Redundancy and analogue slicing for precise in-memory machine learning - Part I: Programming techniques

Giacomo Pedretti, *Member, IEEE,* Piergiulio Mannocci, *Student Member, IEEE,* Can Li, Zhong Sun, John Paul Strachan, and Daniele Ielmini, *Fellow, IEEE*

***Abstract*—In-memory computing (IMC) is receiving considerable interest for accelerating artificial intelligence (AI) tasks, such as neural network training and inference. However, IMC can also accelerate other machine learning (ML) and scientific computing problems, such as recommendation systems, regression and PageRank, which are ubiquitous in datacenters. These applications typically have higher precision requirements than neural networks, which can challenge analogue-based IMC and sacrifice some of the expected energy efficiency benefits. Here we address these challenges experimentally, presenting new techniques improving the accuracy of the solution of linear algebra problems, such as eigenvector extraction for PageRank, in a fully integrated circuit (IC) with analogue resistive switching memory (RRAM) devices. Our custom redundancy algorithm can improve the programming accuracy by using multiple memory devices for representing a single matrix entry. Accuracy is further improved by error compensation with analogue slicing, which allows an ever more precise value representation.**

***Index Terms*—In-memory computing, memristor, RRAM, pagerank, memory reliability, neural network, artificial intelligence.**

I. Introduction

ARTIFICIAL intelligence (AI) has an energy consumption problem [1]. It has in fact been reported that training a Transformer model emits the same amount of CO_2 of 5 cars in their lifetime. This carbon footprint is due to the electricity used to power up and cool down large scale computing systems, where training of AI models and scientific computing takes place. Such computing systems are usually based on the conventional von-Neumann architecture, where memory and processing units are physically separated, and most of energy (and time) is spent for transferring information from the memory to the processor and back [2]. This architecture is not optimized to perform matrix-vector multiplication (MVM) which is the backbone of most AI and scientific computing algorithms. Custom digital architecture, such as tensor processing units (TPU) [3], have been developed to carry out efficiently this kind of computations. A recent approach, named in-memory computing (IMC) [4], aims to perform computation directly within the memory, without the need for data movement. Among the device technologies explored for IMC, emerging memory devices such as resistive random access memory (RRAM), often dubbed memristor, are particularly attractive due to the low energy consumption, high speed operation [4], high densities in compact crosspoint arrays [5], and the opportunity to encode analogue states [6]. The last two features enable the hardware acceleration of MVM exploiting Ohm's law and Kirchoff's law. Crosspoint arrays have been used to accelerate various MVM-based computing tasks, such as neural networks [7], [8], image processing [9], optimization problems [10], [11] and the solution of linear systems of equations [12], [13]. The latter can be either carried out iteratively by performing numerical methods, such as Jacobi [12] or Krylov techniques [13], or can be accelerated in one step with analogue circuits [14], [15]. Also, scientific computing usually requires high precision [13], thus memory devices capable of precise analogue programming are needed. While integrated circuits (IC) comprising CMOS driving circuitry and memristive devices for computing have already been developed [16]–[18], the feasibility of high-precision mapping for RRAM-based IMC has not been reported yet.

Here, we demonstrate an eigenvector calculation by fully integrated IMC system [18], particularly focusing on the precision requirements for the Pagerank application [19], [20]. While various techniques have been presented to mitigate the variation of memristor device conductance [6], [21], a comprehensive study of the noise contribution in a fully integrated IMC system has yet to be performed. After showing the performance and impact of variation, we propose two programming techniques based on redundancy and analogue slicing to improve the accuracy of analogue memory states in RRAM devices. Programming errors are experimentally evaluated and demonstrated as negligible compared to background noise when redundancy/slicing techniques are applied. Application-specific improvements thanks to this technique and benchmark are presented in the companion paper [22].

II. Hybrid CMOS-memristor Integrated circuit

Figure 1 shows a crosspoint array of memristive devices for IMC. The crosspoint is made by crossing top electrode (TE) lines with bottom electrode lines (BE), resulting in a

G. Pedretti is with Politecnico di Milano, Milano, Italy and Hewlett Packard Labs, Milpitas, CA, USA.

P. Mannocci and D. Ielmini are with Politecnico di Milano, Milano, Italy

C. Li is with the University of Hong Kong (HKU), Hong Kong SAR, China and Hewlett Packard Labs, Milipitas, CA, USA

Z. Sun is with Peking University (PKU), Beijing, China and Politecnico di Milano, Milano, Italy

J. P. Strachan is with Hewlett Packard Labs, Milpitas (CA), USA

Correspondance should be addressed to giacomo.pedretti@polimi.it, john-paul.strachan@hpe.com and daniele.ielmini@polimi.it.

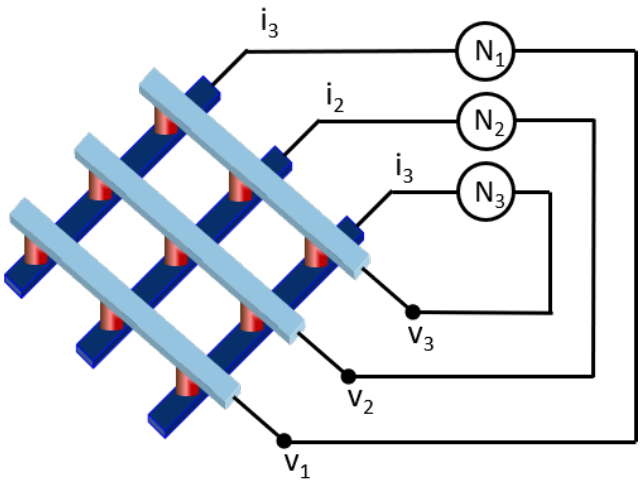

Fig. 1. Crosspoint array for performing accelerated MVM. TE lines (light blue) are polarized to a voltage v, while BE lines (dark blue) are tied to ground. The current flowing is equal to $i = Av$ where A is a conductance matrix programmed in the memristive devices (red), which are created at each intersection between TE and BE.

compact array structure where the memristive device is placed at each intersection [4]. Interestingly, this structure can be used not only as a memory, but also as a computational unit to accelerate MVM. By programming the matrix A in the conductance of the memristors in the crosspoint array and applying a vector v to the columns, the resulting current vector i on the grounded rows is equal to the matrix-vector product of A and v, namely $i = Av$ [4]. The IMC circuit was implemented in a hybrid analogue-digital IC as shown in Fig. 2(a) [18], comprising routing and sensing peripheral circuits integrated in 180nm CMOS node whose layout is shown in Fig. 2(b), with monolithically integrated TaO_x memristive devices [23] (Fig. 2(c)) with large conductance windows up to $10^6\times$ [23]. Each chip includes three crosspoint arrays of 4 kb (64×64) analogue memristors in a 1-transistor-1-resistor (1T1R) structure and integrated analogue sensing/driving circuitry. It is possible to access any memristor either in random fashion for programming and reading, or in a parallel way, performing MVM in-situ on a whole array, which can be executed in parallel with the other two 64×64 arrays and in any additional chips. TE lines are connected to a simplified digital to analogue converter (DAC) namely a switch from ground to an arbitrary analogue voltage, typically V_{set}/V_{reset} or V_{read}, while for performing MVM a shift and add technique is used [24] where input signals are applied as a digital stream spanning from 0 to V_{read} and the output is then reconstructed as digital summation in the digital domain. The mixed analogue-digital approach allows to program a given input precision on the fly, as well as avoid problems due to conductance non-linearity which can be significant in high resistance modes [6] thanks to the use of a single input voltage equal to V_{read}. Currents flowing into the BE are converted to voltage with a trans-impedance amplifier (TIA) with programmable gain and then read with an analogue to digital converter (ADC). In particular, columns are organized in blocks of 4 which are multiplexed to a single TIA and the 16 TIA outputs are further multiplexed and read with a single 10 bit ADC. The IC was packaged on a printed circuit board (PCB) supplying power, analogue references, and external communication via a microcontroller. To show the analogue programming performance of the IC, we programmed the cover art of the studio album of Abbey Road by The Beatles [25], in the three 64×64 memristor arrays by writing each red/green/blue (RGB) color component into a different array. To this purpose, we tuned the conductance with a program and verify (PV) algorithm, which iteratively performs set/reset operations on the whole array by modulating TE and BE voltages until reaching convergence.

Fig. 3(a) shows the final result while Fig. 3(b) shows the standard deviation of G after reading the conductance values 100 times indicating relatively small fluctuations ($< 5\mu S$ or $< 5\%$) thus supporting the good performance of our system.

III. In-memory Pagerank

An interesting and ubiquitous algebraic operation based on MVM is the calculation of matrix eigenvectors. The principal eigenvector, i.e. the eigenvector corresponding to the maximum eigenvalue, can be obtained by power iteration where, given a matrix A, the eigenvector x is obtained iteratively by solving $x_{k+1} = Ax_k/||Ax_k||$ where $||\cdot||$ is any suitable norm. The IMC circuit can be used naturally to accelerate such a problem by mapping matrix A into the conductance of the memristor array, applying x_k to the TE lines and reading the currents at each iteration. As an example, a random problem was mapped in a 8×8 section of one of the memristor arrays as shown in Fig. 4(a). The output voltage was monitored and the mean absolute error (MAE) was computed as

$$MAE = \frac{1}{n}\sum_{k=1}^{N}|x(k) - x'(k)| \quad (1)$$

where $N = 8$ is the problem size, x is the analytical solution and x' the measured solution. Fig. 4(b) shows the result where it is possible to see that the error correctly approaches 1% in just a few cycles, while the analytical and measured output voltages (inset) are in good agreement.

Although IMC can accelerate algebraic tasks, most scientific computing problems require a relatively high precision (e.g. floating point) both in the matrix elements to be stored and in the output solution. For this reason, practical implementation of IMC for scientific problems needs a mixed-precision architecture [13]. Nonetheless, improving the low precision operations can speed up the overall execution significantly. On the other hand, machine learning and data-analysis algorithms generally do not require floating point precision, only that the correct ordering or probability are given in the result. This is the case for Pagerank, the algorithm behind web search engines and recommender systems [19], [20]. The problem consists of ranking the authority of various web-pages on a graph, which can be done with different techniques such as randomly walking the graph and computing the probability of landing in a given page or by computing the principal eigenvector of a proper matrix. Fig. 5(a) shows a graph of

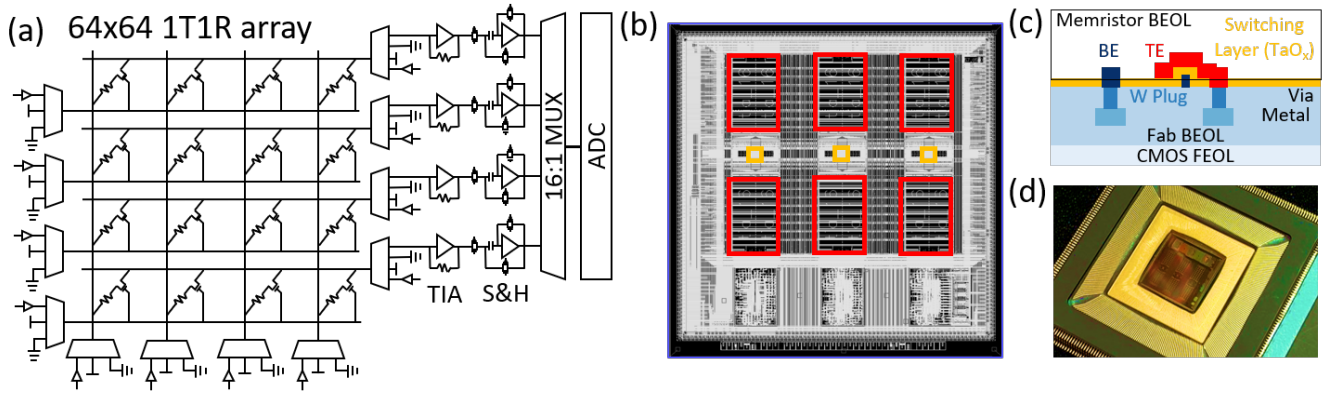

Fig. 2. Schematic of the integrated circuit (IC) comprising 3 64x64 1T1R TaOx RRAM, driving and sensing circuity. (b) Layout of the IC with highlighted RRAM arrays (orange) and sensing circuits (red). (b) Fabricated CMOS-memristive stack. (d) Picture of the realized IC.

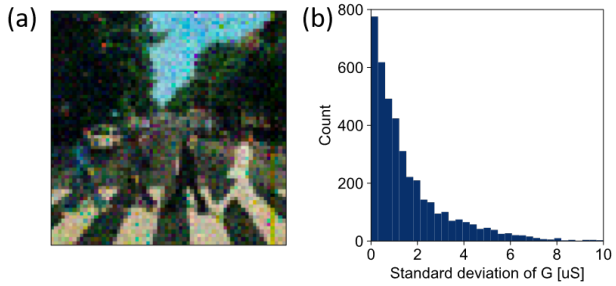

Fig. 3. (a) RGB image of Abbey Road programmed in 3 64×64 arrays in the conductance range from 0.1 μS to 100μS . (b) Distribution of fluctuation error over 100 reads.

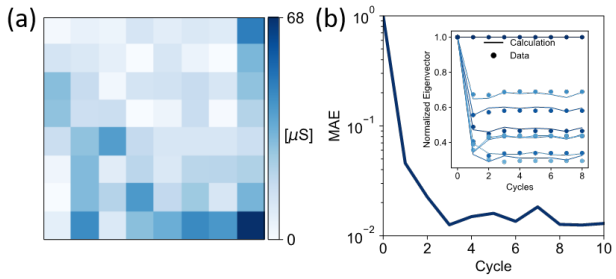

Fig. 4. (a) Programmed 8x8 matrix. (b) Mean absolute error (MAE) as function of number of cycles and measured and calculated normalized eigenvector as function of iteration cycles (inset).

web-pages (circles) connected to each other by arrows that represent citations. In this representation, page 1 cites page 2 and is cited by page 4, and so on. To find the Pagerank solution through the eigenvector calculation, a stochastic link matrix has to be computed [20] by normalizing the adjacency matrix, i.e. a Boolean matrix where element A_{ij} is equal to one if the i-th page cites the j-th page and is equal to zero in all other cases. The stochastic matrix principal eigenvalue is $\lambda = 1$ and the corresponding eigenvector is the Pagerank of the network. However null elements can introduce both algebraic issues, such as leading to a singular matrix, and technological issues, since a conductance $g = 0$ is virtually impossible to

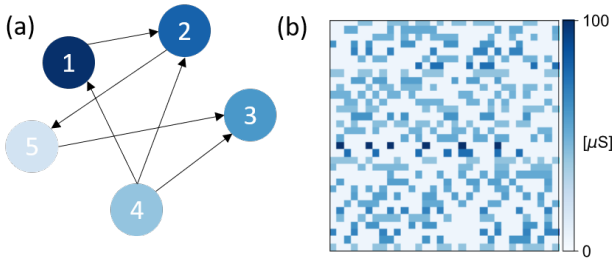

Fig. 5. (a) To retrieve the most cited webpage (circle) among a network of citation (arrows) it is possible to compute the principal eigenvector of a stochastic matrix (b) which is constructed from the adjacency matrix.

obtain, therefore '0's are replaced by relatively low numbers, namely $1/N$ [20]. Thus the analogue target matrix G can be defined as:

$$G_{ij} = \begin{cases} \frac{p \cdot A_{i,j}}{\sum_i A_{ij}} + \delta, & \text{if } \sum_i A_{ij} \neq 0 \\ 1/N, & \text{if } \sum_i A_{ij} = 0 \end{cases} \quad (2)$$

with $p = 0.85$ random walk probability and $\delta = \frac{1-p}{N}$ the probability of randomly picking a page [20]. The matrix G has a principal eigenvalue $\lambda \sim 1$ and $argmax(x_\lambda)$ gives the order of the web-pages based on their authority. Fig. 5(b) shows an example of an analogue matrix G of a 32×32 problem, which is the reference problem for this work. The matrix was programmed in a 32×32 subsection of the 64×64 memristor array, however the read matrix is different from the target one, due to both circuit and device non-idealities such as limited precision of the analogue states, noise, fluctuations [27] and devices becoming stuck in a non-ideal status, i.e. on, off, or even in an intermediate state [6]. This is shown in Fig. 6(a) where the stuck off and stuck on devices in the 32×32 sub-array are plotted in red and blue, respectively. This issue is particularly important in Pagerank problem, since a '0' or a '1' in the stochastic matrix can be seen as a removed connection or strongly added one, respectively. Thus, the graph changes significantly and the result will show a large error. Fig. 6(b) shows the measured MAE as a function of iteration cycles for computing the principal eigenvector with

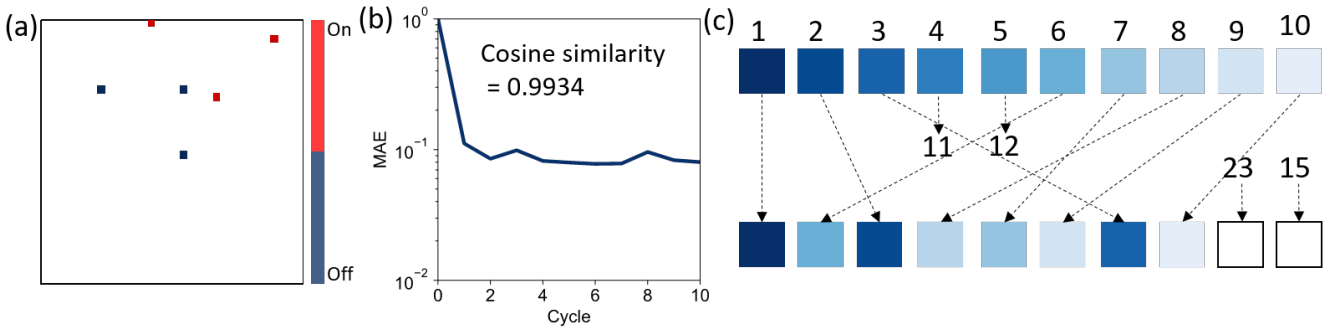

Fig. 6. (a) Stuck off (grey) and stuck on (red) devices on the 32x32 subarray with programmed problem. As an example, we considered stuck off and stuck on a device that after 1000 PV iteration could not increase its conductance to 25 μS or decrease it below 125 µS, respectively on a single sub array (b) MAE as function of the number of cycles, after 10 iterations the result shows a cosine similarity = 0.9934 with the ideal result. (c) Correlation graph between the correct ranking (top) and measured ranking (bottom).

the programmed matrix, which can not go below 10%, while the cosine similarity, namely

$$cosim(x, x') = xx'/(||x||||x'||) \tag{3}$$

which is an indicator of how close two vectors are [20], is equal to an acceptable value. The effect of this error can be understood by looking at the correlation graph in Fig. 6(c) where correct and measured ranking are shown on the top and bottom, respectively. Not only are some of the first pages mixed up, but very low authority pages (as page 23^{rd}) appear in the top 10 references, which is a misleading result. However, since solving eigenvector problems requires multiple iterations, it is important to characterize all possible sources of noise that could be iterated and amplified, not only related to the memristive device variations but also to the limited precision of analogue circuitry and the corresponding noise. In fact the TIA and the 10-bit ADC may be affected by process variations, noise, and non-linearity that could limit the precision by several bits. To measure conversion precision, an equivalent number of bits (ENOB) metric is usually adopted.

We characterize the eigenvector calculation of the 32×32 problem of Fig. 5 under various conditions to understand the impact of different noise sources. First we evaluated the error due to programming variations with respect to the target matrix of Fig. 5(b). We adopted a program/verify (PV) algorithm by applying incremental set/reset pulses to achieve a given target conductance G_{target}. Set pulses with pulsewidth $t_{set} = 10\ \mu s$, TE voltage V_{set} from 1 V to 3.1 V and gate voltage $V_{gate,set}$ from 0.5 V to 1.5 V were applied to increase the conductance, while reset pulses with pulsewidth $t_{reset} = 1\ ms$, TE voltage V_{reset} from -0.4 V to -2.6 V and a fixed gate voltage $V_{gate,reset} = 5$ V were used for decreasing the conductance. When the conductance reaches the G_{target} within an internal tolerance $\pm G_{tol,in}$ the algorithm stops while more PV iterations are applied whenever the conductance exceeds G_{target} by an external tolerance $\pm G_{tol,ext}$. Fig. 7(a) shows the resulting distribution of the programming error, namely $G_{error} = G - G_{target}$ with G measured conductance, after 1000 iterations of the PV algorithm showing a relatively small error in programming. Then we characterized the error in performing MVM, by applying random bipolar input vectors to the programmed matrix and comparing the result with the analytical product of the same random vectors with the read matrix G. Fig. 7(b) shows the distribution of the MVM error, confirming an inherent stochasticity of the operation [26]. Finally, we characterize the resulting ultimate noise floor (NF), namely all the background noise including CMOS non-linearity (such as IR drops), noise and memristor read-to-read fluctuations. To this purpose, we measured the conductance matrix several times and recorded the variation between multiple reads whose distribution is plotted in Fig. 7(c). Note that this noise is not only due to random fluctuations of memristive devices [27], since single devices characterization has shown little read-to-read variation with $\sigma_G < 0.5\ \mu S$ [23] compared with a $\sigma_{NF} = 1.5\ \mu S$ measured in this experiment. To compare the various contributions, we applied the same experimental procedure in the calculation of the eigenvector of matrix in Fig. 5(b) and evaluated the MAE due to each source, which is reported in Fig. 7(d). Final results suggest that the error is mainly due to programming variation, thus supporting the relevance of an improved analogue programming algorithm.

IV. Redundancy

To limit the impact of programming variations and stuck devices, we first adopt a redundancy technique where multiple memristors connected in parallel are used as a single matrix entry [28], [29]. Figure 8(a) shows the concept where the problem matrix G_p (same as Fig. 5(b)) is addressed in 4 different array areas, leading to a 64×64 target matrix G_{target} to be programmed in the memristor array. Fig. 8(b) shows a conceptual outcome, where variations are still present in the array. The equivalent matrix is given by the average of these 4 sub-matrixes, namely G_r, obtained by reducing the input voltage by a factor 4 and summing the currents from columns containing the same coefficient. Figure 8(c) shows the distribution of $G_{error} = G_p - G_r$ for increasing redundancy factor M namely the number of devices used for a single coefficient, demonstrating the reduction of the error even with a small M. Redundancy is particularly effective in the case of stuck on/off cells. For example, it is possible to compensate a device stuck off with a redudancy $M = 2$, by

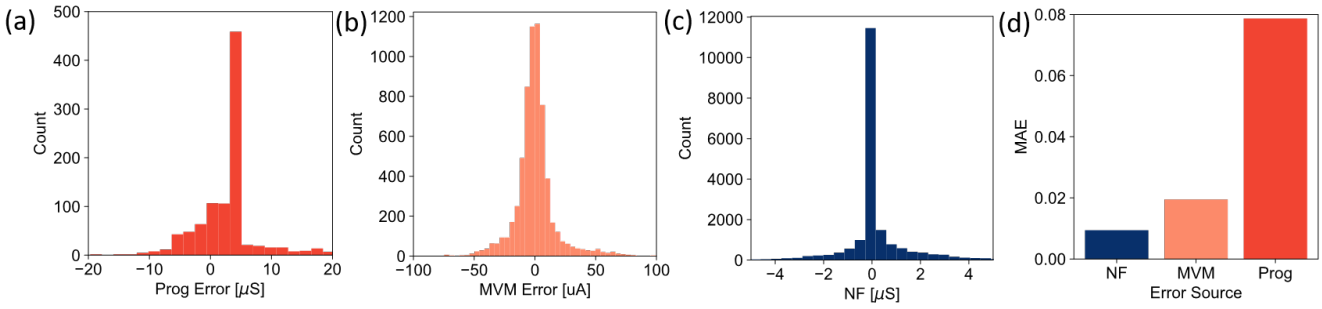

Fig. 7. Distribution of different sources of error, (a) programming error, namely $G_{programmed} - G_{target}$, of the 32×32 memristors, (b) error in the MVM of random vector inputs and 32×32 matrix of Fig. 5(b), (c) remaining noise due to other memristor and CMOS variability sources. (d) Resulting MAE in computing eigenvectors due to different noise sources.

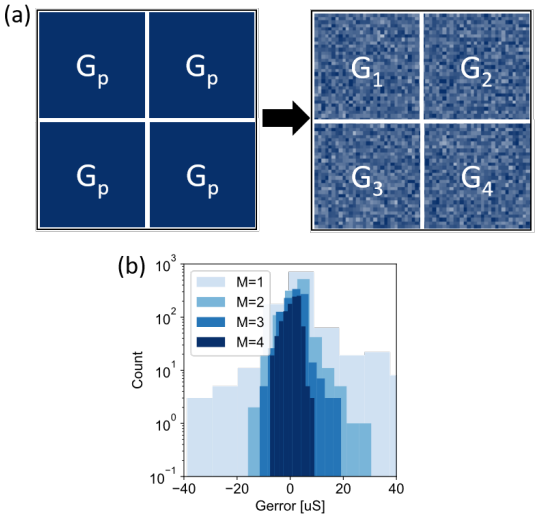

Fig. 8. Redundancy. (a) The problem matrix G_p is programmed in 4 different sub-arrays resulting in 4 different programming matrices whose average $G_r = (G_1 + G_2 + G_3 + G_4)/4$ gives the overall matrix. (b) Programming error as function of the redundancy factor M.

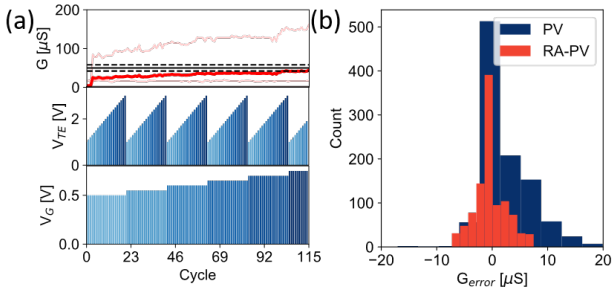

Fig. 9. Redundancy aware PV. (a) Conductance (top) with light red showing single traces for $M = 4$ conductance and strong red line the average conductance, TE (center) and gate voltage (bottom) as function of programming cycles. (b) Comparison of the conductance error for a simple PV (blue) and redundancy aware PV. The decrease of the positive G_{error} is due to stuck-on devices being more frequent than stuck-off devices in the PV case, which were then corrected with the RA-PV algorithm

assuming a state in which $G_1 = G_2 = G_{target}$ but is highly probable to reach $G_1 = 0$ and $G_2 = 2G_{target}$, such that $G_{MSB} = G_1 + G_2 = G_{target}$.

To fully take advantage of redundancy for suppressing the effect of stuck cells, it is necessary to develop a redundancy-aware program/verify (RA-PV) algorithm, which is illustrated in Fig. 9(a). The desired conductance G_{target} is reached by averaging M different memristive conductance (top), while the same programming pulses namely TE voltage V_{TE} (center) and gate voltage V_G (bottom) are applied to all the M devices. At each iteration, RA-PV reads the conductance in all the M sub-arrays, then it makes a decision of either setting, resetting or stopping programming each entry. In this way, if a stuck device is found in the last of the M sub-array, at the next iteration all the other M-1 devices will be programmed in order to compensate for the new error. Fig. 9(b) shows the distributions of G_{error}, demonstrating an improvement larger then $2\times$ for RA-PV compared to PV. Note that RA-PV is also beneficial for endurance, since less cycles are needed to program the desired conductance. The algorithm can be used both for improving accuracy and reducing device stress.

V. ANALOGUE SLICING

To further increase the programming quality, we adopted an analogue slicing technique. Slicing is a well-known strategy [12], [24] to improve the number of bits in an MVM operation. With a memristor capable of representing k bits and a specification requirement of $N_b = 2k$ bits, it is possible to write the k most significant bits (MSB) on a device and the k least significant bits on the second device which are connected in parallel. By properly setting the ADC, it is thus possible to perform $2k$-bit precision. Taking inspiration from this, we developed a new analogue slicing (AS) technique. First, the target matrix G_p is written into a sub-array resulting in matrix G_r, then an error matrix $G_\epsilon = G_p - G_r$ is computed as shown in Fig. 10(a). Note that G_ϵ includes both positive and negative entries. Positive and negative errors are multiplied to a magnification factor α and β, respectively, fit in the conductance window (i.e. 1-100 μS). Finally, the resulting error contributions namely $G_{\epsilon+}$ and $G_{\epsilon-}$ are programmed with the same M factor of G_r and according to the RA-PV algorithm. Note that G_ϵ compensate the errors in G_r including stochastic conductance variations and deterministic errors such as IR-drop errors. However, to further increase the

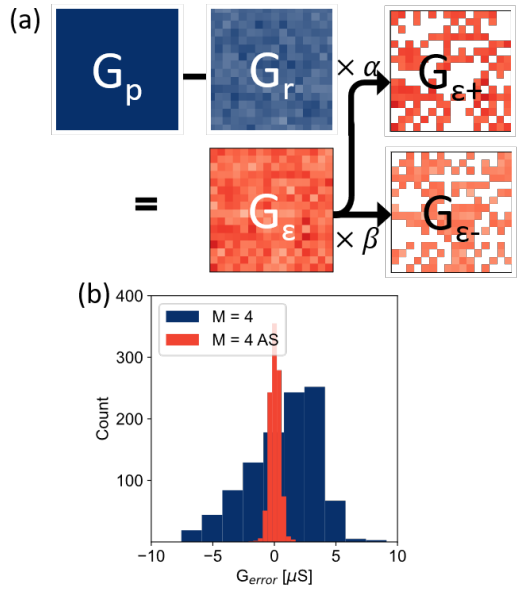

Fig. 10. analogue slicing. (a) The measured average matrix G_{MSB} is compared with the problem matrix G_p to obtain the error matrix G_ϵ which is divided in its positive and negative part, magnified to fit the maximum conductance window and programmed. (b) Comparison of the error in programming G_p without analogue slicing (blue) and with analogue slicing (red).

MVM accuracy, a linear correction is applied to reduce as much as possible the impact of IR drop. Fig. 10(b) shows the distribution of programming error G_{error} with and without AS considering $M = 4$, indicating a clear improvement in precision by AS. In fact, the standard deviation $\sigma_{Gerror} = 0.4$ μS is significantly smaller than the nNF standard deviation $\sigma_{NF} = 1.5\mu S$ which now becomes the dominant factor in the IMC precision.

VI. Conclusion

We demonstrated in-memory accelerated solution of eigenvectors calculated in a fully-integrated hybrid CMOS-memristive system. We characterized the sources of noise and developed a programming methodology based on redundancy and analogue slicing to improve the programming performance. The results indicate that programming errors can be reduced below the background noise thus significantly improving the precision of IMC circuits. These results support the use of the hybrid CMOS/memristor IC for analogue computing acceleration.

Acknowledgment

This work was supported in part by the European Research Council (grant ERC-2014-CoG-648635-RESCUE and grant ERC-2018-PoC- 842472-CIRCUS).

References

[1] E. Strubell, A. Ganesh and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP", Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pagg. 3645-3650, jul. 2019, doi:10.18653/v1/P19-1355

[2] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," Nat Electron, vol. 1, no. 1, pp. 22–29, Jan. 2018, doi: 10.1038/s41928-017-0006-8.

[3] N. P. Jouppi et al., "In-Datacenter Performance Analysis of a Tensor Processing Unit," in Proceedings of the 44th Annual International Symposium on Computer Architecture - ISCA '17, Toronto, ON, Canada, 2017, pp. 1–12, doi: 10.1145/3079856.3080246.

[4] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," Nat Electron, vol. 1, no. 6, pp. 333–343, Jun. 2018, doi: 10.1038/s41928-018-0092-2.

[5] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," Nature Nanotech, vol. 8, no. 1, pp. 13–24, Jan. 2013, doi: 10.1038/nnano.2012.240.

[6] D. Ielmini and G. Pedretti, "Device and Circuit Architectures for In-Memory Computing", Advanced Intelligent Systems, vol. 2, n. 7, pag. 2000040, jul. 2020, doi: 10.1002/aisy.202000040.

[7] M. Hu et al., "Memristor-Based analogue Computation and Neural Network Classification with a Dot Product Engine," Adv. Mater., vol. 30, no. 9, p. 1705914, Mar. 2018, doi: 10.1002/adma.201705914.

[8] C. Li et al., "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," Nat Commun, vol. 9, no. 1, p. 2385, Dec. 2018, doi: 10.1038/s41467-018-04484-2.

[9] C. Li et al., "analogue signal and image processing with large memristor crossbars," Nat Electron, vol. 1, no. 1, pp. 52–59, Jan. 2018, doi: 10.1038/s41928-017-0002-z.

[10] F. Cai et al., "Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks," Nat Electron, Jul. 2020, doi: 10.1038/s41928-020-0436-6.

[11] G. Pedretti et al., "A Spiking Recurrent Neural Network With Phase-Change Memory Neurons and Synapses for the Accelerated Solution of Constraint Satisfaction Problems," IEEE J. Explor. Solid-State Comput. Devices Circuits, vol. 6, no. 1, pp. 89–97, Jun. 2020, doi: 10.1109/JXCDC.2020.2992691.

[12] M. A. Zidan et al., "A general memristor-based partial differential equation solver", Nat Electron, vol. 1, n. 7, pagg. 411–420, jul. 2018, doi: 10.1038/s41928-018-0100-6.

[13] M. Le Gallo et al., "Mixed-precision in-memory computing", Nat Electron, vol. 1, n. 4, pagg. 246–253, apr. 2018, doi: 10.1038/s41928-018-0054-8.

[14] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, and D. Ielmini, "Solving matrix equations in one step with cross-point resistive arrays", Proc Natl Acad Sci USA, vol. 116, n. 10, pagg. 4123–4128, mar. 2019, doi: 10.1073/pnas.1815682116.

[15] Z. Sun, G. Pedretti, P. Mannocci, E. Ambrosi, A. Bricalli, and D. Ielmini, "Time Complexity of In-Memory Solution of Linear Systems," IEEE Trans. Electron Devices, vol. 67, no. 7, pp. 2945–2951, Jul. 2020, doi: 10.1109/TED.2020.2992435.

[16] P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," Nature, vol. 577, no. 7792, pp. 641–646, Jan. 2020, doi: 10.1038/s41586-020-1942-4.

[17] F. Cai et al., "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," Nat Electron, vol. 2, no. 7, pp. 290–299, Jul. 2019, doi: 10.1038/s41928-019-0270-x.

[18] C. Li et al., "CMOS-integrated nanoscale memristive crossbars for CNN and optimization acceleration", in 2020 IEEE International Memory Workshop (IMW), Dresden, Germany, may 2020, pagg. 1–4, doi: 10.1109/IMW48823.2020.9108112.

[19] K. Bryan and T. Leise, "The $25,000,000,000 Eigenvector: The Linear Algebra behind Google", SIAM Rev., vol. 48, n. 3, pagg. 569–581, jan. 2006, doi: 10.1137/050623280.

[20] Z. Sun, E. Ambrosi, G. Pedretti, A. Bricalli, and D. Ielmini, "In-Memory PageRank Accelerator With a Cross-Point Array of Resistive Memories", IEEE Trans. Electron Devices, vol. 67, n. 4, pagg. 1466–1470, apr. 2020, doi: 10.1109/TED.2020.2966908.

[21] C. Mackin, H. Tsai, S. Ambrogio, P. Narayanan, A. Chen and G. W. Burr,"Weight Programming in DNN analogue Hardware Accelerators in the Presence of NVM Variability", Adv. Electron. Mater. 2019, 5, 1900026, doi:10.1002/aelm.201900026.

[22] G. Pedretti,et. al., "Redundancy and analogue slicing for precise in-memory machine learning - Part II: Applications and benchmark"

[23] X. Sheng et al., "Low-Conductance and Multilevel CMOS-Integrated Nanoscale Oxide Memristors", Adv. Electron. Mater., vol. 5, n. 9, pag. 1800876, sep. 2019, doi: 10.1002/aelm.201800876. 2001

[24] A. Shafiee et al., "ISAAC: A Convolutional Neural Network Accelerator with In-Situ analogue Arithmetic in Crossbars", in 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, South Korea, jun. 2016, pagg. 14–26, doi: 10.1109/ISCA.2016.12.

[25] I. Inglis, "Nothing You Can See That Isn't Shown: the album covers of the Beatles", Popular Music 20 (1), pagg. 83-97, Cambridge University Press, 2001

[26] M. R. Mahmoodi, M. Prezioso, and D. B. Strukov, "Versatile stochastic dot product circuits based on nonvolatile memories for high performance neurocomputing and neurooptimization", Nat Commun, vol. 10, n. 1, pag. 5113, dec. 2019, doi: 10.1038/s41467-019-13103-7.

[27] S. Ambrogio, S. Balatti, V. McCaffrey, D. C. Wang, e D. Ielmini, "Noise-Induced Resistance Broadening in Resistive Switching Memory—Part II: Array Statistics", IEEE Trans. Electron Devices, vol. 62, n. 11, pagg. 3812–3819, nov. 2015, doi: 10.1109/TED.2015.2477135.

[28] D. Garbin et al., "HfO2-Based OxRAM Devices as Synapses for Convolutional Neural Networks", IEEE Trans. Electron Devices, vol. 62, n. 8, pagg. 2494–2501, ago. 2015, doi: 10.1109/TED.2015.2440102.

[29] I. Boybat et al., "Neuromorphic computing with multi-memristive synapses," Nat Commun, vol. 9, no. 1, p. 2514, Dec. 2018, doi: 10.1038/s41467-018-04933-y.