HAL
open science

Experimental Demonstration of Multilevel Resistive Random Access Memory Programming for up to Two Months Stable Neural Networks Inference Accuracy

Eduardo Esmanhotto, Tifenn Hirtzlin, Djohan Bonnet, Niccolo Castellani, Jean-Michel Portal, D. Querlioz, Elisa Vianello

To cite this version:

Eduardo Esmanhotto, Tifenn Hirtzlin, Djohan Bonnet, Niccolo Castellani, Jean-Michel Portal, et al.. Experimental Demonstration of Multilevel Resistive Random Access Memory Programming for up to Two Months Stable Neural Networks Inference Accuracy. Advanced Intelligent Systems, 2022, 10.1002/aisy.202200145 . hal-03861116

HAL Id: hal-03861116

https://cnrs.hal.science/hal-03861116

Submitted on 19 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ADVANCED INTELLIGENT SYSTEMS Open Access
www.advintellsyst.com

Experimental Demonstration of Multilevel Resistive Random Access Memory Programming for up to Two Months Stable Neural Networks Inference Accuracy

*Eduardo Esmanhotto, Tifenn Hirtzlin, Djohan Bonnet, Niccolo Castellani, Jean-Michel Portal, Damien Querlioz, and Elisa Vianello**

Crossbars of resistive memories, or memristors, provide a road to reduce the energy consumption of artificial neural networks, by naturally implementing multiply accumulate operations, their most basic calculations. However, a major challenge of implementing robust hardware neural networks is the conductance instability over time of resistive memories, due to the local recombination of oxygen vacancies. This effect causes resistive memory-based neural networks to rapidly lose accuracy, an issue that is sometimes overlooked. Herein, this conductance instability issue is shown, which can be avoided without changing the material stack of the resistive memory by exploiting an original programming strategy. This technique relies on program-and-verify loops with appropriately chosen wait times and ensures that the resistive memories are programmed into states with stable filaments. To test the strategy, a 32 × 32 in-memory computing system, fabricated in a hybrid complementary metal-oxide-semiconductor (CMOS)/hafnium oxide technology, is programmed to classify heart arrhythmia from electrocardiogram. When the resistive memories are programmed conventionally, the system loses accuracy within hours. In contrast, when using this technique, the system maintains an accuracy of 95% over more than 2 months. These results highlight the potential of resistive memory for the implementation of low-power neural networks with long-term stability.

1. Introduction

In recent years, artificial intelligence has reached significant milestones with the development of deep neural networks, but it suffers from a major limitation: its considerable energy consumption.[1] This limitation is primarily due to the energy cost of exchanging information between computation and memory units.[2,3] Memristors, also called resistive random access memories (RRAMs) in industrial laboratories, now provide an opportunity to increase the energy efficiency of AI dramatically. In contrast to the complementary metal-oxide-semiconductor (CMOS)-based memories such as static or dynamic random access memories, which store one bit per unit cell, they can be programmed to intermediate states between their lowest and highest resistance values, allowing memorizing the synaptic weights of a neural network in a particularly compact manner.[4] In addition, using the fundamental laws of electric circuits, arrays of memristors can implement deep learning's most basic operation, multiply and accumulate (MAC): the multiply operation corresponds to Ohm's law, whereas the accumulate operation corresponds to Kirchhoff's current law. This type of "in-memory" computation consumes less power than equivalent digital implementations[5–9]: the computation is performed directly within memory, allowing the suppression of the energy associated with weight movement.[4,10,11] Moreover, nonvolatility offers an instant on/off feature: memristor-based systems can perform inference immediately after being turned on, allowing to cut the power supply entirely as soon as the system is not used.

E. Esmanhotto, T. Hirtzlin, D. Bonnet, N. Castellani, E. Vianello
CEA-Leti
Université Grenoble Alpes
38400 Grenoble, France
E-mail: elisa.vianello@cea.fr

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202200145.

J.-M. Portal
IM2NP
Aix-Marseille Université
13007 Marseille, France

D. Querlioz
CNRS
Centre de Nanosciences et de Nanotechnologies
Université Paris-Saclay
91120 Palaiseau, France

© 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH. This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.1002/aisy.202200145

26404567, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202200145 by Cochrane France, Wiley Online Library on [19/11/2022]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

However, RRAM devices are far from ideal analog memories: they suffer from conductance instability issues mainly due to the local random diffusion of oxygen vacancies in and out of the conductive filament.[12–15] The chemical nature of the materials in the memory stack changes the magnitude of the instability (see Figure S1, Supporting Information). Resistive memories based on TaO_x and HfAlO,[14,16,17] for example, show a slight advantage over HfO_x, but the conductance instability remains inevitably present in the tails of the distributions regardless of the memory stack materials.[14,18] Moreover, there is an experimental interplay between retention and endurance: better conductance stability is correlated with poorer endurance.[19,20]

Until recently, the conductance instability issue has been relatively overlooked, as industrial developments around RRAMs have focused on their use as single-level cells, where RRAMs are used with only two levels meaning zero and one.[21–23] In this case, conductance instability is not a major concern, as long the zero and one levels remain well separated, and 10 years retention can be obtained.[24] In contrast, conductance instability is a major concern in the analog or multilevel uses envisioned for neural network implementation. Conductance instability may be overlooked during the development of a technology when sample cells are measured, mainly due to insufficient statistics. Instability occurs at the tails of the conductance distribution of arrays exceeding several thousands of devices. Conductance instability varies in magnitude depending on technologies, but recent works have revealed that it has a major impact on the medium-term and long-term accuracy of RRAM-based neural networks. In the study by Xi et al.,[12] a neural network with HfO_x RRAM trained on handwritten digit recognition mixed national institute of standards and technology (MNIST) loses 7% points in accuracy in only 1 s. Zhao et al.[25] estimate that the accuracy rate of a neural network programmed on a Al:HfO_x RRAM degrades from 99% to 93% in 23 days at 85 °C. Lin et al.[26] show a degradation from 98% to only 10% accuracy with tungsten oxide in 4 days. These results mean that an analog or multilevel RRAM neural network would need to be reprogrammed very regularly to fight this conductance instability. Alternatively, Lin et al.[26] propose embedding conductance instability–compensation circuitry, with a very high overhead cost and still incapable of eliminating accuracy reduction entirely.

In this work, we propose to solve this issue by transforming the way RRAMs are programmed. We introduce a robust programming technique for RRAMs that combines a smart multistep programming methodology with a wait time able to compensate for short-term conductance instability. We program an analog neural network on a fabricated hybrid hafnium oxide RRAM/CMOS fully integrated system using this technique, which implements in-memory computing. The resulting neural network maintains accuracy without any degradation over 2 months on a task of recognition of heart arrhythmia from electrocardiogram (ECG) recording, demonstrating the potential of properly programmed RRAM for long-term stable neural network accelerators.

Chalcogenide glass-based phase-change memories (PCMs) are another technology that has been widely studied for analog in-memory neural network implementation, due to the possibility to adjust the conductance of these devices finely.[5] PCMs suffer from a major conductance drift issue due to structural relaxation.[27] Current proposals are efforts toward drift-immune devices using projected architectures[28] or compensation techniques at the circuit level involving complex and high-energy consumption overheads.[29,30] Our work suggests that RRAM might be used for long-term multilevel synaptic applications with simpler mitigations than PCM.

The article is organized as follows. We first introduce the RRAM technology used throughout this work, its conductance instability characteristics, and our original programming technique. We then program a complete neural network on a hybrid CMOS/RRAM in-memory computing system and show that it maintains accuracy over two months when RRAMs have been programmed with our programming technique.

2. Overcoming Conductance Instability with a Dedicated Programming Strategy

Our work relies on a resistive RAM technology integrated into the back end of line (BEOL) of a 130 nm foundry CMOS process. A titanium nitride bottom electrode is defined on the top of the fourth metal level (copper) of the CMOS process. A chemical mechanical polishing touch is performed, and an HfO_2/Ti/TiN stack is deposited where the HfO_2 and the Ti layers are 5 and 10 nm thick, respectively (see **Figure 1**a and Experimental Section). The RRAM cells are initially in a low-conductance state (LCS) (pristine state). An oxygen-poor filamentary path is first formed electrically by soft electrical breakdown.[31] The device can then be switched between low and high conductance by field-induced migration and diffusion of the oxygen vacancies in the conductive filament. A positive voltage applied to the top electrode (SET operation) causes oxygen vacancies' migration toward the bottom electrode, inducing the transition to a high-conductance state (HCS). The conductive filament can then be disrupted with the application of a negative voltage pulse (RESET operation), inducing oxygen vacancies migration back to the top electrode, thus flipping the device into an LCS (see Figure 1c).

During the forming and SET operations, the current is limited by an n-channel metal-oxide-semiconductor field-effect transistor (nMOSFET) selector device (Figure 1b) to avoid the breakdown of the device. This current compliance determines the conductance of the HCS at the end of the SET process, and it can be adjusted by choosing the gate voltage on the nMOSFET. Therefore, it is possible to use RRAM as a multilevel cell, by modulating the value of the HCS. Unfortunately, RRAM is prone to a high level of conductance variability, meaning that the HCS obtained after an SET operation presents a relatively broad statistical distribution: the HCS measured more than 16 384 devices follows a normally distributed device-to-device conductance probability density (Figure 1d).

To program RRAMs into well-defined resistance states, a conventional technique is to rely on program-and-verify strategies: an RRAM cell is programmed multiple times, until its resistance reaches its targeted value. **Figure 2**a shows a standard program-and-verify methodology.[32] The different conductance levels are chosen in an optimized manner with regard to the properties of RRAM (we take into account the increase in conductance variability as its average value increases to allocate conductance ranges instead of dividing them equally, see Experimental

© 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202200145 by Cochrane France, Wiley Online Library on [19/11/2022]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

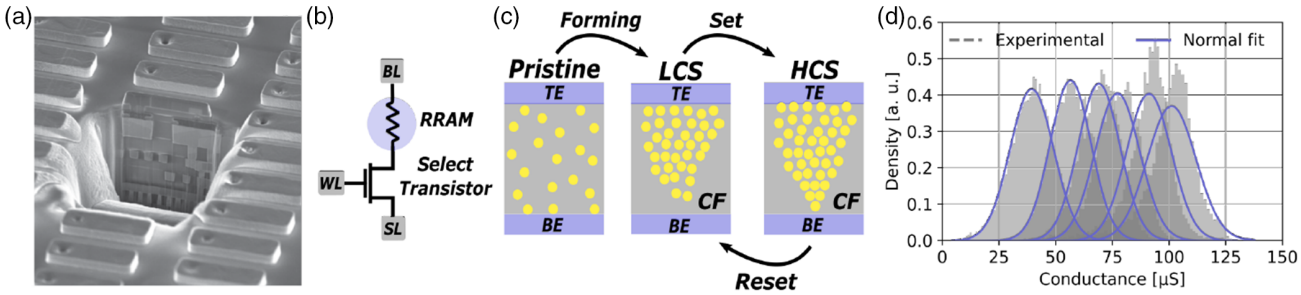

Figure 1. a) Scanning electron microscope (SEM) crosssection of the TiN/HfO_2/Ti/TiN resistive memory integrated into the BEOL of a CMOS 130 nm technology node. b) 1T1R circuit schematic. c) Oxygen vacancy-based working principle of the pristine, LCS, and HCS. d) Probability density of the conductance variability measured on 16 384 devices under six different SET programming currents without iterative programming fit with a normal distribution (blue line).

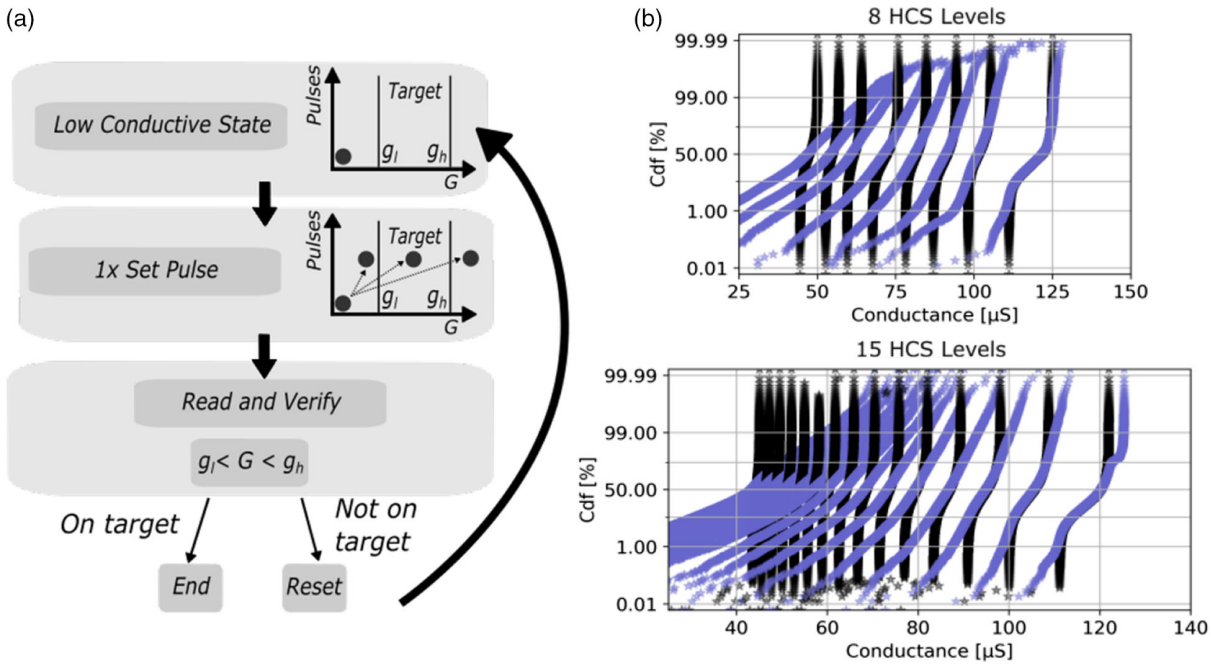

Figure 2. a) Standard iterative programming algorithm.[32] b) Conductance cumulative probability distribution (black, $t = 0$ s and blue, $t = 60$ s) for 8 and 15 distinct conductance levels programmed using the standard iterative programming algorithm. Conductance instability is observed after $t = 60$ s.

Section). Our experimental results show that this program-and-verify strategy is indeed efficient. Figure 2b illustrates the cumulative distribution of the conductance of 16 384 resistive memory cells in 8 and 15 different conductance states using the standard iterative programming algorithm shown in Figure 2a. Initially, the conductance states are separated (black curve), suggesting the possibility of using RRAM for multilevel storage. However, this strategy only works short term: many RRAM cells have conductance states that are not stable over time, meaning that they have been programmed in a state with an unstable filament. This effect is shown in Figure 2b: 60 s after programming, conductance instability over time causes the initial distributions to spread toward both higher and lower values. Regarding the lowest conductance level (the most prone to conductance instability), after 60 s, only 85% of the programmed devices remain in the target conductance range for eight HCS levels and only 70% with 15 HCS levels.

Figure 3 shows the cumulative distribution of the conductance of 425 resistive memory cells in eight different conductance

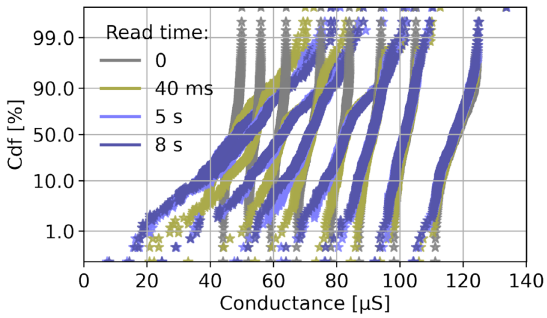

Figure 3. Cumulative distributions of 425 devices in eight different conductance levels read between $t = 0$ s (gray) and $t = 8$ s (light blue) after standard iterative programming.

© 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202200145 by Cochrane France, Wiley Online Library on [19/11/2022]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

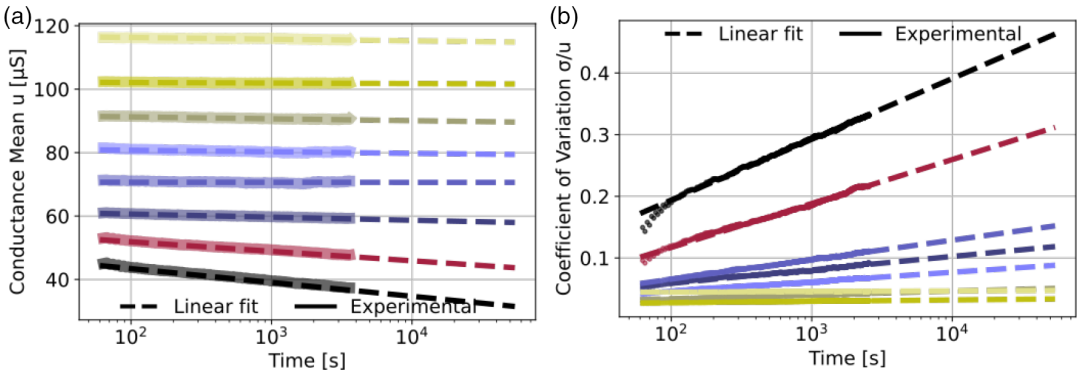

Figure 4. a) Mean value evolution over time of eight conductance distributions programmed with the standard iterative strategy. Experimental data and linear fit. b) Coefficient of variation (i.e., the ratio of the standard deviation to the mean) evolution over time of eight conductance levels programmed with the standard iterative strategy. Experimental data (symbols) and linear fit (dashed lines).

states read shortly after standard iterative programming. A spread of the conductance of the devices occurs rapidly after programming, between $t = 0$ s (gray) and $t = 8$ s (light blue). This effect occurs on the same time scale throughout the conductance range, and it is most pronounced for high-conductance values. Conductance spread then continues at a slower pace over longer time scales. Finally, **Figure 4** shows the mean and the coefficient of variation (i.e., the ratio of the standard deviation to the mean) values measured over 1 h on 4,096 devices, as well as a linear fit of the curves, showing that the conductance values continue spreading on a longer time scale. The lower the conductance is, the higher the effect of conductance instability.

The literature about hafnium oxide-based resistive memories suggests that the conductive filament may consist of oxygen vacancies (V_o) and metal precipitates.[33] The movement of the individual oxygen interstitial (O^i, scavenged by the reactive Ti top electrode during the cell stack deposition and forming process) and the random diffusion of the oxygen vacancies in or out-of the conductive filament causes conductive filament instability that is at the origin of the conductance spread.[17,34] Using ab initio simulations, Clima et al.[14] also demonstrated that the strongly relaxing conductance tails are due to the unstable conducting filaments constituted by high-energy oxygen vacancies that tend to relax in time toward lower-energy positions.

To overcome the conductance instability effect, it is essential to ensure that RRAM cells are programmed into states with stable filaments. Therefore, we proposed a dedicated programming method (**Figure 5**a), which builds on the program-and-verify technique, with the addition of a wait time of Δt after each

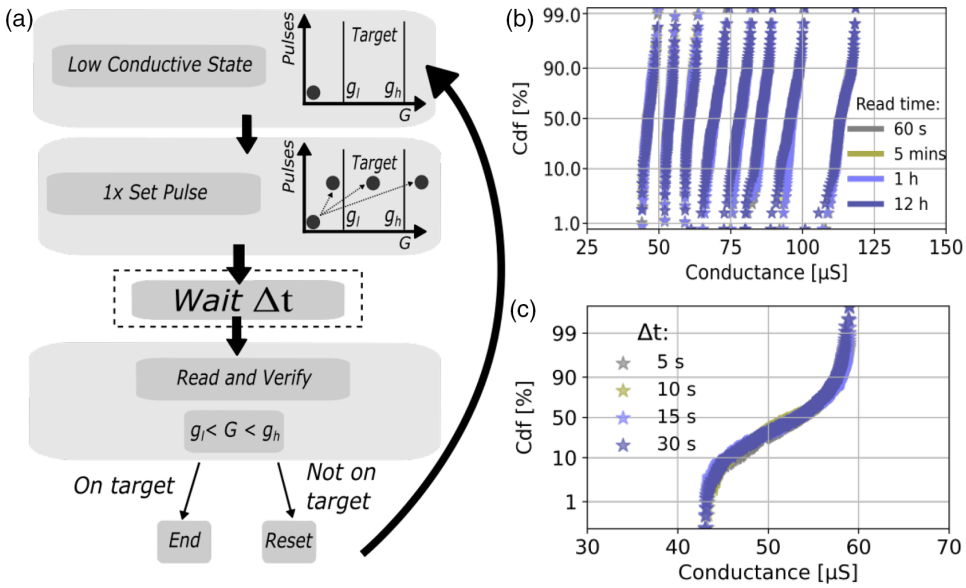

Figure 5. a) Dedicated smart programming flow. b) Conductance cumulative probability distributions for eight conductance levels programmed using the new dedicated programming technique read after 60 s and 12 h. c) Effect of the waiting time, Δt, over the conductance distribution of one level after dedicated smart programming. The plotted conductance distributions are read after 1 h.

© 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202200145 by Cochrane France, Wiley Online Library on [19/11/2022]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

SET operation and before the verify operation. This way cells that suffered from conductance instability during the waiting period are rescheduled to the next programming iteration, until they are programmed in a state with less short-term instability. The resulting cells are highly stable, even in the long term: Figure 5b shows that the conductance distributions are stable after 1 min and 12 h, in strong contrast with the previous figures.

The impact of the choice of waiting time is shown in Figure 5c. Only a minor change is observed between a wait time of 5 and 30 s. This result means that waiting 5 s is sufficient to predict the long-term stability of a programmed state; therefore, Δt is fixed to 5 s in all subsequent experiments of the article.

Our technique increases the programming time of the array, due to the addition of the wait time. It also requires, on average, three times more program-and-verify iterations steps per device than the conventional technique with no wait time (see Figure S2, Supporting Information). Consequently, the programming energy also increases by a factor of 3. Therefore, our technique is particularly appropriate for implementing accelerators of neural network inference. In this specific application, the programming operation is performed only once, and the cells are subsequently only subjected to reading operations. We validate the efficiency of our technique on such an accelerator in the next section.

3. Experimental Demonstration of a Stable RRAM-Based Neural Network

We now test our new dedicated programming technique on a complete neural network, using the fully integrated RRAM crossbar shown in **Figure 6**, capable of performing parallel MAC operations (see Experimental Section for fabrication details). This integrated system is fabricated in a 130 nm CMOS commercial process, with RRAM integrated into the CMOS BEOL. This system implements artificial neural networks (ANNs) with analog weights and binary stochastic neurons. We use this chip to perform a medical artificial intelligence task: identifying the heart arrhythmia from ECG recordings (using the ECG database of the study by Moody et al.[35], as shown in **Figure 7**a). Our system uses a two-layer perceptron ANN (see Experimental Section) and can differentiate between normal heart beat and four different types of heart arrhythmia (see Experimental Section).

Figure 8a shows the accuracy of the in-memory neural network, right after it has been programmed, using the conventional programming technique of Figure 2a. The measured results, consistent with simulations, are presented for different experiments with different number of conductance levels used in

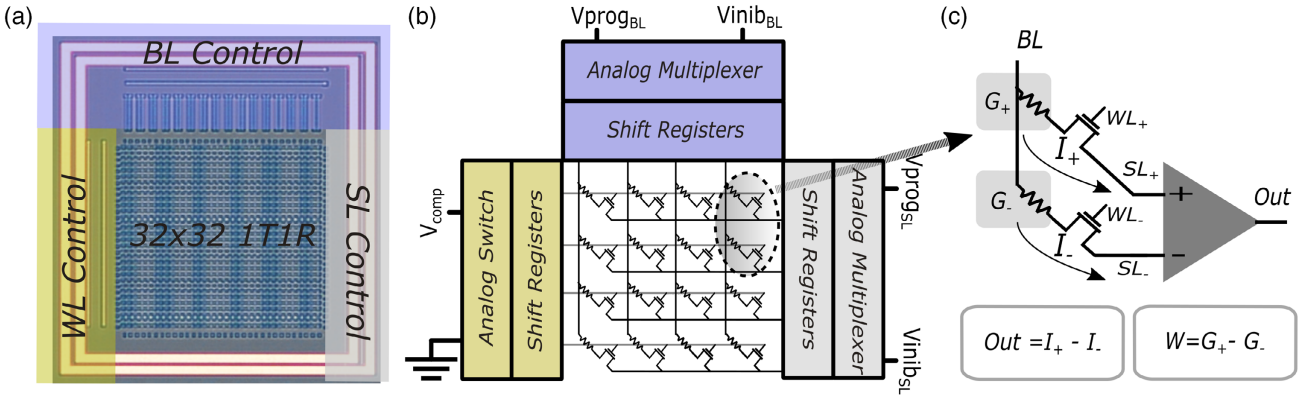

Figure 6. a) Photograph of the fabricated chip. b) Detailed version of the crossbar circuit architecture. Independent shift register chains allow the control of the bit lines, SLs, and WLs separately. Analog multiplexers and switches drive the voltage and current necessary to operate the array. c) A single synaptic weight is encoded by the conductance difference between positive and negative sets of devices, located in adjacent SLs.

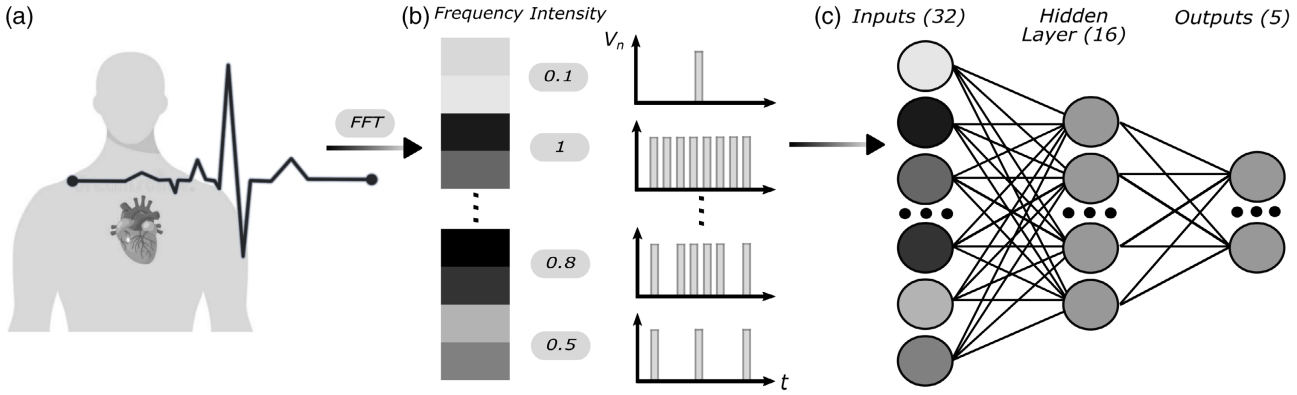

Figure 7. a) Example of an ECG signal. b) Conversion of the input analog signal into binary inputs. c) A two-layer perceptron architecture.

© 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202200145 by Cochrane France, Wiley Online Library on [19/11/2022]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

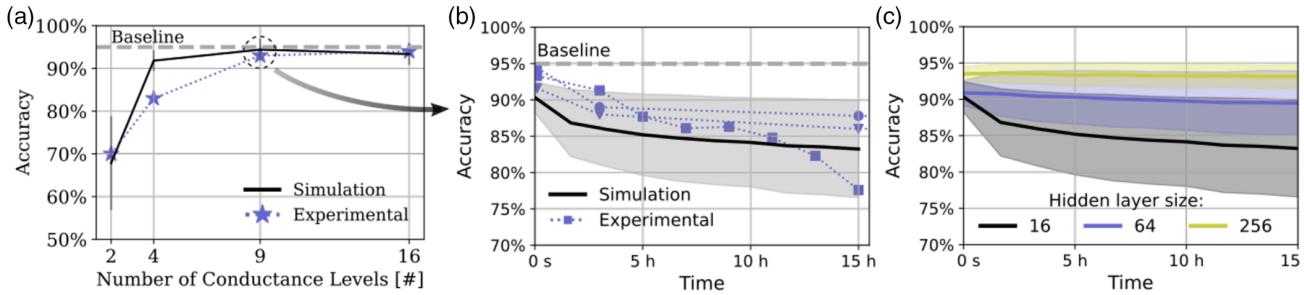

Figure 8. a) Inference accuracy (simulated and experimental) on ECG dataset for different number of conductance levels per device. b) Inference accuracy over time (simulated and experimental) on ECG dataset with nine conductance levels per device. Experiments have been performed on three different RRAM crossbar dies. c) Simulated inference accuracy value for different sizes of the neural network (number of neurons in the hidden layer). The baseline is calculated via software taking into account binary stochastic inputs and using 32 bits floating-point synaptic weights.

RRAM. Beyond nine levels per cell (eight HCS levels and one LCS), the inference accuracy reaches the maximum value of 95%. Therefore, we focus on crossbars programmed with nine levels in the rest of the article. For applications requiring more than 16 levels per memory cell (i.e., the limit of our technology, see Figure 2), a technique called bit slicing can be adopted. The key idea is to map a single logical row of the matrix across multiple physical rows of the array. This technique enables full-precision computation with limited precision memory elements.[36]

However, as expected, when the devices have been programmed conventionally, this accuracy is not maintained long term. Figure 8b shows the accuracy drop over time due to the conductance spread. This experiment was repeated on three different crossbar arrays and shows consistent results: a strong degradation of the accuracy of the neural network in a few hours. The behavioral model (based on the linear fit of Figure 4) accurately reproduces the experimental data.

In contrast, crossbar arrays programmed with our dedicated programming technique maintain much more stable accuracy. **Figure 9**, also based on the measurements on three different arrays, shows that the dedicated programming strategy guarantees an accuracy of 95% stable over two months. The dies were stored at room temperature, unpackaged, under a normal atmosphere over 2 months. This result presents for the first time that an analog RRAM-based fabricated neural network showed stable accuracy on a long-term period and therefore highlights the potential of RRAM programmed appropriately for stable operation.

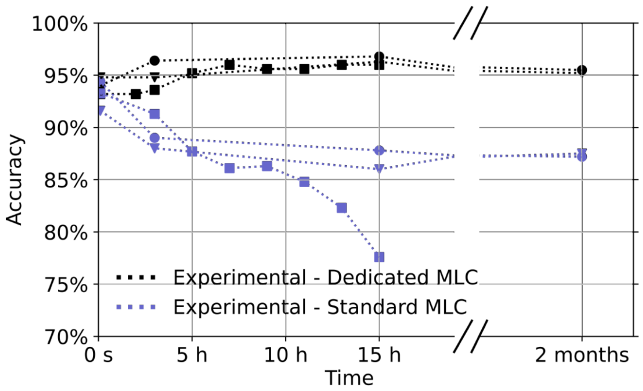

Figure 9. Experimental inference accuracy of ECG dataset using nine conductance levels programmed with standard iterative strategy (Figure 2a, blue) and the new dedicated smart programming flow (Figure 5a, black) from three different RRAM crossbar dies.

We also estimated how the RRAM technology would behave in larger neural networks. Figure 8c shows the simulated evolution of the inference accuracy over time of a neural network on an ECG dataset for neural networks with a hidden layer of 16, 64, and 256 neurons, without the dedicated programming technique. The three structures require a crossbar array of 1, 4, and 16 kilo devices, respectively. The temporal conductance instability strongly degrades the accuracy for the smaller neural networks, whereas it has a much smaller effect for the larger neural networks.[32] This mechanism is related to the redundancy inherent to large neural networks and the redundant synaptic weights provide resilience to temporal changes in conductance. This possibility of compensating the conductance spread effect by increasing the size of the neural network is, however, not satisfactory, as it requires larger memory arrays to solve a given task. Second, it works well in a fully connected layer but would be less effective in different neural networks architectures such as convolution filters with less redundancy.

4. Conclusion

We have experimentally demonstrated a two-layer perceptron inference based on an RRAM crossbar array with a new multilevel programming algorithm to enable stable accuracy over time. The new programming method is a "smart" programming method that reduces conductance variability and stabilizes programmed levels up to more than 2 months. The two-layer perceptron classifies ECG recording between normal heart beat and four classes of heart arrhythmia with an accuracy of 95%, when programming the RRAMs with nine levels per cell. This result validates the potential of analog RRAM for in-memory deep learning inference accelerator able to retain accuracy on the long term without adjusting the RRAM material stack.

These results also highlight the complexity of the time scales intrinsic to RRAM devices. Conventional programming techniques can be very effective over short-term periods and can be useful, for example, for learning accelerators where devices

© 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202200145 by Cochrane France, Wiley Online Library on [19/11/2022]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

get reprogrammed frequently. In contrast, stable long-term inference requires specific efforts.

5. Experimental Section

Fabrication/Integration: The resistive RAM technology used in our experiments was an oxide-based RRAM fully integrated in the BEOL of a 130 nm commercial CMOS process (see Figure 1a). The thick oxide nMOSFET select transistor was 1 µm wide and a 0.5 µm length. The RRAM active layer consisted of HfO_2, deposited at 300 °C by atomic layer deposition with $HfCl_4$ and H_2 precursors in ASM polygon pulsar chamber. The metal TiN bottom and Ti top electrodes were deposited by physical vapor deposition. The HfO_2 and Ti layers were 5 nm thick and had a 300 nm diameter mesa structure.

Device and Circuit Measurements: For programming and reading the RRAM devices, voltage pulses were generated off-chip by an RIFLE NplusT engineering test system, which incorporates a digital sequencer, 100 MHz arbitrary waveform generators, 70 Msample/s cell current measurement capability, and a C++ programmable computer. The computer configured pulses applied by the arbitrary waveform generator to the RRAM cells. All of these signals were interfaced to our 200 mm wafer through a 25-pin probe card, which contacted 25 metal pads integrated on top of the BEOL of the wafer. Using this setup, the computer was therefore able to program the conductance states of devices integrated in the array, read the resulting states, and then, based on these results, reprogram the devices in the array. In this fashion, we could implement the proposed program and verify algorithms.

In-Memory Computing Chip: Two different versions of fabricated RRAM memory arrays were used in the presentation of this article. These two arrays featured different sizes and routing systems. In both arrays, each resistive memory was connected to the drain terminal of a transistor, in a one-transistor–one-resistor (1T1R) configuration. The transistor, used as a selector, was essential to control the programming current allowing multilevel programming of the RRAM devices. The first (used in Figure 1, 2) was a 16 384 device array of 1T1R structures, only individually addressable, suitable for extensive statistical analysis (Section 2 of the article). The second one was smaller (1,024 1T1R structures, arranged in a 32 row and 32 column crossbar structure) but more flexible; this array enabled the selection of multiple memory points along the selected word line (WL), allowing for in-memory computing (Section 3 of the article). The current that flowed through each source line (SL) was the dot product of the input voltage vector (**V**) applied on the bit lines and the corresponding column memory conductance vector (**g**). The fabricated circuit (Figure 6a) details are shown in Figure 6b. Digital drivers were used to select single or multiple cells in parallel controlling the transistors WL. The flexibility of this structure allowed addressing the bit lines, SLs, and WLs independently using digital registers (scan chains). Each device was programmed sequentially through analog multiplexers designed to drive the programming signals with minimal voltage drop.

The correspondence between synaptic weights and stored conductance is shown in Figure 6c. A single synaptic weight (G) was encoded by the conductance difference between positive (G_+) and negative (G_-) sets of devices, located in adjacent SLs. The input current of each output neuron was the difference between the current through the positive and the negative SLs.

A single voltage level was applied to the BLs; therefore, the activation function of the output neurons corresponded to a current comparison between the positive and the negative SLs. In our test system, this comparison was performed externally to the integrated circuit; in a commercial system, it could easily be implemented with a current-sensing circuit such as the one presented in other studies[37,38] between two branches.

ECG Task: The ECG recording was cut in series of 700 ms used to extract 32 features through a fast-Fourier transform. The 32 features were the input of the ANN. As the digital drivers generated only one read voltage level, each extracted feature was rescaled on a value between 0 and 1 and used as a probability. This probability was transformed into N binarized stochastic inputs V_n that were applied sequentially to the input of the first layer. The network computed the dot-product operations through the two layers, and the output of the output neurons was summed up over the number N of stochastic versions of the input V_n (Figure 7b).

The implemented perceptron featured 16 hidden neurons in the first layer and five output neurons, corresponding to the five different labels, in the second layer (Figure 7c). Our crossbar array could take 32 inputs and produce 32 outputs at a given time. To implement the two layers perceptron with the single RRAM crossbar array, two arrays were required. The input-to-first layer matrix multiplication was calculated using the one crossbar array. The 16 binary outputs were recorded and used as an input for the first-to-second layer. The matrix multiplications were calculated using 10 out of 32 rows. The 10 rows were arranged in pairs to provide five outputs.

Our hardware targeted solely inference; training was performed in software, with binarized stochastic input features presented instead of analog values.[39] Moreover, to improve the inference robustness to the variability of the RRAM technology and the analog circuits, we artificially added noise to each neuron during the training process.[40] Training was performed using 32 bits floating-point representation. The procedure was completed by transferring the learned weights to the RRAM array: the learned weights were quantized to 3 bit values and converted into the RRAM conductance levels for inference.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

This work was financially supported by the H2020 MeM-Scales project (871371). This work was also supported by the ECSEL TEMPO project (826655), the ANR grant NEURONIC (ANR-18-CE24-0009), and the ERC DIVERSE project (101043854).

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords

in-memory computing, memristors, multilevel cells, neural networks, neuromorphic, resistive random access memories

Received: June 27, 2022
Published online:

[1] *Nature* **2018**, *554*, 145, cg type: Editorial, Number: 7691 Publisher: Nature Publishing Group, Subject term: Computer science, Mathematics and computing.
[2] A. Pedram, S. Richardson, M. Horowitz, S. Galal, S. Kvatinsky, *IEEE Des. Test* **2017**, *34*, 39.
[3] D. Marković, A. Mizrahi, D. Querlioz, J. Grollier, *Nat. Rev. Phys.* **2020**, *2*, 499.

© 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202200145 by Cochrane France, Wiley Online Library on [19/11/2022]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

[4] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, D. B. Strukov, *Nature* **2015**, *521*, 7550.
[5] S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. Di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. Farinha, B. Killeen, C. Cheng, Y. Jaoudi, G. W. Burr, *Nature* **2018**, *558*, 60.
[6] D. Ielmini, G. Pedretti, *Adv. Intell. Syst.* **2020**, *2*, 2000040.
[7] A. Amirsoleimani, F. Alibart, V. Yon, J. Xu, M. R. Pazhouhandeh, S. Ecoffey, Y. Beilliard, R. Genov, D. Drouin, *Adv. Intell. Syst.* **2020**, *2*, 2000115.
[8] L. Yuan, S. Liu, W. Chen, F. Fan, G. Liu, *Adv. Electron. Mater.* **2021**, *7*, 2100432.
[9] W. Wang, W. Song, P. Yao, Y. Li, J. Van Nostrand, Q. Qiu, D. Ielmini, J. J. Yang, *iScience* **2020**, *23*, 101809.
[10] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. Yang, H. Qian, *Nature* **2020**, *577* 641.
[11] T. Dalgaty, N. Castellani, C. Turck, K.-E. Harabi, D. Querlioz, E. Vianello, *Nat. Electron.* **2021**, *4*, 151.
[12] Y. Xi, B. Gao, J. Tang, X. Mu, F. Xu, P. Yao, X. Li, W. Zhang, M. Zhao, H. Qian, H. Wu, in *2020 IEEE 4th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*, IEEE, Piscataway, NJ **2020**.
[13] V. G. Karpov, D. Niraula, I. V. Karpov, R. Kotlyar, *Phys. Rev. Appl.* **2017**, *8*, 024028.
[14] S. Clima, Y. Y. Chen, A. Fantini, L. Goux, R. Degraeve, B. Govoreanu, G. Pourtois, M. Jurczak, *IEEE Electron Dev. Lett.* **2015**, *36*, 769.
[15] E. Pérez, C. Zambelli, M. K. Mahadevaiah, P. Olivo, C. Wenger, *IEEE J. Electron Dev. Soc.* **2019**, *7*, 740.
[16] A. Fantini, L. Goux, S. Clima, R. Degraeve, A. Redolfi, C. Adelmann, C. Polimeni, Y. Chen, M. Komura, A. Belmonte, D. Wouters, M. Jurczak, in *2014 IEEE 6th Int. Memory Workshop (IMW)*, IEEE, Piscataway, NJ **2014**.
[17] B. Traoré, P. Blaise, E. Vianello, H. Grampeix, S. Jeannot, L. Perniola, B. De Salvo, Y. Nishi, *IEEE Trans. Electron Dev.* **2015**, *62*, 4029.
[18] C. Li, R. M. Roth, C. Graves, X. Sheng, J. P. Strachan, in *2020 IEEE Int. Electron Devices Meeting (IEDM)*, IEEE, Piscataway, NJ **2020**.
[19] M. Azzaz, E. Vianello, B. Sklenard, P. Blaise, A. Roule, C. Sabbione, S. Bernasconi, C. Charpin, C. Cagli, E. Jalaguier, S. Jeannot, S. Denorme, P. Candelier, M. Yu, L. Nistor, C. Fenouillet-Beranger, L. Perniola, in *2016 IEEE 8th Int. Memory Workshop (IMW)*, IEEE, Piscataway, NJ **2016**.
[20] C. Nail, G. Molas, P. Blaise, G. Piccolboni, B. Sklenard, C. Cagli, M. Bernard, A. Roule, M. Azzaz, E. Vianello, C. Carabasse, R. Berthier, D. Cooper, C. Pelissier, T. Magis, G. Ghibaudo, C. Vallée, D. Bedeau, O. Mosendz, B. De Salvo, L. Perniola, in *2016 IEEE Int. Electron Devices Meeting (IEDM)*, IEEE, Piscataway, NJ **2016**.
[21] O. Golonzka, U. Arslan, P. Bai, M. Bohr, O. Baykan, Y. Chang, A. Chaudhari, A. Chen, J. Clarke, C. Connor, N. Das, C. English, T. Ghani, F. Hamzaoglu, P. Hentges, P. Jain, C. Jezewski, I. Karpov, H. Kothari, R. Kotlyar, B. Lin, M. Metz, J. Odonnell, D. Ouellette, J. Park, A. Pirkle, P. Quintero, D. Seghete, M. Sekhar, A. Sen Gupta, et al., in *2019 Symp. on VLSI Technology*, IEEE, Piscataway, NJ **2019**, pp. T230–T231.
[22] S.-S. Sheu, P.-C. Chiang, W.-P. Lin, H.-Y. Lee, P.-S. Chen, Y.-S. Chen, T.-Y. Wu, F. T. Chen, K.-L. Su, M.-J. Kao, et al., in *2009 Symp. on VLSI Circuits*, IEEE, Piscataway, NJ **2009**, pp. 82–83.
[23] A. Benoist, S. Blonkowski, S. Jeannot, S. Denorme, J. Damiens, J. Berger, P. Candelier, E. Vianello, H. Grampeix, J. F. Nodin, E. Jalaguier, L. Perniola, B. Allard, in *2014 IEEE Int. Reliability Physics Symp.* IEEE, Piscataway, NJ **2014**, pp. 2E.6.1–2E.6.5.
[24] W. Kim, S. I. Park, Z. Zhang, Y. Yang-Liauw, D. Sekar, H.-S. P. Wong, S. S. Wong, in *2011 Symp. on VLSI Technology-Digest of Technical Papers*, IEEE, Piscataway, NJ **2011**, pp. 22–23.
[25] M. Zhao, H. Wu, B. Gao, Q. Zhang, W. Wu, S. Wang, Y. Xi, D. Wu, N. Deng, S. Yu, H.-Y. Chen, H. Qian, in *2017 IEEE Int. Electron Devices Meeting (IEDM)*, IEEE, Piscataway, NJ **2017**, pp. 39.4.1–39.4.4.
[26] Y.-H. Lin, C.-H. Wang, M.-H. Lee, D.-Y. Lee, Y.-Y. Lin, F.-M. Lee, H.-L. Lung, K.-C. Wang, T.-Y. Tseng, C.-Y. Lu, *IEEE Trans. Electron Dev.* **2019**, *66*, 1289.
[27] D. Ielmini, S. Lavizzari, D. Sharma, A. L. Lacaita, in *2007 IEEE Int. Electron Devices Meeting*, IEEE, Piscataway, NJ **2007**, pp. 939–942.
[28] W. W. Koelmans, A. Sebastian, V. P. Jonnalagadda, D. Krebs, L. Dellmann, E. Eleftheriou, *Nat. Commun.* **2015**, *6*, 8181.
[29] S. Ambrogio, M. Gallot, K. Spoon, H. Tsai, C. Mackin, M. Wesson, S. Kariyappa, P. Narayanan, C.-C. Liu, A. Kumar, A. Chen, G. W. Burr, in *2019 IEEE International Electron Devices Meeting (IEDM)*, IEEE, Piscataway, NJ **2019**, pp. 6–1.
[30] I. Muñoz-Martn, S. Bianchi, O. Melnic, A. G. Bonfanti, D. Ielmini, *IEEE Trans. Electron Dev.* **2021**, *68*, 6076.
[31] S. Blonkowski, *J. Appl. Phys.* **2010**, *107*, 084109.
[32] E. Esmanhotto, L. Brunet, N. Castellani, D. Bonnet, T. Dalgaty, L. Grenouillet, D. R. B. Ly, C. Cagli, C. Vizioz, N. Allouti, F. Laulagnet, O. Gully, N. Bernard-Henriques, M. Bocquet, G. Molas, P. Vivet, D. Querlioz, J. Portal, S. Mitra, F. Andrieu, C. Fenouillet-Beranger, E. Nowak, E. Vianello, in *2020 IEEE Int. Electron Devices Meeting (IEDM)*, IEEE, Piscataway, NJ **2020**, pp. 36.5.1–36.5.4.
[33] K.-H. Xue, B. Traoré, P. Blaise, L. R. C. Fonseca, E. Vianello, G. Molas, B. De Salvo, G. Ghibaudo, B. Magyari-Köpe, Y. Nishi, *IEEE Trans. Electron Dev.* **2014**, *61*, 1394.
[34] B. Traoré, P. Blaise, E. Vianello, L. Perniola, B. De Salvo, Y. Nishi, *IEEE Trans. Electron Dev.* **2016**, *63*, 360.
[35] G. B. Moody, W. E. Muldrow, *Comput. Cardiol.* **1984**, *101*, 215.
[36] A. Shafiee, A. Nag, N. Muralimanohar, J. P. Balasubramonian, R. Strachan, M. Hu, R. S. Williams, V. Srikumar, in *2016 ACM/IEEE 43rd Annual Int. Symp. on Computer Architecture*. IEEE, Piscataway, NJ **2016**.
[37] W. Zhao, C. Chappert, V. Javerliac, J.-P. Noziere, *IEEE Trans. Magn.* **2009**, *45*, 3784.
[38] T. Hirtzlin, M. Bocquet, B. Penkovsky, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, D. Querlioz, *Front. Neurosci.* **2020**, *13*, 1383.
[39] T. Hirtzlin, B. Penkovsky, M. Bocquet, J.-O. Klein, J.-M. Portal, D. Querlioz, *IEEE Access* **2019**, *7*, 76394.
[40] B. Liu, M. Hu, H. Li, Z.-H. Mao, Y. Chen, T. Huang, W. Zhang, in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, IEEE, Piscataway, NJ **2013**, pp. 1–6.

© 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

26404567, 0, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/aisy.202200145 by Cochrane France, Wiley Online Library on [19/11/2022]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License