Mixed signal multiply and adder parallel circuit for deep learning convolution operations

José A. Díaz-Madrid[1], Ginés Doménech-Asensi[2], Ramón Ruiz-Merino[2]. Juan Zapata-Pérez[2], José J. Martínez-Álvarez[2]

[1]Dpto. de Ingeniería y Técnicas Aplicadas, Centro Universitario de la Defensa, Santiago de la Ribera, Spain, jose.diaz@cud.upct.es

[2]Dpto. de Electrónica, Tec. de Computadoras y Proyectos, Universidad Politécnica de Cartagena, Cartagena, Spain, gines.domenech@upct.es

***Abstract*—This work presents a new analog architecture to perform image convolution for deep learning purposes in CMOS imagers in the analog domain. The architecture is focused to reduce both power dissipation and data transfer between memory and the analog operators. It uses mixed signal multiply and add operators arranged following a row-parallel architecture in order to be fully scalable for different CMOS imager sizes. The multiplier circuit used is based on a current mode architecture to multiply the value of analog inputs by the digital stored weights and produce current mode outputs which are then added to obtain the convolution result. A digital control circuit manages the pixel readout and the multiply and add operations. The architecture is demonstrated performing 3x3 convolutions on 64x64 images with a padding equal to 1. Convolution weights are locally stored as 4-bit digital values. The circuit has been synthesized in 110 nm CMOS technology. For this configuration, the simulation results show that the circuit is able to perform a whole convolution in 32 us and achieve an efficiency of 2.13 TOPS/W. These results can be extrapolated to larger CMOS imagers and different mask sizes.**

Keywords—Multiplier, CMOS, computer vision, deep neural network

I. INTRODUCTION

In recent years, deep neural networks (DNNs) have become a paradigm in computer applications, being the focus for more and more researches to obtain faster and more accurate networks. On the other hand, hardware implementation of such systems is such a difficult task, given the effort needed to simplify computer algorithms in order to fit them into electronic circuits [1]. Multiply and accumulate (MAC) operations are widely used in convolutional neural networks, a particular type of network used for identification and tracking purposes. In a typical implementation, convolutional layers are responsible for more than 90% of execution time during the inference. These operations require large power dissipation and also large movements of data, and so, memory bandwidth often becomes the bottleneck of the network, given the large number of weights that must be transferred from the memory to the MAC operators.

Compared to digital processing, analog signal processing circuits are able to achieve much higher energy efficiency. However, its main drawback is an increased sensitivity due to device and circuit non-idealities. Although these types of circuits are not as precise as digital circuits, deep learning characteristics allow the use of circuits with lower accuracy specifications, since the loss of precision can be accounted for during the training of the DNNs [1]. Moreover, analog-domain operations can operate directly on raw analog data obtained from the image sensor before digitization, thus alleviating the requirements of the analog to digital conversion circuitry. Over the past few years, so called analog computing has been applied for the design of deep learning circuits in different works [2].

There are different approaches to designing analog MACs, usually depending on resolution, speed or power dissipation specifications. Switched capacitor circuits demand significantly less energy than its digital counterparts. However, the overhead area of capacitors represents a serious drawback [3] in these types of circuit. There have also been proposals of MAC operations in the time domain [4], although these require a large number of digital-to-time converters arranged in series which results in a significant silicon area. On the other hand, continuous time operators such us current mode circuits allow for analog computation with large dynamic ranges and low supply voltages [5].

In this paper, a mixed signal multiply-add architecture to perform convolution on images is presented. The circuit is designed to be synthetized in the analog domain close to the image pixels in order to reduce power dissipation and data transfer between memory and the analog operators and it presents a row-parallel row architecture to be fully scalable for different CMOS imager sizes. In this paper a 64x64 CMOS imager and a 3x3 convolution mask have been synthesized at device level to evaluate the circuit performance. The circuit combines analog current mode processing with digital storage of weights. The rest of the paper is organized as follows. Section II describes the structure and operation of the proposed architecture. Results are shown in Section III. Finally, conclusions are summarized in Section IV.

II. CIRCUIT DESCRIPTION

Fig. 1 shows a general overview of the implementation of the convolution circuitry for a generic size CMOS imager, where pixels are arranged following a row pattern. Each active pixel sensor (APS) in a row is connected to a multiply and add (MAD) operator through one of the switches sw_i, which performs partial MAD operations on each row, and these results are then added to obtain the convolution. The imager is framed by a border of dummy pixels whose function is to provide the capability of performing convolution with a padding factor of 1.

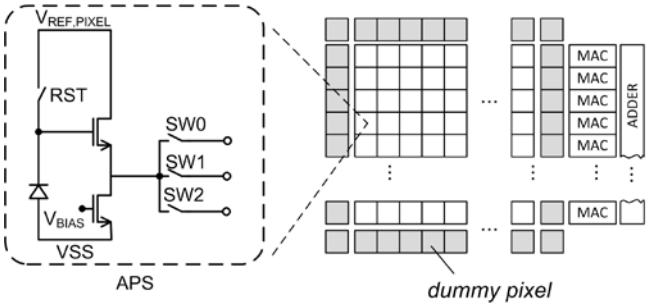

Fig. 1. General overview of the row parallel convolution architecture.

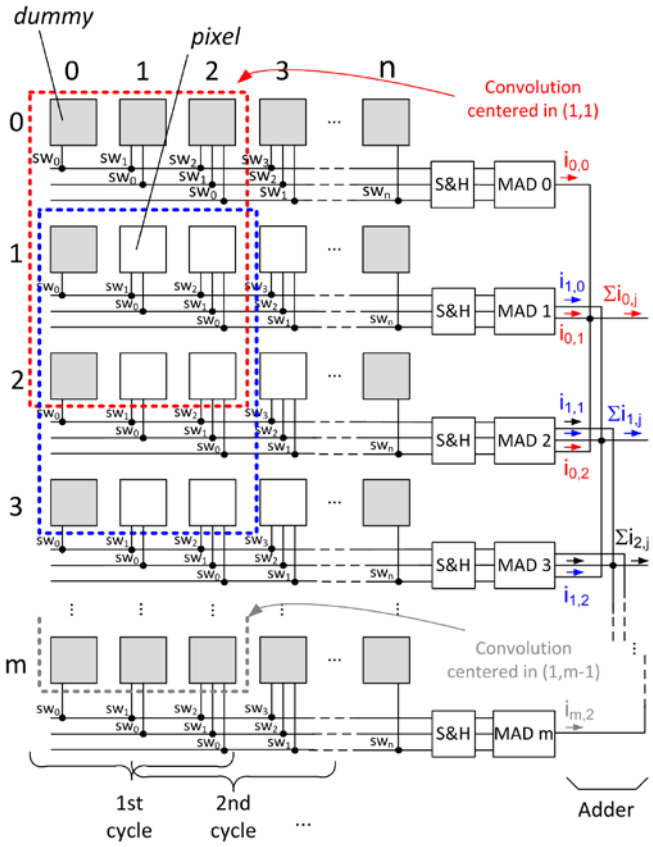

Fig. 2. Struture of the convolution circuitry. a) pixel and MAC arrangement b) c) d) mixed signal current mode MACs, e) 3x3 mask.

The circuit uses a mixed mode operation combining voltage and current modes as well as digital values for the 4-bit weights. A detailed implementation of this architecture is shown in Fig. 2. The figure represents an *m* rows *x n* columns imager, which includes two additional columns and two additional rows corresponding to the dummy pixels border. The architecture is organized following a row-parallel arrangement, where each pixel is connected to its respective row-MAD through a set of analog switches sw_i which drive the pixel output to an intermediate Sample & Hold (S&H) circuit. The number of analog switches is the same as the side size of the convolution mask. In the figure, a 3x3 convolution mask is being applied, and therefore this is the number of switches per pixel. However, pixels in columns 0, 1, *n-1* and *n* use one or two switches respectively, since they take part in the convolution only once or twice. Each convolution is decomposed in a number of parallel MAD operations carried out on the pixels of the same row. So, for a 3x3 convolution, the three sets of pixels in each one of the three first rows are processed simultaneously (Fig. 2).

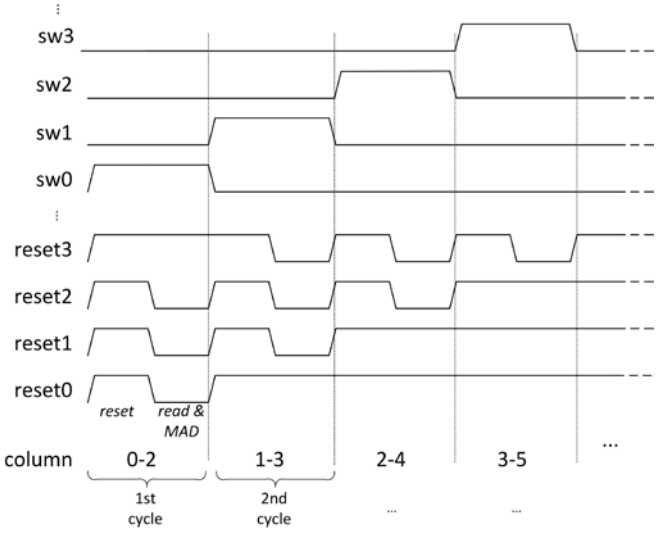

Fig. 3. Timing of the convolution operations

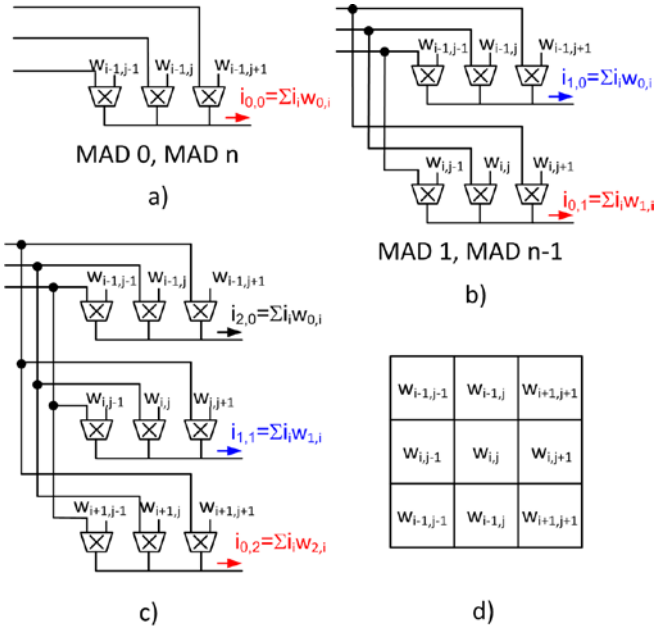

Fig. 4. Row-parallel MAD arrangement a) rows 0 and 65 b) rows 1 and 64 c) rest of rows, d) 3x3 convolution mask.

In fact, the architecture is designed to process the three neighboring columns involved in a convolution of a single clock period. The operation for this 3x3 convolution is the following: in each period, pixels are reset during the first semi-period, while the pixel readout and the MAD operation are performed during the second semi-period, as is shown in Fig. 3. The processing of an image starts enabling columns 0 to 2 through switches sw_0 to sw_2 and resetting the involved pixel values along all the three columns. The analog convolution for these columns is performed during the second semi-period in a parallel fashion for all the rows of pixels involved in the operation, i.e. convolutions centered in column 1 are simultaneously obtained during the first cycle. So, for the convolution centered in pixel (1,1), MAD 0 performs the operation $i_{0,1} = \Sigma i_j \cdot w_{0,j}$ for $j \in (0,2)$, while the values of $i_{0,2}$ and $i_{0,3}$ are obtained simultaneously in MAD 1 and MAD 2 respectively. The result of these three operations is then added to yield the convolved pixel output value.

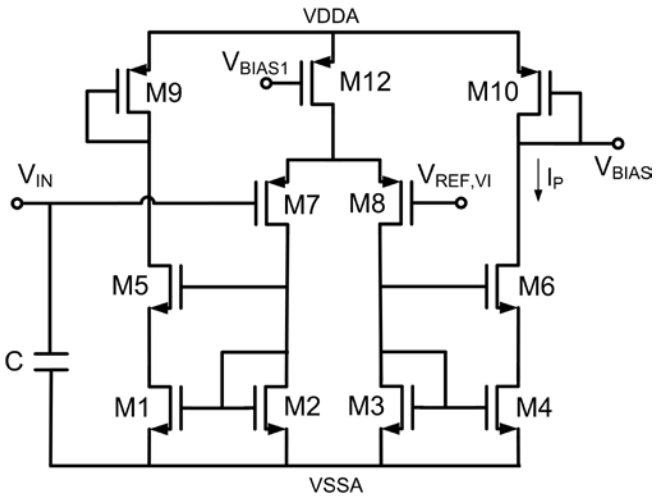

Fig. 5. S&H and interface circuit.

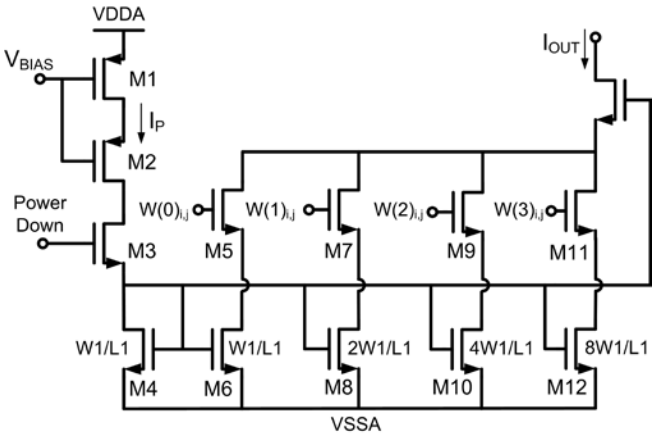

Fig. 6. 4-bit analog/digital multiplier.

The structure of the MAD circuits is detailed in Fig.4. Fig 4.a details the MAD used for rows 0 and m. It only outputs one value of current given that the pixels involved only take part in the convolution centered in row 1. Fig 4.b corresponds to MAD 1 and MAD m-1. The two outputs are required since pixels in row 1 take part in the convolution centered in column 0 and in column 1. Finally, the circuit described in Fig. 4 c corresponds to the rest of the MADs, where pixels take part in three simultaneous convolutions. This process is repeated for all columns in the imager to obtain the convolution of the whole image. Fig. 5 shows the S&H-VI circuit used to sample the pixel value and interface with the current mode analog multiplier. The S&H circuit is composed of the capacitor and the switch sw_i placed at the focal plane shown in Fig.2. The value held at the capacitor drives the differential cascaded structure to provide a current I_P representing the pixel value. In order to achieve the proposed parallelization schema (Fig. 4) without increasing the number of current mirrors, the analog input to the multiplier is the voltage at the output of the S&H-VI module (V_{BIAS}). Fig. 6. shows the architecture of the analog multiplier with digital weights. The multiplier is designed to work with 4-bit weights, which are stored close to each multiplier cell, allowing a reduction of the data transfer compared to the use of a same memory for all the weights.

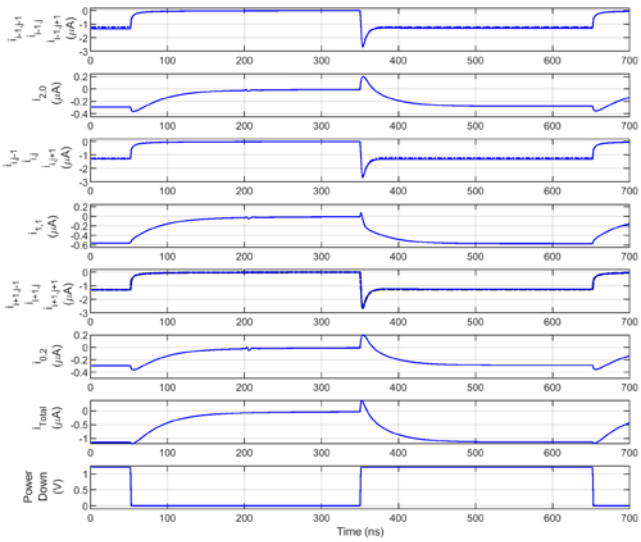

Fig. 7. Evolution of analog signals in the multiplier circuit.

III. RESULTS

The proposed architecture has been synthesized at device level using a CMOS 110 nm CIS technology. Fig. 7 shows the evolution of the multiplier output current during a full operation cycle. During the first semi-period the pixel is operating and the multiply circuit is powered down. During the second semi-period, the multiply circuit is powered on and the output signal is almost stabilized after 250 ns. Thus, the full pixel readout and processing could be done in 500 ns (2 MHz) per group of columns, although additional accuracy can be achieved using larger periods. This means that the processing of the 64 columns of a whole image plus the dummy borders for padding equal to 1 and a frequency of 2 MHz requires 32 μs. These numbers allow the processing of high density images in real-time applications. Due to its row-parallel arrangement this schema is fully scalable to larger image sizes because the silicon height of the MAD and adder circuitry (Fig. 2) is independent of the image size. However, for larger convolution masks, additional signal buses would be required since each pixel would have additional outputs, and the estimated silicon area for both the MAD and the adder would be wider. In order to decrease the power dissipation, the MAC circuitry is powered down during the first semi-period of each cycle. Fig. 7 shows the output currents involved in a convolution operation of a single pixel applying a mask of 3x3 for an operating frequency of 1.66 MHz. The plots have been labeled according to Fig. 4.c. During this semi-cycle, the currents decrease in value to almost 0 reducing the power consumption of this module. During the second semi-period, the multiply circuit is powered on and the currents of each module are multiplied by its corresponding weight. Therefore, a current proportional to the binary weight of the digit is generated. In this topology, the output currents of the nine modules are added simultaneously. i_{Total} shows the total output current of a convolution operation while $i_{2,0}$, $i_{1,1}$, $i_{0,2}$ are the output current of the addition of the top, middle and bottom rows. Finally, the power down signal is shown at the bottom of Fig. 7.

TABLE I. PERFORMANCE OF THE CONVOLUTION ARCHITECTURE

	This work	**[5]**	**[6]**	**[7]**	**[8]**
Technology	110 nm	130 nm	130 nm	40 nm	180 nm
Resolution	Analog / 4b	Analog	Analog / 4b	Analog / 3b/ 4b	Analog/ 7b
Frequency	2 MHz	8.3 kHz	20 kHz	1 GHz	10 MHz
Supply (V)	1.2	3	1.2	1	2
Power	599 μW	11.4 μW	663 nW	228 μW	- [a]
Efficency (TOPS/W)	2.13	1	0.0603	8,77	0.545

[a] The whole sensor image consumes 1.8mW, but not value is reported for the convolution operator.

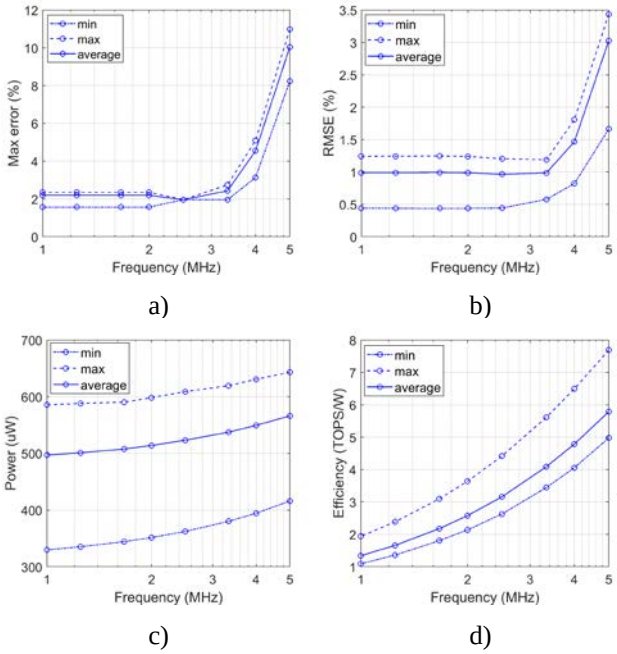

Fig. 8. Error power and efficiency versus frequency

The performance achieved by the proposed convolution architecture is shown in Table I and compared to other works which implement front-end neural networks. Compared to other implementations in a similar technology, the simulated results offer better efficiency in terms of TOPS/W, for similar or smaller supply voltages. It also offers a comparable efficiency to [7], taking into account the technology scaling factor. Moreover, the proposed work is fully scalable for different imager sizes. Because the power dissipation scales with the imager size, in the same proportion as the number of operations, the efficiency remains constant. The efficiency of the circuit is related to the operation frequency. Fig. 8 shows the relationship between different circuit performance and the clock frequency for a set of five images taken from the USC-SIPI image database [9], and cropped to 64 x64 pixels size. In each plot, the minimum, the maximum and the average values of each performance parameter is displayed. As expected, the average power dissipation increases linearly with the frequency (Fig. 8.c). It can be noted, however, the large variation of the power dissipated depending on the characteristics of the processed image. The efficiency is also increased with frequency achieving a maximum average of 5.79 TOPS/W for 5 MHz (Fig. 8.d), given that the rate of operations per unit of time rises faster with frequency than the power dissipation.

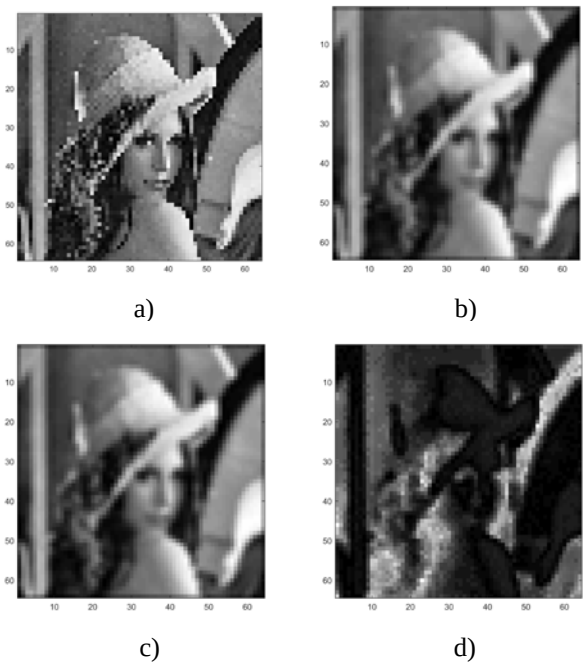

Fig. 9. Example of analog filtering a) original image, b) image filtered using the proposed architecture c) ideal filter d) normalized difference.

However, both the absolute maximum error and the RMSE show large values for this operation frequency (Figs. 8.a and 8.b). Stable values of error are achieved for frequencies under 2.5 MHz. For that frequency the achieved efficiency reaches a

value of at least 2.13 TOPS/W. At this frequency an VGA image would be convoluted in 256µs, two orders of magnitude faster that real-time requirements.

Fig. 9.a. shows an example of analog filtering using the architecture described in this work. Figures 9.b and 9.c show, respectively, the image processed using an ideal filter with MATLAB and the image filtered using the proposed architecture. This result has been obtained at device level simulation. The difference between the real analog filter and an ideal filter is normalized to the extreme value and shown in Fig. 4.d.

IV. Conclusions

This work has presented a new parallel mixed signal architecture capable of performing fast image convolutions in the analog domain in CMOS imagers. Due to its row-parallel arrangement, the architecture has been designed to be fully scalable for different image sizes, and is capable of processing very high resolution imagers at frame rates beyond real time. Moreover, it operates at a moderate clock frequency, avoiding the use of a costly precise clock circuitry. Although in this work a 3x3 convolution mask has been used, it can be resized to other sizes. This would require an increase in the number of multiply-add circuits to fit the size of the mask, which can still be easily placed close to the image sensor by widening the layout in each row for the same height. In the same way, the padding factor can be expanded using additional rows and columns of dummy pixels around the pixels frame. Other convolution parameters such as the stride can also be synthesized by modifying the control circuitry. Although in this work this control circuitry is fixed, it could easy be adjusted to offer such type of programmability.

V. Acknowledgements

This work has been partially funded by Spanish government through project RTI2018-097088-B-C33 (MINECO/FEDER, UE).

References

[1] V. Sze, "Designing Hardware for Machine Learning: The Important Role Played by Circuit Designers," in *IEEE Solid-State Circuits Magazine*, vol. 9, no. 4, pp. 46-54, Fall 2017. doi: 10.1109/MSSC.2017.2745798

[2] D. Bankman and B. Murmann, “An 8-bit, 16 input, 3.2 pJ/op switchedcapacitor dot product circuit in 28-nm FDSOI CMOS,” in Proc. IEEE Asian Conf. Solid-State Circuits, Nov. 2016, pp. 21–24.

[3] Y. Toyama, K. Yoshioka, K. Ban, S. Maya, A. Sai and K. Onizuka, "An 8 Bit 12.4 TOPS/W Phase-Domain MAC Circuit for Energy-Constrained Deep Learning Accelerators," in *IEEE Journal of Solid-State Circuits*, vol. 54, no. 10, pp. 2730-2742, Oct. 2019.

[4] D. Miyashita et al., “An LDPC decoder with time-domain analog and digital mixed-signal processing,” IEEE J. Solid-State Circuits, vol. 49, no. 1, pp. 73–83, Jan. 2014.

[5] J. Lu, S. Young, I. Arel and J. Holleman, "A 1 TOPS/W Analog Deep Machine-Learning Engine With Floating-Gate Storage in 0.13 µm CMOS," in *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 270-281, Jan. 2015. doi: 10.1109/JSSC.2014.2356197

[6] J. Zhang, Z. Wang and N. Verma, "18.4 A matrix-multiplying ADC implementing a machine-learning classifier directly with data conversion," *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, San Francisco, CA, 2015, pp. 1-3.doi: 10.1109/ISSCC.2015.7063061

[7] E. H. Lee and S. S. Wong, "Analysis and Design of a Passive Switched-Capacitor Matrix Multiplier for Approximate Computing," in IEEE Journal of Solid-State Circuits, vol. 52, no. 1, pp. 261-271, Jan. 2017.doi: 10.1109/JSSC.2016.2599536.

[8] Z. Chen *et al*., "Processing Near Sensor Architecture in Mixed-Signal Domain With CMOS Image Sensor of Convolutional-Kernel-Readout Method," in *IEEE Transactions on Circuits and Systems I: Regular Papers*. Prepient 2019. doi: 10.1109/TCSI.2019.2937227

[9] [Online]. USE-SIPI Image Database. Available: http://sipi.usc.edu/database/database.php

.