock cycles. This is required to perform the analog multiplication, described in the next section, at high bandwidth.

B. Analog Multiplication

As a next step, the analog multiplication of the voltage, which is proportional to the weight, with the input must be performed. It is assumed, similar to the weight w_n, that the input x_n is represented as an n_{x}-bit fixed-point number in SMR. Thus, the sign of the multiplication result s_{result}^n can be immediately obtained through an XOR operation between the respective signs of weight (b_{sign}^n) and input (i_{sign}^n)

$$s_{\mathrm{result}}^n = \begin{cases} +1, & \text{if } \left(b_{\mathrm{sign}}^n \oplus i_{\mathrm{sign}}^n = 0\right) \\ -1, & \text{otherwise.} \end{cases} \tag{7}$$

Consequently, the precharge voltage V_{pre} can be selected based on s_{result}^n. The corresponding circuit implementation is shown in Fig. 3, which illustrates an example of a transistor-level implementation of an SRAM-based 3-bit signed IMCU. In the case of an unsigned multiplication, the precharge voltage selection step and the related circuitry can be omitted. Furthermore, if the distributive law is used, a multibit fixed-point multiplication of an input x_n by a weight w_n can be reformulated as a sum of n_{x} binary products

$$s_{\mathrm{result}}^n \cdot |w_n \cdot x_n| = s_{\mathrm{result}}^n \cdot |w_n| \cdot \sum_{p=1}^{n_{\mathrm{x}}} \left(i_p^n \cdot 2^{-p}\right). \tag{8}$$

Therefore, the multiplication can be carried out successively in n_{x} multiply and add steps while going through the input bits one by one. In hardware, this can be optimally implemented by modifying the control signals on the switches of the MSB capacitor $C_{n_{\mathrm{w}}}$, which, at the $n_{\mathrm{cyc,w}}$-th cycle, is charged to $V_{\mathrm{w},n}$

$$\phi_{\mathrm{MSB,add}} = i_p^n \text{ AND } \phi_{(n_{\mathrm{w}})\bmod 3} \tag{9}$$

$$\phi_{\mathrm{MSB,rst}} = \overline{i_p^n} \text{ AND } \phi_{(n_{\mathrm{w}})\bmod 3}. \tag{10}$$

The key advantage of the proposed implementation is that, in an IMC system, these two signals need only be generated once for the full row and not on a per matrix element. Depending on the input bit i_p^n, in every three cycles, the capacitor $C_{n_{\mathrm{w}}}$ either produces the weight proportional voltage $V_{\mathrm{w},n}$ if $i_p^n = 1$ or otherwise zero

$$V_{\mathrm{C},n_{\mathrm{w}}}[n_{\mathrm{cyc,w}} + 3(p-1)] = i_p^n \cdot V_{\mathrm{w},n}, \quad p = 1, \ldots, n_{\mathrm{x}}. \tag{11}$$

This binary multiplication result is then accumulated via charge-sharing on a dedicated capacitor $C_{\mathrm{out},n}$. Starting from an initial voltage $V_{\mathrm{C,out},n}^0 = V_{\mathrm{CM}}$ on $C_{\mathrm{out},n}$, each step of charge-sharing halves the voltage on $C_{\mathrm{out},n}$ and, depending on the current input bit, adds $0.5 \cdot V_{\mathrm{w},n}$ to it

$$V_{\mathrm{C,out},n}^p = 0.5 \cdot \left(i_p^n \cdot V_{\mathrm{w},n} + V_{\mathrm{C,out},n}^{p-1}\right). \tag{12}$$

The input bits need to be traversed from LSB to MSB to ensure that the added charge corresponds to the respective bit's significance. Conversely, traversing in the opposite direction, i.e., from MSB to LSB, would require doubling a charge and adding it to that of lower significance, which is nontrivial in terms of a circuit design implementation.

The accumulation of the binary multiplication can start as soon as the first pipelined D/A conversion is flushed. To this end, the signal d_{valid} indicates, after $n_{\mathrm{cyc,w}}$ cycles, that the correct voltage $V_{\mathrm{w},n}$ is available, after which the accumulation

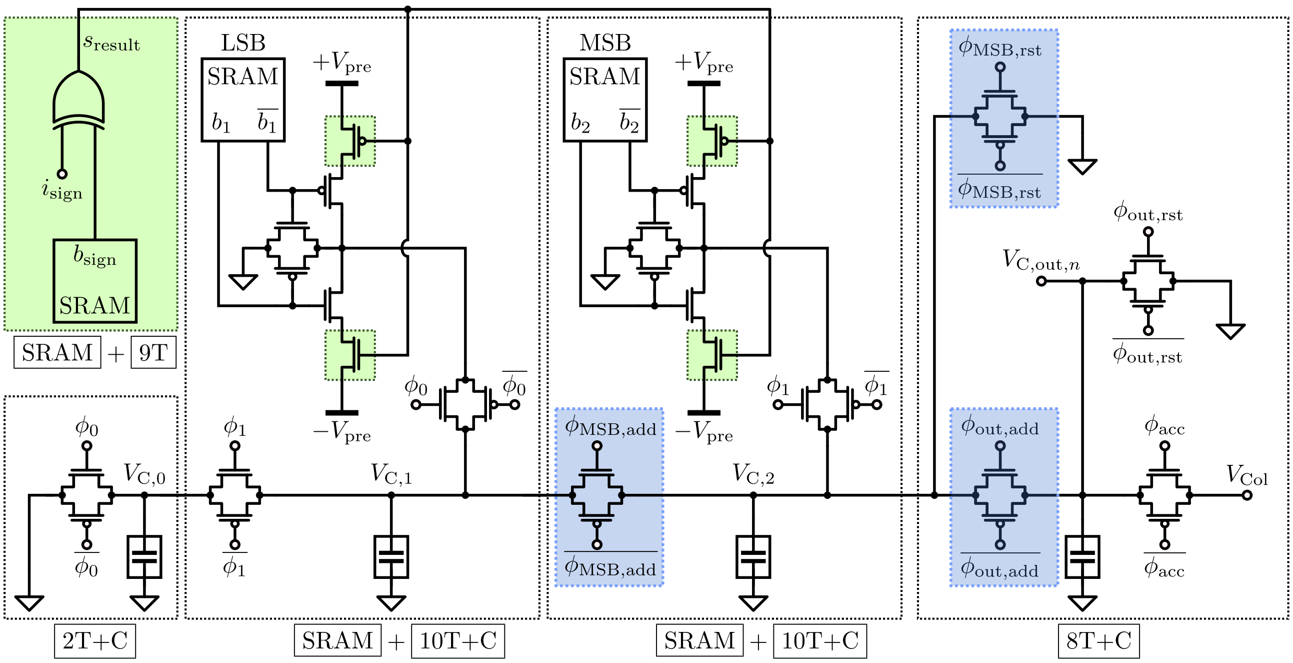


Fig. 3. Transistor-level implementation of a 3-bit signed in-memory compute unit (IMCU) in combination with SRAM cells. All the transmission-gates (TGs) are comprised of a pMOS and nMOS pair so that the charge-sharing procedures are rapidly executed for the full voltage range. In the shown configuration, both 6T and 8T SRAM cells would be supported since the IMCU is not multiplexed to be shared with several memory cells. The XOR logic element is implemented here using nine transistors.

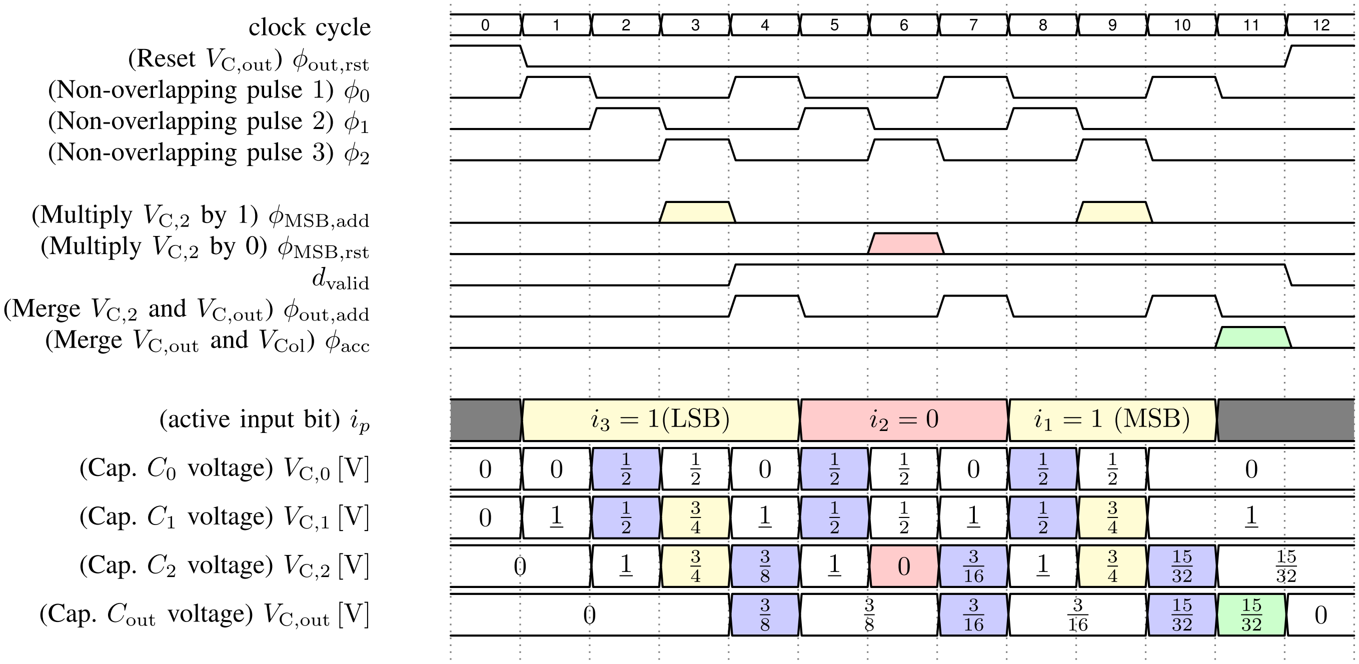


Fig. 4. Timing diagram for the control signals and capacitor voltages of an IMCU multiplying 3-bit signed weight $w = -3 = 1|11_2$ with 4-bit signed input $x = -5 = 1|101_2$. Charge-sharing cycles are highlighted in blue and binary multiplications of the MSB capacitor voltage with 1 or 0 are in yellow and red, respectively. Completion of the MAC procedure is signaled by ϕ_{acc}, highlighted in green. The ADC sample signal is not shown here. Cycles during which energy is drawn from the precharge voltage supplies are indicated by the underlined capacitor voltage values.

is initiated

$$d_{\text{valid}} = \begin{cases} 1, & \text{if } (t > n_{\text{cyc,w}}) \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

$$\phi_{\text{out,add}} = d_{\text{valid}} \text{ AND } \phi_{(n_{\text{w}}+1) \bmod 3}. \tag{14}$$

To summarize, an additional input bit is processed every three clock cycles, until all n_{x} bits of the input magnitude have been multiplied and accumulated. The required number of cycles is obtained with the following formula:

$$n_{\text{cyc,i}} = 3 \cdot (n_{\text{x}} - 1) + 1. \tag{15}$$

Note that the first input bit is processed one cycle after the pipeline DAC settles the first time. Ultimately, the following correspondence between a final output voltage $V_{\mathrm{C,out,final},n}$ and the multiplicands w_n and x_n is established

$$V_{\mathrm{C,out,final},n} = s^n_{\mathrm{result}} \cdot \frac{x_n}{2^{n_{\mathrm{x}}}} \cdot \frac{w_n}{2^{n_{\mathrm{w}}}} \cdot V_{\mathrm{pre}} + V_{\mathrm{CM}}. \tag{16}$$

C. Analog Accumulation

At this stage, all multiplications between the rows of the weight matrix and the input data vector have been executed. The last operation needed to complete the matrix-vector multiplication is to sum up the results along each column. In the analog domain, this is achieved by shorting all output capacitors along one column to the node V_{col} using dedicated switches controlled by the ϕ_{acc} signal. Since only one capacitor size is used throughout the entire array, the respective voltages $V_{\mathrm{C,out,final},n}$ are averaged

$$V_{\mathrm{col}} = \frac{1}{N} \cdot \sum_{n=1}^{N} V_{\mathrm{C,out,final},n}. \tag{17}$$

Fig. 2(c) shows how the analog accumulator performs the summation of all the multiplication results of the IMCUs along the first column of the crossbar array.

The number of cycles $n_{\mathrm{cyc,acc}}$ required to finalize the accumulation is given by the RC time constant of the charge-sharing procedure. Given the high number of switches that are used and the relatively small unit capacitance size, a settling time of $n_{\mathrm{cyc,acc}} = 1$ will be assumed. All the resulting column voltages are proportional to the entries in the result vector $\vec{y}$ and can finally be digitized using integrated ADCs. By including the number of cycles $n_{\mathrm{cyc,adc}}$ that the ADC needs to sample V_{col}, the following equation for the total number of cycles needed for the in-memory MAC operation can be defined

$$\begin{aligned} n_{\mathrm{cyc}} &= n_{\mathrm{cyc,w}} + n_{\mathrm{cyc,i}} + n_{\mathrm{cyc,acc}} + n_{\mathrm{cyc,adc}} + n_{\mathrm{cyc,rst}} \\ &= n_{\mathrm{w}} + 3 \cdot n_{\mathrm{x}} + 2 \end{aligned} \tag{18}$$

where, in one cycle, $n_{\mathrm{cyc,rst}} = 1$ is used to reset the full system. Moreover, the assumption is made that the ADC sampling time for an 8-bit output resolution is below the digital circuits' cycle time [23] so that $n_{\mathrm{cyc,adc}} = 1$. Note that (18) gives evidence for the linear relation between latency and the number of weight and input bits.

D. Numerical Example

In the following, a simple numerical example will be given to demonstrate the functionality of the MAC circuitry. To this end, the weights will be quantized as 3-bit signed fixed-point numbers ($n_{\mathrm{w}} = 2$), and 4-bit signed quantization will be used for the inputs ($n_{\mathrm{x}} = 3$). A possible transistor-level implementation with appropriate control signals is given in Fig. 3. Waveforms and capacitor voltage evolution throughout the analog multiplication process are given in Fig. 4. For the reasons of simplicity, the common-mode voltage will be defined as 0 V, and the precharge voltages will be set to ± 1 V. A weight value of $w = -3$ and an input of $x = -5$ will be used

$$\begin{aligned} w &= -3 = 1|11_2 = (-1) \cdot [b_1, b_2] \times [2, 1]^{\mathsf{T}} \\ x &= -5 = 1|101_2 = (-1) \cdot [i_1, i_2, i_3] \times [4, 2, 1]^{\mathsf{T}}. \end{aligned}$$

Using (7), the sign $s_{\mathrm{result}} = +1$ of the multiplication result is determined immediately. As the result will be positive, the pipeline D/A will only use the positive precharge voltage $+V_{\mathrm{pre}}$. The pipeline D/A starts synthesizing the weight voltage V_{w} from the magnitude bits b_1 and b_2. Accordingly, the first capacitor (LSB) V_{C1} settles on a voltage $1/2$ V and the second one on $V_{\mathrm{C2}} = 1/2 \cdot (1\ \mathrm{V} + 1/2\ \mathrm{V}) = 3/4\ \mathrm{V} = V_{\mathrm{w}}$. In accordance with the first input bit that is processed ($i_3 = 1$), this voltage is accumulated on the output capacitor in cycle 4, thus yielding

$$V_{\mathrm{C,out}}[4] = 1/2 \cdot 3/4\ \mathrm{V} = 3/8\ \mathrm{V}.$$

Since the second input bit is zero, the MSB capacitor V_{C2} is discharged in cycle 6, after which the common-mode voltage V_{CM} is merged with $V_{\mathrm{C,out}}$ in cycle 7

$$V_{\mathrm{C,out}}[7] = 1/2 \cdot (0\,\mathrm{V} + 3/4\ \mathrm{V}) = 3/16\ \mathrm{V}.$$

Finally, $V_{\mathrm{w},n}$ is merged a second time with $V_{\mathrm{C,out}}$ in cycle 10, thus processing the last input bit $i_1 = 1$

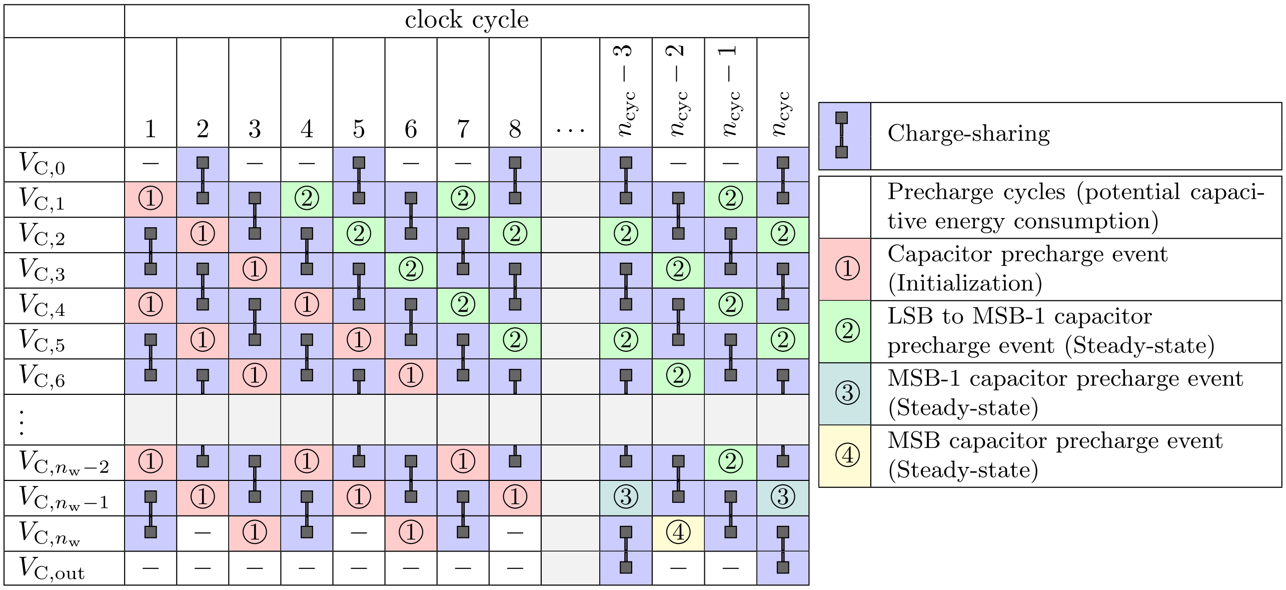
$$V_{\mathrm{C,out}}[10] = 1/2 \cdot (3/4\ \mathrm{V} + 3/16\ \mathrm{V}) = 15/32\ \mathrm{V}.$$

This value can also be obtained via (16) and corresponds to the final result of the multiplication operation. Had the sign been negative, the negative precharge voltage $-V_{\mathrm{pre}}$ would have been used, and $V_{\mathrm{C,out}}[10]$ would have been $-15/32$ V.

E. Energy Consumption

In order to quantify the energy consumed by the charge-based analog multiplier, one full operational cycle will be analyzed. Since only one single IMCU will be examined, the row index n will be omitted in the next sections for simplicity. The basic circuit operation of each 10T + C cell can be summarized in three steps: an initial unit-capacitor precharge or discharge cycle depending on weight (b_k) and input bits (i_p) and then two charge-sharing procedures: first, with the previous capacitor, and then, with the consecutive capacitor. Note that charge-sharing itself is quasi-passive and consumes no energy except when switching the connected TG. Only when the capacitors are precharged from the $\pm V_{\mathrm{pre}}$ supplies, electrical energy is consumed. In addition to data-dependent energy consumption occurring during precharge events, there is a data-independent part caused by the switching events of the TGs. Both contributions to the circuit's energy consumption will be examined in the following paragraphs, and the corresponding analytical formulas will be presented.

1) Precharge Events During Initialization: Assuming all unit capacitors C_k in the circuit are initially reset to the common-mode voltage V_{CM}, the energy drawn during the initial precharge cycles differs from energy drawn once the D/A circuit operates in steady state. The following formula



	clock cycle												
	1	2	3	4	5	6	7	8	…	$n_{\text{cyc}}-3$	$n_{\text{cyc}}-2$	$n_{\text{cyc}}-1$	n_{cyc}
$V_{\text{C},0}$	–	■	–	–	■	–	–	■		■	–	–	■
$V_{\text{C},1}$	①	■	■	②	■	■	②	■		■	■	②	■
$V_{\text{C},2}$	■	①	■	■	②	■	■	②		②	■	■	②
$V_{\text{C},3}$	■	■	①	■	■	②	■	■		■	②	■	■
$V_{\text{C},4}$	①	■	■	①	■	■	②	■		■	■	②	■
$V_{\text{C},5}$	■	①	■	■	①	■	■	②		②	■	■	②
$V_{\text{C},6}$	■	■	①	■	■	①	■	■		■	②	■	■
⋮													
$V_{\text{C},n_{\text{w}}-2}$	①	■	■	①	■	■	①	■		■	■	②	■
$V_{\text{C},n_{\text{w}}-1}$	■	①	■	■	①	■	■	①		③	■	■	③
$V_{\text{C},n_{\text{w}}}$	■	–	①	■	–	①	■	–		■	④	■	■
$V_{\text{C,out}}$	–	–	–	–	–	–	–	–		■	–	–	■

■	Charge-sharing
	Precharge cycles (potential capacitive energy consumption)
①	Capacitor precharge event (Initialization)
②	LSB to MSB-1 capacitor precharge event (Steady-state)
③	MSB-1 capacitor precharge event (Steady-state)
④	MSB capacitor precharge event (Steady-state)

Fig. 5. Diagram indicating all events throughout an IMCU's full operational cycle that potentially results in charge transfer from the supplies to the unit capacitors. $V_{\text{C},0}$ to $V_{\text{C,out}}$ denote the voltages on each of the unit capacitors. Note the circuit's mode of operation, consisting of a precharge event followed by two charge-sharing procedures with the previous and subsequent unit capacitor. The different precharge cycles are categorized from ① to ④, depending on whether they occur during initialization or steady state and also depending on the capacitor number.

characterizes the capacitive energy consumed during any precharge event:

$$E_{\text{C},k} = \frac{1}{2} \cdot C_{\text{unit}} \cdot b_k \cdot (V_{\text{pre}} - V_{\text{C,L}})^2 \tag{19}$$

where C_{unit} denotes the unit capacitor size, $V_{\text{C,L}}$ represents the capacitor's last voltage before the precharge event occurs, and b_k indicates the fact that only precharging to nonzero-bit values consumes energy. By using the pipelined DAC output voltage equation from (5), $V_{\text{C,L}}$ can be determined as

$$V_{\text{C,L}}(k_1, k_2) = V_{\text{pre}} \cdot \sum_{a=k_1}^{k_2} b_{\text{a}} \cdot 2^{a-1-k_2}. \tag{20}$$

This voltage corresponds to the pipeline DAC output at the bit number k_2, where the bits of higher significance from k_2+1 to n_{w} have not yet been processed. To capture the case of an uninitialized pipeline D/A, the parameter k_1 is introduced. This missing initialization can be observed in Fig. 5, where the initial voltages in cycle 1 on all capacitors except C_1 do not contain any LSB information. Since all capacitors are initially reset to the common-mode voltage V_{CM}, these unprocessed bits are assumed to be zero for all voltages that are developed on the subsequent capacitors of the pipeline DAC via charge-sharing. The total number of these initialization pipeline runs $n_{\text{r,init}}$ depends on the number of weight bits n_{w} used in the IMCU

$$n_{\text{r,init}} = \left\lceil \frac{n_{\text{w}}}{3} \right\rceil. \tag{21}$$

In each of these runs, a number of LSBs is missing, depending on the position of the start bit. For the runs starting in cycle 1 at capacitor 1, no bits are omitted, and for the run starting at capacitor 4, bits 1–3 are missing. Furthermore, due to the special signals of the analog multiplier [see (9) and (10)], the MSB capacitor is not precharged when the first input bit is zero ($i_1 = 0$). Finally, the capacitor voltage prior to a precharge event can be given in dependence of $V_{\text{C,L}}$ for a run number r and unit capacitor number k_{x}

$$V_{\text{C,L,i}}(r, k_{\text{x}}) = V_{\text{C,L}}(3r+1, \min\{k_{\text{x}}+1, n_{\text{w}} - \overline{i_1}\}). \tag{22}$$

The total amount of energy dissipated in the initial phase can now be obtained as the sum of all consumed capacitive energy

$$E_{(1)} = \frac{C_{\text{unit}}}{2} \sum_{r=1}^{n_{\text{r,init}}} \sum_{k_{\text{x}}=3r-2}^{n_{\text{w}}} b_{k_{\text{x}}} (V_{\text{pre}} - V_{\text{C,L,i}}(r, k_{\text{x}}))^2. \tag{23}$$

2) Steady-State Pipeline D/A Precharge Events: Once in steady state, each run in the pipeline D/A will consume the same amount of energy, except for the input-dependent power dissipation in the MSB and the MSB-1 capacitor. The input bit count n_{x} determines the number of the pipeline runs necessary to perform the analog multiplication. Note that the first bit is processed at the end of the initial phase ① in Fig. 5. Since the circuit operates in pipeline mode, an additional number of incomplete runs is initiated depending on the number of weight bits n_{w}. The number of steady-state pipeline runs $n_{\text{r,st}}$, thus, becomes

$$n_{\text{r,st}} = (n_{\text{x}} - 1) + \left\lfloor \frac{n_{\text{w}}+1}{3} \right\rfloor. \tag{24}$$

By using (6) and (15), the duration of each steady-state run, including the incomplete ones, can be determined

$$n_{\text{d,st}}(r) = \min\{n_{\text{w}} - 2, n_{\text{cyc,w}} + n_{\text{cyc,i}} - 3 \cdot r\}. \tag{25}$$

Finally, the total energy dissipated in the pipeline D/A during steady state can be written as

$$E_{(2)} = \frac{C_{\text{unit}}}{2} \sum_{r=1}^{n_{\text{r,st}}} \sum_{k_{\text{x}}=1}^{n_{\text{d,st}}(r)} b_{k_{\text{x}}} (V_{\text{pre}} - V_{\text{C,L}}(1, k_{\text{x}}+1))^2. \tag{26}$$

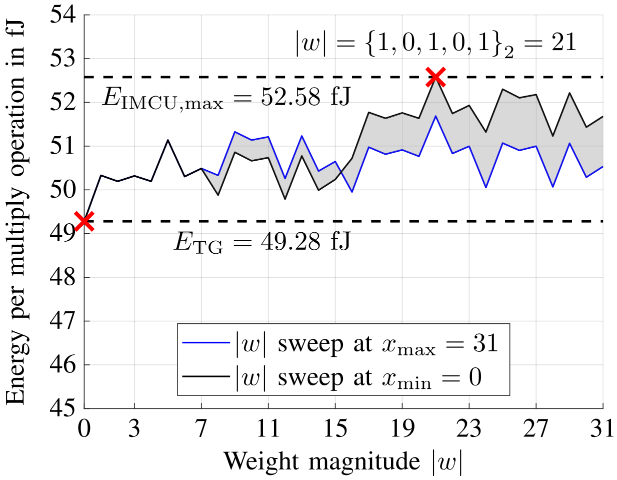


Fig. 6. Energy consumed by the multibit IMCU during one complete analog multiplication operation. The weight and input magnitudes $|w|$ and $|x|$ are both quantized to 5 bits. For $|w| = 0$, the least amount of energy E_{TG} is consumed since only the TGs operate and no energy is drawn from the $\pm V_{\text{pre}}$ supplies. The maximum amount of energy $E_{\text{IMCU,max}}$ is consumed for $|w| = 21$ and $|x| = 0$. A C_{unit} of 2 fF and a capacitive load $C_{\text{TG}} = 1$ fF were selected for the inputs of one TG in 14 nm.

3) MSB-1 Capacitor Precharge Events: Although, in steady state, the MSB-1 capacitor does not draw a constant amount of charge from the supply, its energy consumption depends on the input bit i_p that is currently processed in the analog multiplier. If this input bit i_p is equal to one, then MSB and MSB-1 capacitor are shorted, which does not occur if $i_p = 0$. The various conditions are captured in the following formula:

$$V_{\text{C,L,MSB}-1}(i_p) = V_{\text{pre}} \cdot \left(\frac{b_{n_{\text{w}}} \cdot i_p}{2} + \sum_{k=1}^{n_{\text{w}}-1} \frac{b_k \cdot 2^{k-1}}{2^{n_{\text{w}}-1+i_p}} \right). \quad (27)$$

Given that, in steady state, the MSB capacitor is precharged exactly n_{x} times, once for each input bit, the energy consumption can be computed as follows:

$$E_{(3)} = \frac{C_{\text{unit}}}{2} \sum_{x=1}^{n_{\text{x}}} b_{n_{\text{w}}-1} \cdot (V_{\text{pre}} - V_{\text{C,L,MSB}-1}(i_{\text{x}}))^2. \quad (28)$$

4) MSB Capacitor Precharge Events: Similar to the MSB-1 capacitor, the energy consumption on the MSB capacitor also exhibits an input data dependence. In steady state, the MSB capacitor is shorted to the output capacitor C_{out} to accumulate the results of the single-bit multiplication. During a subsequent precharge event, not only the current input bit i_p but also all input bits that have been multiplied and added so far must be considered when calculating the voltage on the capacitor prior to precharging. To this end, the instantaneous output capacitor voltage is defined as a function of the number of input bits that have been processed

$$V_{\text{C,out,L}}(p) = V_{\text{pre}} \cdot \sum_{a=1}^{p} i_a \cdot 2^{a-1-p} \cdot \sum_{k=1}^{n_{\text{w}}} b_k \cdot 2^{k-1-n_{\text{w}}}. \quad (29)$$

The total energy consumption in the MSB capacitor is the sum over all precharge events occurring throughout the multiplication and accumulation of each input bit

$$E_{(4)} = \frac{C_{\text{unit}}}{2} \cdot b_{n_{\text{w}}-1} \cdot \sum_{p=1}^{n_{\text{x}}} (V_{\text{pre}} - V_{\text{C,out,L}}(p))^2. \quad (30)$$

5) Switching of TGs: All TGs used in the presented analog multiplication circuit are assumed to be built exactly the same, from equally sized pMOS and nMOS transistors. While turning on the TG, the charging procedure of the nMOS transistor gate consumes energy, and while turning it off, the same is true for the pMOS transistor gate. From circuit simulation with the extracted TG netlist, the energy consumed during one turning on and off transient $E_{\text{TG,transient}}$ can be obtained

$$E_{\text{TG,transient}} = 640 \,\text{aJ}. \quad (31)$$

If the total number of turning on and off events $n_{\text{TG,events}}$ is determined, then the total amount of energy consumed can also be obtained. From observation of the circuit in Fig. 3, an expression for the number n_{TG,ϕ_0} of TGs connected to the signal ϕ_0 can be derived

$$n_{\text{TG},\phi_0} = 2 + 2 \cdot \left\lfloor \frac{1}{3} \cdot (n_{\text{w}} - 1) \right\rfloor. \quad (32)$$

The corresponding number of turn-on events $n_{\text{TG,ev},\phi_0}$ follows from Fig. 4

$$n_{\text{TG,ev},\phi_0} = \left\lfloor \frac{1}{3} \cdot (n_{\text{cyc}} + 2) \right\rfloor. \quad (33)$$

This procedure can be applied to all TGs in the circuitry so that all switching events are captured

$$n_{\text{TG,ev},\phi_0} = \left\lfloor \frac{n_{\text{cyc}} + 2}{3} \right\rfloor \cdot \left(2 + 2 \cdot \left\lfloor \frac{n_{\text{w}} - 1}{3} \right\rfloor \right) \quad (34)$$

$$n_{\text{TG,ev},\phi_1} = \left\lfloor \frac{n_{\text{cyc}} + 1}{3} \right\rfloor \cdot 2 \cdot \left\lfloor \frac{n_{\text{w}} + 1}{3} \right\rfloor \quad (35)$$

$$\vdots$$

The sum over the number of TGs multiplied by their respective number of switching events is finally multiplied by $E_{\text{TG,transient}}$, thus yielding $E_{\text{TG,total}}$, which is the total amount of energy consumed for switching the TGs in one analog multiplication unit. If added to the capacitive energy figures, the total amount of energy consumed during one IMCU multiplication procedure can be obtained

$$E_{\text{IMCU,total}} = E_{(1)} + E_{(2)} + E_{(3)} + E_{(4)} + E_{\text{TG,total}}. \quad (36)$$

6) Discussion: The significance of each contribution in the overall energy balance can be seen in Fig. 6 for an implementation using a quantization of 5 bits for both the weight and the input magnitude ($n_{\text{w}} = n_{\text{x}} = 5$). It is clear that the energy E_{TG} spent for switching the TGs dominates the overall consumption. The peak, denoted by $E_{\text{IMCU,max}}$, occurs at $|w| = \{1, 0, 1, 0, 1\}_2$ when each charge-sharing procedure leads to the maximum ΔV possible, and thus, the highest energy is consumed during precharging. Conversely, the minimum amount of energy E_{TG} is consumed for $w = 0$ when only the TGs are switched and no precharging occurs.

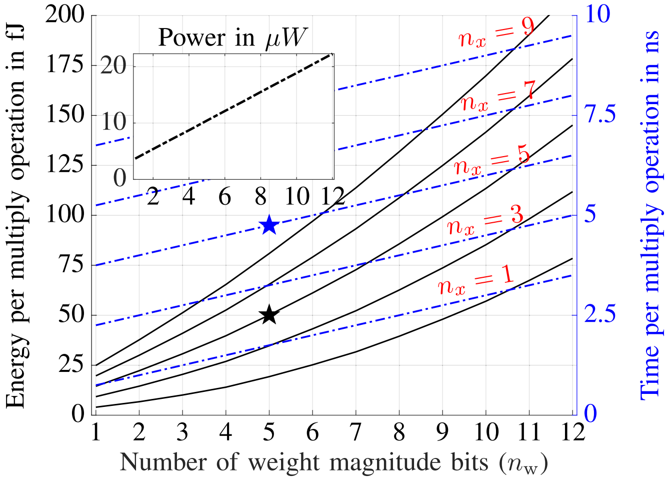


Fig. 7. Dependence of weight (n_{w}) and input (n_{x}) magnitude quantization on the energy, time, and average power consumed for one complete analog multiplication operation of a multibit IMCU. The sign bit is not included since it does not significantly impact the overall energy consumption. The red and blue stars indicate the design point selected for the implementation study.

Fig. 7 shows the impact of weight and input quantization on the peak energy, average power, and time per multiply operation. Since each additional weight bit implies another set of capacitors and switches, as well as one additional clock cycle, both latency and power consumption increase linearly. Thus, energy consumption, which is obtained as their product, shows a square dependence. Nonetheless, with respect to the input bits n_{x}, the scaling versus energy remains completely linear since the circuit only needs to operate for three additional cycles because of pipelining, without any additional hardware required within the IMCU.

F. Noise and Mismatch Impact

To ensure the highest possible accuracy of an analog IMC operation, all sources of nonlinearity and noise need to be known before an adequate circuit can be designed. Since the presented IMC circuit utilizes charge-sharing procedures between capacitors for the analog computation, the impact of TG ON-resistance $R_{\mathrm{TG,on}}$ mismatch remains negligible. This is because the circuit's cycle time is defined to always ensure complete voltage settling on a unit capacitor given a target $n_{\mathrm{acc,mac}}$-bit precision (from [21])

$$R_{\mathrm{TG,on}} \cdot C_{\mathrm{unit}} \cdot n_{\mathrm{acc,mac}} \cdot \ln 2 < T_{\mathrm{cycle}}. \tag{37}$$

The thermal $k_{\mathrm{b}}\mathrm{T}/\mathrm{C}$ noise impact is reduced by determining a minimum size for C_{unit} such that, once all capacitors along one column are shorted, the remaining noise amplitude is below the LSB of the employed ADC. Basically, a tradeoff between analog computation precision and latency, as well as energy efficiency, has to be made.

The main source of nonlinearity in this system is the mismatch between the C_{unit} capacitors due to manufacturing tolerances. As a consequence, charge-sharing procedures will not result in perfect averaging of the respective capacitor voltages. Given equally designed unit capacitors $C_0, C_1, \ldots, C_{n_{\mathrm{w}}}, C_{\mathrm{out}}$, the relative errors $\epsilon_{C_k} = C_k/C_{\mathrm{unit}}$ can be modeled as independent random variables, normally distributed with $\mathcal{N}(0, \sigma^2)$. These errors impact the D/A conversion process of the two multiplicands, weight w and input x, differently. Consequently, the weight is converted into the nonideal voltage $\widetilde{V}_{\mathrm{w}}$ by the pipeline DAC. In addition, the subsequent analog multiplier performs a nonideal scaling operation by a factor $\widetilde{\alpha}_{\mathrm{x}}$, which is proportional to an input value x_n. Specifically,

$$\widetilde{V}_{\mathrm{w}} = V_{\mathrm{pre}} \cdot \sum_{k=1}^{n_{\mathrm{w}}} b_k \cdot \frac{C_k}{C_k + C_{k-1}} \cdot \prod_{l=k}^{n_{\mathrm{w}}-1} \frac{C_l}{C_{l+1} + C_l} \tag{38}$$

$$\widetilde{\alpha}_{\mathrm{x}} = \frac{C_{n_{\mathrm{w}}}}{C_{n_{\mathrm{w}}} + C_{\mathrm{out}}} \cdot \sum_{p=1}^{n_{\mathrm{x}}} i_p \cdot \left(\frac{C_{\mathrm{out}}}{C_{n_{\mathrm{w}}} + C_{\mathrm{out}}}\right)^{n_{\mathrm{x}}-p}. \tag{39}$$

Using (38) and (39), the nonideal multiplication result $\widetilde{V}_{\mathrm{out}}$ is obtained as

$$\widetilde{V}_{\mathrm{out}}(w, x) = \widetilde{V}_{\mathrm{w}} \cdot \widetilde{\alpha}_{\mathrm{x}}. \tag{40}$$

Note that the pipeline DAC output is impacted by the mismatch of all capacitors except C_{out}, whereas the analog multiplier is impacted only by the relative mismatch between $C_{n_{\mathrm{w}}}$ and C_{out}. Since the IMCU can be described as a dual input DAC, the common metrics of integral nonlinearity (INL) and differential nonlinearity (DNL) can be used to obtain a quantitative measure of the analog multiplication accuracy. A change in the stored digital values of either weight or input alters the analog output voltage $\widetilde{V}_{\mathrm{out}}$. The impact of these variations can be measured by the weight- and input-related DNL metrics, i.e., DNL_w and DNL_x. Specifically,

$$\mathrm{DNL}_w(w, x) = \frac{\widetilde{V}_{\mathrm{out}}(w+1, x) - \widetilde{V}_{\mathrm{out}}(w, x)}{V_{\mathrm{LSB}} \cdot x} - 1 \tag{41}$$

$$\mathrm{DNL}_x(w, x) = \frac{\widetilde{V}_{\mathrm{out}}(w, x+1) - \widetilde{V}_{\mathrm{out}}(w, x)}{V_{\mathrm{LSB}} \cdot w} - 1. \tag{42}$$

Given a technology's capacitor fabrication tolerance σ and a maximum allowed DNL value for both weight and input $\mathrm{DNL}_{\mathrm{max}} = \max\{\max_w |\mathrm{DNL}_{w,x}|, \max_{w,x} |\mathrm{DNL}_x|\}$, a target yield $\mathrm{Y}_{\mathrm{DNL}}$ can be defined to determine the usable design space via Monte Carlo simulations, as shown in Fig. 8. From the confined areas shown in the plot, it becomes evident that an accurate BEOL fabrication process is required for high-precision analog computing. Advanced patterning techniques in deep submicron nodes allow to keep the mismatch below 0.05%, even for small C_{unit} sizes [24]. Furthermore, matching will gradually improve in upcoming technologies since the metal fabrication precision is increasing as the transistors are shrinking in accordance with Moore's law [25], [26].

Additional improvements of the analog computing precision could be achieved by adding calibration circuitry but only at a significant cost of area and complexity. Considering that both these factors are critical in determining the IMCU's competitiveness, calibration overhead will be avoided in the presented implementation, and only the precision provided "for free" by the underlying technology node will be used. Assuming a σ between 0.1% and 0.02%, the design point with five weight bits and five input bits will be chosen for implementation. Since only the magnitude is accounted for in

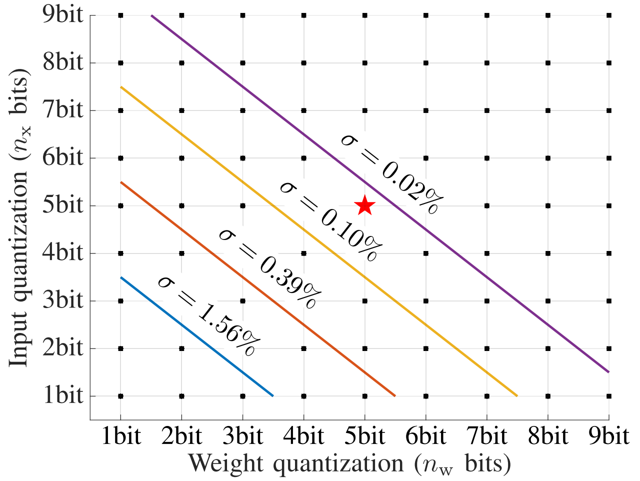


Fig. 8. IMCU design space obtained from 2000 Monte Carlo simulations considering mismatch between the unit capacitors C_{unit}. A target yield of $Y_{\text{DNL}} \geq 99\%$ for a maximum DNL smaller than 0.5 was assumed. Only the magnitude bits are considered. The sign bit is not included for either weight or input as it only determines the polarity of the precharge voltage and does not impact the DNL. The red star indicates the selected design point.

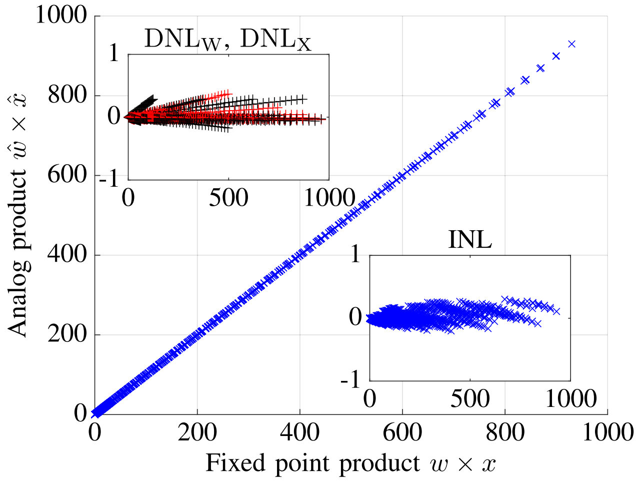


Fig. 9. Output characteristics of an IMCU using 5-bit unsigned weights and inputs. It is based on simulations using ideal switch models and nonideal unit capacitors C_{unit} with a relative matching variation of $\sigma = 0.1\,\%$. The respective DNL curves for the weight DNL$_w$ (red) and input DNL$_x$ (black) are shown in the top left plot. Both the DNL and INL remain bounded within ±1 LSB out of 10 bits.

the plot, this can be extended to a 6-bit signed weight and a 6-bit signed input. This is because the sign changes only the polarity of the precharge voltage V_{pre} and, thus, doubles the output range unimpaired by the capacitor mismatch. Instead, the precharge voltages $\pm V_{\text{pre}}$, which are assumed to be provided from externally, are required to be highly accurate with regards to symmetry around the common-mode V_{CM} to avoid inconsistent scaling of positive and negative multiplication results. In Fig. 9, the transfer characteristic of the analog multiplier is shown. Despite the presence of mismatch, the 10-bit multiplication result tracks the ideal output curve to a great extent.

In order to assess the performance of a full column of 128 IMCUs in the presence of other nonidealities, such as device mismatch and thermal noise, a transistor-level simulation is performed on a 14-nm implementation. The results are presented in Fig. 10 alongside the ideal outputs of a fixed-point digital implementation. It can be seen that the error waveform of the analog MAC output, plotted in the lower-right insert of Fig. 10, exhibits the typical S-shape characteristic, which arises from the v_{gs}-dependent TG ON-resistances, superimposed on the thermal noise error waveform. Furthermore, since this error is bounded between ±0.2 of an 8-bit LSB, it can be deduced that, when digitizing using an ADC with a minimum ENOB of 8, the obtained result will be very close to the truncated output of a full-precision fixed-point operation.

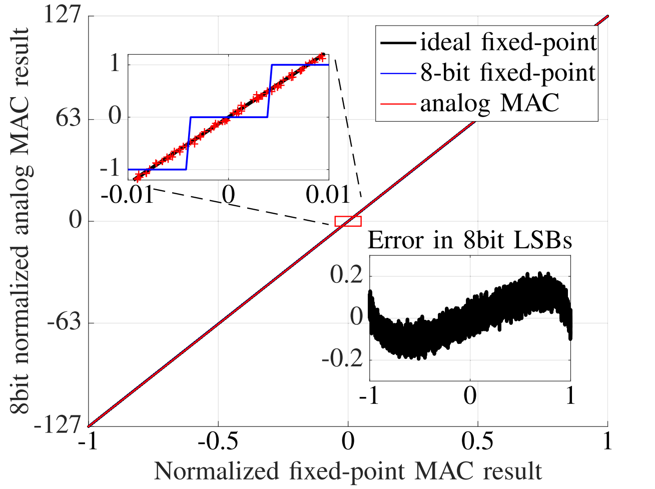


Fig. 10. MAC output characteristics of a transistor-level spectre simulation that involves a full column of 128 IMCUs implemented in 14 nm. Each IMCU multiplies a 6-bit signed input with a 6-bit signed weight. The effects of thermal noise, transistor, and capacitor mismatches were considered. The x-axis, representing the ideal fixed-point result, is normalized to $[-1, 1]$, and the y-axis, showing the analog MAC operation result, is normalized to the 8-bit ADC output range.

Deep neural network inference tasks, which are the designated applications for the presented IMC system, can tolerate this small reduction of precision of the MAC operation with usually no loss or in certain cases with insignificant loss in classification accuracy. The effects of ADC quantization, to which any reduced-precision implementation is subjected, are studied in detail in [27].

III. System Architecture

The goal of this work is to enhance standard SRAMs with IMC capabilities while maintaining as much as possible the original memory architecture. Conventional wordwise read and write procedures for instance are still needed to carry out fundamental memory I/O routines. In the system architecture, as depicted in Fig. 11, these basic elements are maintained with the same functionality. Their design and the mode of operation are described in [1]. The three building blocks that differentiate the novel SRAM architecture from the original will be described in the following sections.

A. IMC Subblocks

Support for performing multibit in-memory MAC operations is achieved by integrating the IMCU described in Fig. 3

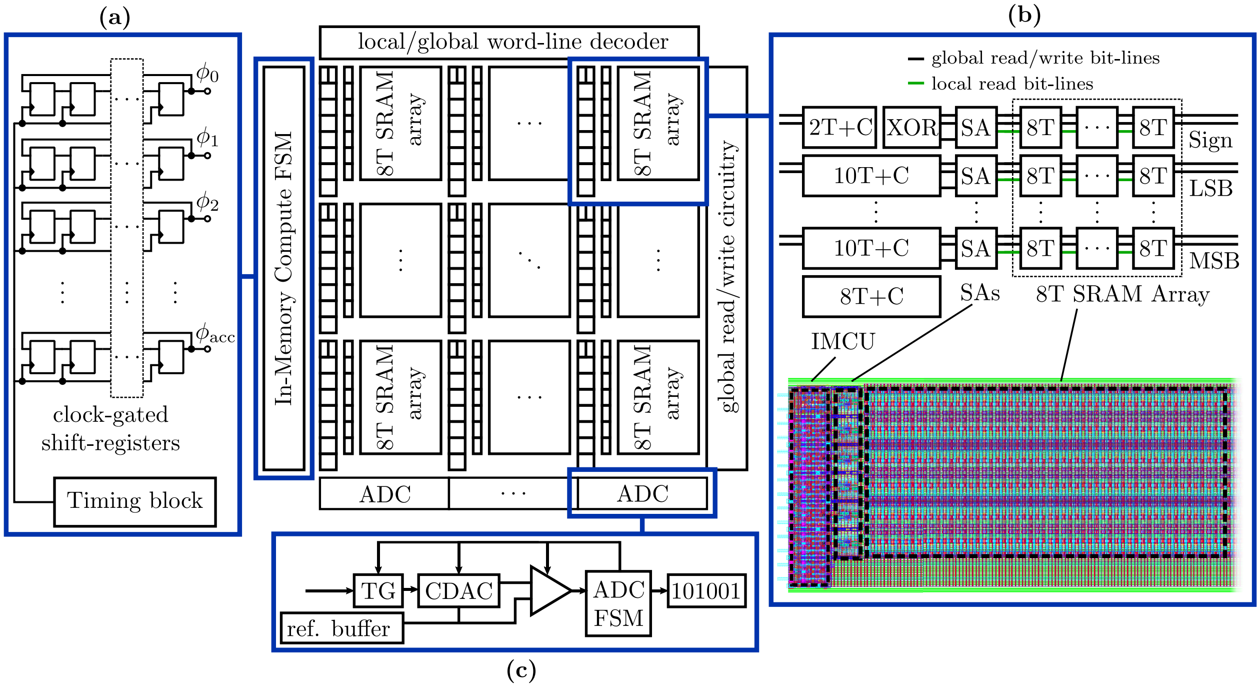


Fig. 11. System architecture of an 8T SRAM array with integrated in-memory compute units (IMCUs) to support matrix-vector multiplications. The classical SRAM architecture is complemented with (a) finite state machine to control the in-memory multiplication process, (b) IMC subblocks that support highly parallel local read operations to feed data to the IMCUs, and (c) array of ADCs for rapid digitization of the analog multiplication results. Standard read and write operations are carried out via the conventional global read–write circuitry.

into the SRAM array. If the bandwidth requirements for the IMC operation can be relaxed, then significant amounts of silicon area can be saved by time-multiplexing each analog multiplication circuitry between a set of n_{shared} words organized in a memory subarray. As a result, the execution of an in-memory MAC operation on the complete array will require an additional number of cycles, which is proportional to n_{shared}. Furthermore, this entails a change in the storage element. Instead of directly using the internal nodes of the 6T cell, the data now need to be read locally, preferably simultaneously in all IMCUs. To this end, the 8T SRAM cell is employed since it allows local read procedures in the memory subarrays via local bit-lines, as well as global read/write operations via the legacy periphery [28].

For performing the local read, additional sense-amplifiers (SAs) will be required, which can be much smaller in size than the peripheral amplifiers due to the lower number of cells that they cover. Finally, the outputs of the SAs are connected to the inputs of the multibit IMCU, as shown in Fig. 11(b). By taking into account the number of cycles $n_{\text{read,local}}$ required for the local read operation, the total latency for a full matrix-vector multiplication becomes

$$n_{\text{cyc,total}} = n_{\text{shared}} \cdot (n_{\text{read,local}} + n_{\text{cyc}}) \tag{43}$$

where n_{cyc} was defined in (18). Consequently, the local read procedures also enter the energy balance as an array-size-dependent term $E_{\text{read,local}}$.

B. IMC FSM

The in-memory MAC operation requires the IMCU circuitry and a well-defined sequence of pulses, similar to those shown in the example of Fig. 4. One way of generating these signals would be to adopt a timing block similar to those in classical SRAM architectures. In this article, a simpler solution using clock-gated shift registers is proposed, which is more flexible in the implementation and gives a more conservative and comprehensible estimation for energy consumption.

After receiving a positive edge signal, the FSM enables the clock signal of a large block of shift registers, as shown in Fig. 11(a), for a well-defined number of cycles. Some signals are common for the entire array, for instance, the three pulse signals: ϕ_0, ϕ_1, and ϕ_2. Others, such as the signal pair of $\phi_{\text{MSB,add}}$ and $\phi_{\text{MSB,rst}}$, are generated per each column depending on the input vector bits. In addition to the signals shown in Fig. 4, all signals required for performing the local read operation must also be provided by the FSM. Finally, all signals need to be buffered sufficiently in order to drive the respective inputs across the array.

C. ADC Design

After the analog MAC operation is completed, the final result, which corresponds to a voltage stored as charge across a column of output capacitors, needs digitization at sufficient precision. This necessitates the use of voltage input A/D converters. Given a large number of input signals, the ADCs, as shown in Fig. 11(c), should be designed to be as small, as fast, and as energy-efficient as possible. With this in mind, the SAR ADC design shown in [23] is used as a starting point.

TABLE I
SYSTEM PARAMETERS AND COMPONENT SIZES

Parameter	**Value**
Technology	14 nm
Voltage (VDD and V_{pre})	0.8 V
Operation frequency	4 GHz
Weight quantization	6 bit signed
Input quantization	6 bit signed
SRAM array size	6 × 128 × 2048
IMC sub-block count	128 × 64
SRAM sub-array size	6 × 1 × 32
C_{unit} capacitance	2 fF
Component	**Area** ($W \times H$)
8T custom SRAM cell	0.312 μm × 0.768 μm
in-memory compute unit (IMCU)	0.756 μm × 5.376 μm
XOR and 2T + C (IMCU)	0.756 μm × 0.768 μm
10T + C (IMCU)	0.588 μm × 0.768 μm
8T + C (IMCU)	0.588 μm × 0.768 μm
Local Sense-Amplifier(SA)	0.504 μm × 0.768 μm
Single IMC sub-block	11.24 μm × 5.376 μm
Full IMC sub-block array	719.62 μm × 688.13 μm
Local/Global Word-Line Decoder	719.62 μm × 44.090 μm
Global Read/Write Circuitry	28.860 μm × 688.13 μm
IMC FSM	21.504 μm × 688.13 μm
8-bit SAR ADC	12.380 μm × 60.180 μm
Complete system	769.980 μm × 792.398 μm

TABLE II
PERFORMANCE METRICS FOR IMC OPERATION

Operation	**Energy**	**Time**
Local SRAM read (in all IMCUs)[1]	196.61 pJ	2 ns
IMC FSM per cycle (signals for 1×local read+1×IMC)[1]	149.16 pJ	6.75 ns
Single IMCU operation[2]	50.1 fJ	4.75 ns
8-bit SAR ADC (per-conversion)[2]	3.3 pJ	1 ns
Full array IMC matrix-vector multiplication (32×local read and 32×IMC in all IMCUs)	30.96 nJ	216 ns

[1]Obtained from schematic simulations,[2]Obtained from post-layout simulations

Since the input consists of charge on a large capacitor, voltage buffers and complicated sampling circuits can be avoided, and the input can instead be transferred by means of charge-sharing to the capacitive DAC (CDAC) of the SAR ADC. The conversion procedure itself can be executed using a self-timed state machine to achieve high-speed conversion cycles of below 1 ns in 14 nm [29]. Moreover, the cost of each conversion is bounded at 3.3 pJ. By pitch-matching the ADC circuit to the width of one IMC subblock, the conversion latency impact on the overall bandwidth can be kept minimal.

IV. SYSTEM IMPLEMENTATION STUDY AND ANALYSIS

To demonstrate the benefit of the IMC-based architecture, a full system implementation study is performed, detailing the various components' area and power consumption. The 6-bit signed weights and 6-bit signed inputs are again assumed. The full memory has 128 × 2048 weights, arranged in 128 × 64 IMC subblocks each of 32 weights.

If standard design rules are employed rather than specialized SRAM push-rules, then the 8T SRAM cell size becomes 0.312 μm × 0.768 μm. Accordingly, the local SA circuit is designed with a matching height, using an area of 0.504 μm × 0.768 μm. Furthermore, this height is maintained in the various IMCU blocks so that, given the subcomponents' size reported in Table I, the total area of one IMCU can be determined to be 0.756 μm × 5.376 μm. Note that the unit capacitors are designed in the metals located above the transistors to keep the footprint as small as possible, similar to the approach taken in [17]. Finally, the size of one IMC subblock, consisting of IMCU, SAs, and SRAM array, is determined as 11.24 μm × 5.376 μm.

If the peripheral circuits, decoders, read/write circuitry, IMC FSMs, and ADCs are added, the area becomes 769.980 μm × 792.398 μm for the full system. Regarding area efficiency, 56.4% is used by the SRAM cells, and the IMC-related overhead amounts to about 35.4%. Figures for the energy spent in each operation are listed in Table II. Initially, before the actual IMC operation begins, the first column of SRAM words has to be locally read to make the corresponding values available to the attached IMCUs. Note that this operation can be completed rapidly in 2 ns because the local SAs cover a relatively small SRAM subarray with an accordingly small local bitline capacitance, and the obtained results are used locally and not transferred to the periphery. If executed in all 8192 subarrays, total energy of 196.61 pJ is consumed, according to simulations.

In the following step, after the local read, the IMCUs generate the voltages, which correspond to the MAC result and are eventually digitized by the ADCs. This process of alternating local-read and IMC operation is repeated 32 times, taking 216 ns, until the full matrix has been processed. Including the FSM and ADC energy, a total amount of 30.96 nJ is spent. These figures can be used to determine the full system throughput as 2.43 TOP/s at an efficiency of 16.94 TOP/s/W.

In relation to the throughput and energy efficiency figures, i.e., TOP/s and TOP/s/W, it has to be noted that the bit precision is not taken into account, thus putting the lowest precision implementations at an advantage. To adequately reflect the additional computational complexity tackled by multibit accelerators, the respective quantization of weight n_{w} and input n_{x} can be factored in, similar to the approach taken in [19], yielding precision scaled TOP/s and TOP/s/W.

This is shown in Table III, where recent implementations of analog in-memory MAC-operation accelerators using SRAM combined with capacitors [17], [18], [30], [31] are compared with the presented work.

For example, a scheme that could scale in terms of weight and input bits is demonstrated in [30]. In this article, a specific implementation for 4-bit inputs is presented. These multibit inputs are realized by expanding the 4-bit input value into a number of pulses, which, based on the different weight bit values along the rows, causes the capacitive read-bit-lines to discharge by a proportional amount. However, this input-to-time conversion creates an inherent exponential dependence of the latency on the number of input bits, leading to a limiting factor for finer input quantization. On the other hand, the multibit weights are realized in a single time step by employing a number of compensation and computation

TABLE III
COMPARISON TABLE OF SRAM- AND CAPACITOR-BASED ANALOG IN-MEMORY MAC-OPERATION ACCELERATORS

Metric	**This work**	**ISSCC'20 [30]**	**JSSC'19 [18]**	**JSSC'19 [17]**	**JSSC'20 [31]**
Technology	14 nm	7 nm	65 nm	65 nm	65 nm
Operating Voltage in V	0.8	1.0	1.0	0.94, 0.68, 1.2	1
Numbers obtained from:	Simulation	Measurement	Measurement	Measurement	Measurement
Input D/A conversion	Fully Digital FSM	PWM-based	Partially PWM-based	Not required (binary only)	Not required (binary only)
Weight D/A conversion	Fully Digital FSM	Charge-sharing-based	Not required (binary only)	Not required (binary only)	Not required (binary only)
Output A/D conversion	8bit SAR ADC	4bit Flash-ADC	7bit charge-sharing ADC	Batch-Norm. using 6bit DAC	5bit Flash-ADC
SRAM size	256 KB	4 KB	2 KB	295 KB	16 KB
SRAM bitcell	8T	8T	10T	10T1C	8T1C
SRAM words per IMCU (n_{shared})	32	1	16	1	1
Number of weight-bits ($n_{\mathrm{w}} + \text{sign}$)	6	4	1	1	1
Number of input-bits ($n_{\mathrm{x}} + \text{sign}$)	6	4	6	1	1
Number of output-bits	8	4	7	1	5
Relation between number of weight-bits n_{w}					
... and (cell) area	linear	exponential[1]	binary only	binary only	binary only
... and latency	linear	constant	binary only	binary only	binary only
... and power	linear	linear	binary only	binary only	binary only
Relation between number of input-bits n_{x}					
... and latency	linear	exponential[1]	exponential[1]	binary only	binary only
... and power	constant	constant	linear	binary only	binary only
Peak Throughput (TOP/s)	2.43 | 0.52*	0.372 | 0.04*	0.008	18.79	1.638
... with precision scaling	87.38 | 18.82*	5.958 | 0.64*	0.048	18.79	1.638
Energy Efficiency (TOP/s/W)	16.94 | 3.65*	351 | 37.8*	40.3	866	671.5
... with precision scaling	609.7 | 131.3*	5616 | 604.8*	241.8	866	671.5
Area Efficiency (TOP/s/mm^2)	3.98 | 0.86*	116.4 | 12.53*	0.092	1.5	20.22
... with precision scaling	143.2 | 30.84*	1862 | 200.5*	0.553	1.5	20.22

[1] scales with power of 2, * Normalized to 65 nm

capacitors. Since the total capacitance has an exponential dependence on the number of weight bits ($\sim 2^{n_{\mathrm{w}}}$), the chip area scales exponentially with the number of bits as well. Finally, note that the overall area overhead introduced for enabling the IMC capabilities remains manageable since, similar to [19], the standard SRAM-macrointernals remained unmodified, and exclusively, pitch-matched components are added to the periphery. In summary, the system described in [30] delivers high energy and area efficiency for the selected 4-bit input and weight quantization with the relatively low throughput being the only downside. This architecture might match well with the common requirements for IoT and edge applications. However, scaling the inputs and the weights beyond 4-bit incurs significant area and latency penalties. The same observations apply to the design described in [18], which demonstrates simultaneous vector processing, though for binary weights only. This architecture supports input precision scalability, albeit at an exponential cost in terms of latency. Furthermore, the high energy-efficiency number shown in [18] becomes less than that presented in this article, once the operands' precision is factored in.

Both the accelerator systems presented in [17] and [31] demonstrate high parallelism with completely binary implementations for inputs and weights. Note that, in binary cases, a multiply operation can be reduced to a single XNOR operation [32]. As a consequence of this simplification, no scalability in terms of precision can be achieved. In addition to restricting the operands' precision to binary only, [17] also applies binary batch-normalization on the analog MAC-operation result instead of digitizing using an ADC, thus contributing to the immensely high efficiency reported there.

In addition to the pure analog IMC approaches listed in Table III, a very interesting combination of a binary IMC approach with digital techniques to increase precision was presented in [33]. Specifically, through the use of digital shift and add circuitry, binary-only analog accelerators as, for example, in [31] and [17], can gain linear scalability in terms of weight and input quantizations. However, this incurs a cost in terms of latency and energy due to the multiple ADC cycles, as well as an increase in the area due to the peripheral digital adder circuits.

V. CONCLUSION

The cost, in terms of time and energy, associated with data movement has driven the concept of in-memory computing for neural network applications. According to this approach, the dominating matrix-vector operations are performed in-place, i.e., in the memory itself by exploiting certain physical properties of memory technologies. However, the main challenge of in-memory computing is the accuracy of the analog MAC operations.

In this article, we introduced a linearly scalable multibit in-memory computing system for accelerating MAC operations in standard SRAM. A novel interleaved switched-capacitor-based IMCU was proposed for conducting the analog computation, and its potential with regards to speed and energy efficiency was demonstrated. Although various SRAM-based matrix-vector multiplication engines have been proposed in the literature, our approach is the first to achieve computational precision that scales linearly in time, power, and area. Moreover, we have shown, via transistor-level spectre simulations, that, by using multibit representations for the input signals and the weights, there is no significant penalty in the accuracy of the SRAM-based analog MAC operations compared with a corresponding all-digital implementation with the same precision.

From a system design perspective, applications requiring a rather high quantization or precision (4–8 bits) will benefit substantially from the linear scalability of the presented IMC circuit and architecture, which can offer high throughput at an acceptable cost of area and energy. Finally, besides SRAM as the underlying memory technology, other volatile or nonvolatile memory technologies using simple SA-based read, such as DRAM, MRAM, or binary PCM, could also potentially be used in conjunction with our IMCU concept to provide multibit MAC computing capabilities.

Acknowledgment

The authors would like to thank A. Sebastian, M. Le-Gallo, and P. A. M. Bezerra for the constructive discussions.

References

[1] A. Pavlov and M. Sachdev, *CMOS SRAM Circuit Design and Parametric Test in Nano-Scaled Technologies Frontiers in Electronic Testing*, vol. 40, V. D. Agrawal, Ed. Dordrecht, The Netherlands: Springer, 2008.

[2] B. Keeth and R. J. Baker, *DRAM Circuit Design: A Tutorial*, 1st ed. Hoboken, NJ, USA: Wiley, 2000.

[3] H. Hidaka, *Embedded Flash Memory for Embedded Systems: Technology, Design for Sub-systems, and Innovations* (Integrated Circuits and Systems). Basel, Switzerland: Springer, 2017.

[4] R. Waser and M. Aono, "Nanoionics-based resistive switching memories," *Nature Mater.*, vol. 6, no. 11, pp. 833–840, Nov. 2007.

[5] M. Le Gallo and A. Sebastian, "An overview of phase-change memory device physics," *J. Phys. D, Appl. Phys.*, vol. 53, no. 21, Mar. 2020, Art. no. 213002.

[6] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnol.*, vol. 15, no. 7, pp. 529–544, Mar. 2020.

[7] S. R. Nandakumar *et al.*, "Mixed-precision deep learning based on computational memory," *Frontiers Neurosci.*, vol. 14, p. 406, 2020.

[8] M. Le Gallo *et al.*, "Mixed-precision in-memory computing," *Nature Electron.*, vol. 1, no. 4, pp. 246–253, Apr. 2018.

[9] E. Eleftheriou *et al.*, "Deep learning acceleration based on in-memory computing," *IBM J. Res. Develop.*, vol. 63, no. 6, pp. 7:1–7:16, Nov. 2019.

[10] A. Sebastian, M. Le Gallo, G. W. Burr, S. Kim, M. BrightSky, and E. Eleftheriou, "Tutorial: Brain-inspired computing using phase-change memory devices," *J. Appl. Phys.*, vol. 124, no. 11, Sep. 2018, Art. no. 111101.

[11] G. F. Close *et al.*, "Device, circuit and system-level analysis of noise in multi-bit phase-change memory," in *IEDM Tech. Dig.*, Dec. 2010, pp. 29.5.1–29.5.4.

[12] I. Giannopoulos *et al.*, "8-bit precision in-memory multiplication with projected phase-change memory," in *IEDM Tech. Dig.*, Dec. 2018, pp. 27.7.1–27.7.4.

[13] A. Chen, "A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics," *IEEE Trans. Electron Devices*, vol. 60, no. 4, pp. 1318–1326, Apr. 2013.

[14] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, Apr. 2017.

[15] A. Jaiswal, I. Chakraborty, A. Agrawal, and K. Roy, "8T SRAM cell as a multibit dot-product engine for beyond von Neumann computing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 11, pp. 2556–2567, Nov. 2019.

[16] X. Si *et al.*, "A twin-8T SRAM computation-in-memory unit-macro for multibit CNN-based AI edge processors," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 189–202, Jan. 2020.

[17] 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.

[18] A. Biswas and A. P. Chandrakasan, "CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, Jan. 2019.

[19] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *IEEE J. Solid-State Circuits*, vol. 53, no. 11, pp. 3163–3173, Nov. 2018.

[20] E. H. Lee and S. S. Wong, "Analysis and design of a passive switched-capacitor matrix multiplier for approximate computing," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 261–271, Jan. 2017.

[21] F.-J. Wang, G. C. Temes, and S. Law, "A quasi-passive CMOS pipeline D/A converter," *IEEE J. Solid-State Circuits*, vol. 24, no. 6, pp. 1752–1755, Dec. 1989.

[22] P. F. Ferguson, X. Haurie, and G. C. Temes, "A highly linear low-power 10 bit DAC for GSM," in *Proc. IEEE Custom Integr. Circuits Conf.*, May 2000, pp. 261–264.

[23] L. Kull *et al.*, "A 24-to-72gs/s 8b time-interleaved SAR ADC with 2.0-to-3.3pj/conversion and 30db SNDR at Nyquist in 14nm CMOS FinFET," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 358–360.

[24] D. Bustamante, D. Janke, E. Swindlehurst, and S.-H. W. Chiang, "High-precision, mixed-signal mismatch measurement of metal–oxide–metal capacitors," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 11, pp. 1272–1276, Nov. 2017.

[25] W. C. Jeong *et al.*, "True 7 nm platform technology featuring smallest FinFET and smallest SRAM cell by EUV, special constructs and 3rd generation single diffusion break," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2018, pp. 59–60.

[26] V. Tripathi and B. Murmann, "Mismatch characterization of small metal fringe capacitors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 8, pp. 2236–2242, Aug. 2014.

[27] A. S. Rekhi *et al.*, "Analog/mixed-signal hardware error modeling for deep learning inference," in *Proc. 56th Annu. Design Automat. Conf. (DAC)*. Las Vegas, NV, USA: ACM Press, 2019, pp. 1–6.

[28] L. Chang *et al.*, "Stable SRAM cell design for the 32 nm node and beyond," in *Dig. Tech. Papers. Symp. VLSI Technol.*, 2005, pp. 128–129.

[29] L. Kull *et al.*, "A 10 b 1.5 GS/s pipelined-SAR ADC with background second-stage common-mode regulation and offset calibration in 14 nm CMOS FinFET," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 474–475.

[30] Q. Dong *et al.*, "A 351TOPS/W and 372.4GOPS compute-in-memory SRAM macro in 7 nm FinFET CMOS for machine-learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 242–244.

[31] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, "C3SRAM: An in-memory-computing SRAM macro based on robust capacitive coupling computing mechanism," *IEEE J. Solid-State Circuits*, vol. 55, no. 7, pp. 1888–1897, Jul. 2020.

[32] X. Si *et al.*, "A dual-split 6T SRAM-based computing-in-memory unit-macro with fully parallel product-sum operation for binarized DNN edge processors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 11, pp. 4172–4185, Nov. 2019.

[33] 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.