TU Delft

Delft University of Technology

Document Version
Accepted author manuscript

Citation (APA)
Hamdioui, S., Du Nguyen, H. A., Taouil, M., Sebastian, A., Le Gallo, M., Pande, S., Schaafsma, S., Catthoor, F., Das, S., G. Redondo, F., Karunaratne, G., Rahimi, A., & Benini, L. (2019). Applications of Computation-In-Memory Architectures based on Memristive Devices. In *Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019: Proceedings* (pp. 486-491). Article 8715020 (Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019). IEEE.
https://doi.org/10.23919/DATE.2019.8715020

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

*This work is downloaded from Delft University of Technology.*

# Applications of Computation-In-Memory Architectures based on Memristive Devices *

Said Hamdioui[1] Abu Sebastian[2] Sandeep Pande[3] Shidhartha Das[4] G. Karunaratne[5]
Hoang Anh Du Nguyen[1] Manuel Le Gallo[2] Siebren Schaafsma[3] Fernando G. Redondo[4] Abbas Rahimi[5]
Mottaqiallah Taouil[1] Francky Catthoor[3*] Luca Benini[5]

[1]Computer Engineering, TU Delft, Delft, the Netherlands; S.Hamdioui@tudelft.nl
[2]IBM Research - Zurich, Switzerland; ASE@zurich.ibm.com
[3]IMEC, Eindhoven, Netherlands, [3*]IMEC., Leuven, Belgium; Francky.Catthoor@imec.be
[4]ARM Limited, Cambridge, UK; Shidhartha.Das@arm.com
[5]Integrated Systems Laboratory, ETH Zurich, Switzerland; lbenini@iis.ee.ethz.ch

December 11, 2018

## Abstract

Today's computing architectures and device technologies are unable to meet the increasingly stringent demands on energy and performance posed by emerging applications. Therefore, alternative computing architectures are being explored that leverage novel post-CMOS device technologies. One of these is a Computation-in-Memory architecture based on memristive devices. This paper describes the concept of such an architecture and shows different applications that could significantly benefit from it. For each application, the algorithm, the architecture, the primitive operations, and the potential benefits are presented. The applications cover the domains of data analytics, signal processing, and machine learning.

## 1 Introduction

Emerging applications are extremely demanding and have surpassed the capabilities of todays computational architectures and technologies [1,2]. Hence, in order for computing systems to continue delivering sustainable benefits for the foreseeable future, alternative computing architectures have to be explored. The emerging new device technologies could play a key role in this exploration. Computation-in-Memory (CIM) computing [3,4], brain-inspired neuromorphic computing [5] and quantum computing [6] are some of the most promising computational approaches being pursued, while memristive devices, quantum dots, spin-wave devices are some of the key emerging device technologies [7].

The EC H2020 MNEMOSENE project aims at demonstrating the *Computation-In-Memory (CIM)* concept based on memristive devices; it is based on integrating the processing units and the memory in the same physical location. As a consequence, it significantly reduces the memory accesses and data movements while supporting massive parallelism, resulting in potentially orders of magnitude improvement in terms of energy and computing efficiency. However, to achieve the ultimate objective of fully integrating the processing units and the memory in the same physical location, several technological challenges need to be overcome.

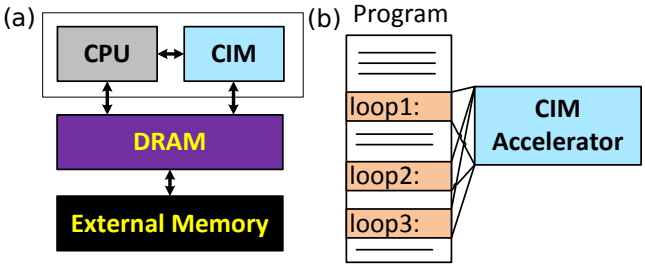



Figure 1: The CIM-based architecture

A realistic implementation which is well within the reach of today's technology is to use the CIM core as an on-chip *accelerator*. Figure 1(a) shows the concept; the CIM core may consist of very dense memristive crossbar array and CMOS peripheral circuitry responsible for the communication and control from/to the crossbar. In a conventional computer, the memory access part of the executed applications is dominating the energy consumption and the performance degradation. If we manage to get this part executed within the CIM core, then significant energy and performance improvement can be

*This research on CIM architecture is supported by EC Horizon 2020 Research and Innovation Program through MNEMOSENE project under Grant 780215.

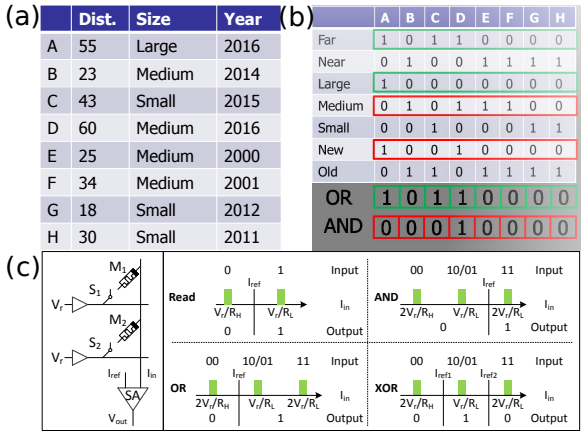



Figure 2: The database query problem and scouting logic

realized. Figure 1(b) illustrates a program that could be executed efficiently on this architecture; multiple loops can be executed within the CIM core while the other parts of the program can be executed on the conventional core. It is worth stressing that computations in CIM core takes place *within* the memory core consisting of a *memory array* and the *peripheral circuits*. Therefore, depending on *where* the result of the computation is produced, CIM core architecture can be divided into *CIM-Array (CIM-A)* [4,8] and *CIM-Periphery (CIM-P)* [9,10]. Even though both CIM-A and CIM-P could impact the design of the memory array, CIM-P entails a lesser impact on the design and hence is particularly attractive for a range of applications

This paper investigates three different application domains that could significantly benefit from the proposed architecture, and is organized as follows. Section II discusses the potential of accelerating two data analytic application kernels, *QUERY SELECT* for database and *XOR encryption* for security encryption. Section III and Section IV investigate the speed-up for two signal processing applications (compressed sensing and recovery and advanced image processing) and two machine learning applications (deep learning inference for IoT sensory applications, and brain-inspired hyperdimensional computing), respectively. Section V concludes the paper.

# 2 CIM for data analytics

One of the potential applications is big-data analytics with a high percentage of logical operations that perform poorly on conventional architectures due to e.g., high cache miss rates.

## 2.1 Targeted problem

We consider to speed up kernels (driven by bit-wise operation); examples are QUERY SELECT kernel (database applications) [11,12] and XOR encryption kernel (security encryption) [13].

- *QUERY SELECT kernel:* it performs the query-06 of the TPC-H benchmark [11], which includes 22 queries written in SQL language. The query-06 performs compare instructions to check if the requested data is available in the database or not.
- *XOR encryption kernel:* it performs an XOR operation of a string sequence and a predefined (secret) key. It is used for one-time-pad cryptography [14].

For QUERY SELECT kernel, we use a bitmap index scheme; it uses bitmaps (i.e., a vector of zeros and ones) to represent a database; generally they work well for low-cardinality columns. Figure 2(a) shows an example dataset with 8 entries, representing information of newly discovered stars. Each entry has three characteristics, i.e., distance (dist.), size and the year in which the star was discovered. Figure 2(b) presents the bitmap transposed representation of the same dataset, where the three characteristics (also called bins) are encoded into seven rows of zeros and ones; each column (e.g., A) is an entry while each row is a characteristic or bin. For example, a star with distance larger than 40 is defined as far, and otherwise as near. Typical database queries consist of searching for specific data patterns. These queries are carried out by performing bitwise operations on the bitmaps.

## 2.2 Implementation with CIM architecture

The implementation of CIM architecture considered for this application it similar to that shown in Figure 1; it consists of a conventional processor, main DRAM memory, novel data-centric CIM core and an external memory. Both the main memory and the CIM core can fetch data from the external memory. Like the main memory, CIM core is addressable from the processor and uses an extended address space. For simplicity, we assume that the data stored in the CIM core is not duplicated on the main DRAM memory; hence, simplified memory coherency schemes are required. The CIM core is initialized with data from the external memory, e.g., database(s); this initialization needs to be performed only once.

The architecture implementation considered here belongs to CIM-P; i.e., computing within CIM takes place within the peripheral circuitry. For the considered application, computing consists mainly of performing bit-wise operations including OR, AND, XOR gates.

Hence, the peripheral circuit should be modified. It is equipped with *Scouting Logic* [15] illustrated in Figure 2(c) using two binary valued memristive devices programmed to resistance values $R_1$ (for $M_1$) and $R_2$ (for $M_2$), respectively. Instead of reading a single memristive device at a time, two (or more) inputs are activated simultaneously (e.g., $M_1$ and $M_2$). The sensing current by the sense amplifier depends on the equivalent input resistance ($R_1//R_2$). By selecting appropriate reference currents $I_{ref}$, the gates OR, AND or XOR gates can be realized.

### 2.3 Analysis of the potential

To evaluate the the potential of considered architecture in terms of (normalized) delay and energy, we developed two analytical models similar to that in [16]; one for conventional architecture and one for CIM architecture. Using an analytical evaluation model makes it faster to perform a design space exploration, although it could be less accurate. It is worth noting that the model for the two bit-wise driven applications considered here (QUERY SELECT kernel and XOR encryption kernel) are similar; it is about the potential impact the CIM core on the overall performance rather than accurately quantifying the impact.

For the conventional architecture, we use the Intel Xeon E5-2680 multicore as a baseline with 4 cores, each with a frequency of 2.5GHz. Each core contains an ALU, and a two level cache (L1 of 32KB and L2 of 256KB). The cores share a main DRAM memory of 4GB. For the CIM architecture, we assume a single host processor with the same characteristics as an individual core in the conventional architecture. It contains an ALU, 32KB L1 cache and 256KB L2 cache, 1GB DRAM, and a CIM unit comprising 1,048,576 parallel memory arrays which has an area equivalent to that of 3GB DRAM. We assume that a logical instruction takes $\sim$ 10ns on CIM core which is equivalent to 20 CPU cycles [15,17].

We investigate the impact of the percentage of logical instructions accelerated by CIM core, as well as the impact of L1 and L2 cache miss rates on the potential improvement. Figure 3 shows the performance metric (defined by the normalized delay in seconds) for the conventional architecture (red planes) with respect to CIM architecture (green planes) for different percentages of accelerated instructions (X) on CIM core (ranging from 30 to 90%), assuming the problem size of $\sim$32 gigabyte (GB). It can be seen that the larger the size of the accelerated part on the CIM core, the higher the performance speed up; the speed up reaches up to 35x for the considered case. This can be clearly observed as the gap between the red and green planes increases. Moreover, the higher the miss rates, the higher the performance

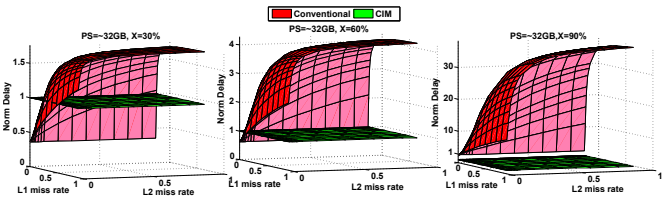



Figure 3: Analytical results of the performance (delay) metric

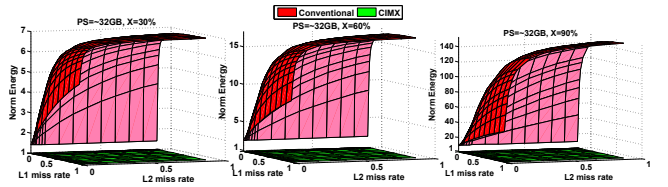



Figure 4: Analytical results of the energy metric

speed up of the CIM architecture. For low miss rates, the CIM could be even worse than conventional architecture especially when the percentage of accelerated instruction is low (e.g., 30% as Figure 3 shows).

Hence, the CIM architecture could be very suitable for applications with large data sizes and heavy memory access instruction (and bad data locality) resulting in a relatively high cache miss rate. Note that it has been shown that at least 30% of a database application could be accelerated using computation-in-memory [18].

Figure 4 shows the energy metrics (defined by the normalized energy in joule) for both architectures. Overall, similar trends are observed with respect to the percentage of accelerated instructions. However, the energy consumption of the CIM architecture is always lower, irrespective of the cache miss rates. In case 30% of the instructions are accelerated, the conventional architecture consumes 6x more energy for the same problem size. This grows up to two orders of magnitude in case 90% of the instructions are accelerated. The high energy consumption of the conventional architecture can be partly attributed to the data movement and leakage current.

## 3 CIM FOR SIGNAL PROCESSING

Next, we will investigate the advantages of a CIM architecture for applications such as advanced image processing and data compression. First, we will motivate an image processing application namely, guided image filter. Thereafter, we will present a detailed investigation of the application of compressed sensing and recovery.

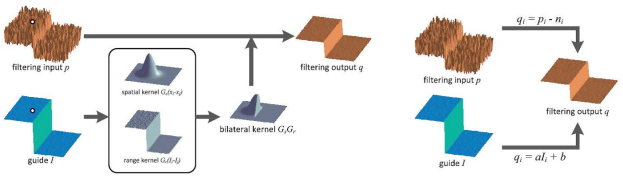



Figure 5: Bilateral Filtering and Guided Filtering Processes [19]

## 3.1 Image and video processing

The next generation of advanced image and video processing kernels often exhibit a mix of regular and irregular (or data-dependent) memory accesses. Moreover, they require data access which goes beyond the immediate local neighbours. Typically, they need a medium-size neighbourhood around the current pixel access. Typical values can be from $7 \times 7$ up to $11 \times 11$ pixels of 23 bits (in the case of colour images); and these do not directly fit in the local register-files, so they need to be accessed from SRAM caches or scratchpad memories. This limits the efficient mapping of these kernels on modern GPUs. The guided image filtering application [19] comprises a guidance image $I$, a filtering input image $p$, and an output image $q$. Both the guidance image $I$ and the input image $p$ act as input to the application, and as a special case, they can even be identical. Figure 5 illustrates the bilateral and guided filtering process. The guided image filtering problem is ideally suited to be implemented in a CIM-P architecture. The essential idea is store the data in a large non-volatile memristive array and enable irregular memory access by modifying the address decoder of the memory macro.

## 3.2 Compressed sensing and recovery

### 3.2.1 Targeted problem

Reconstruction of a sparse high-dimensional signal from low dimensional noisy measurements, for example received by sensor arrays, is used in many application fields, including radio interferometry for astronomical investigations, and magnetic resonance imaging, ultrasound imaging, and positron emission tomography for medical applications. Unfortunately, high-performance sparse signal recovery algorithms typically require a significant computational effort for the problem sizes occurring in most practical applications. While the computational complexity is not a major issue for applications where off-line processing on CPUs or graphics processing units can be afforded, it becomes extremely challenging for applications requiring real time processing at high throughput or for implementations on power-constrained devices.

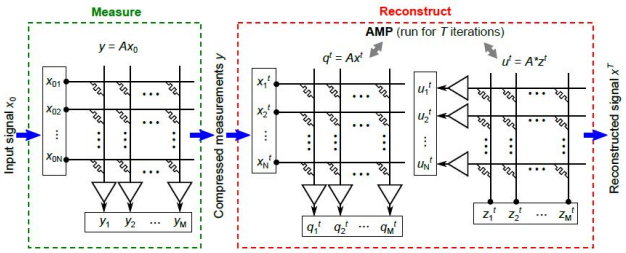



Figure 6: Proposed CIM implementation of compressed sensing with AMP recovery

In practically all the applications mentioned above, the observation model can be formulated as

$$y = Ax_0 + w$$

where A $\in \mathbb{R}^{M \times N}$ is a known measurement matrix, $x_0 \in \mathbb{R}^N$ is the signal of interest, y $\in \mathbb{R}^M$ is the measurement data vector and w $\in \mathbb{R}^M$ represents the measurement noise. The goal is to recover $x_0$ from y when M<N. A first order approximate message passing (AMP) technique for reconstructing $x_0$ given y [20] may be represented as

$$z^t = y - Ax^t + \frac{N}{M} z^{t-1} \langle \eta'_{t-1}(A^* z^{t-1} + x^{t-1}) \rangle$$

$$x^{t+1} = \eta_t(A^* z^t + x^t)$$

where $x^t \in \mathbb{R}^N$ is the current estimate of $x_0$ at iteration t, $z^t \in \mathbb{R}^M$ is the current residual, $A^*$ is the transpose of A, $\eta_t(\cdot)$ is a function, $\eta'_t(\cdot)$ its derivative, $\langle \cdot \rangle$ denotes the mean and $x^0 = 0$. The final value of $x^t$ provides the estimate of $x_0$. The AMP algorithm has a relatively simple formulation and requires only multiplications and additions, making it suitable for a memristive CIM architecture.

### 3.2.2 Implementation with CIM architecture

A CIM architecture comprising CIM-P-type units that can store the measurement matrix $A$ and perform the matrix-vector multiplications within the array would significantly increase the area/time/power efficiency. The elements of $A$ are mapped as conductance values of memristive devices organized in a crossbar array, as depicted in Figure 6 [21]. One possible method to program the conductance values is by an iterative program-and-verify procedure. The compressed measurements $y$ are acquired by applying $x_0$ as voltages to the crossbar rows via digital-to-analog conversion, and obtaining $y$ through analog-to-digital conversion of the resulting output currents at columns. The positive and negative elements of A can be coded on separate devices together with a subtraction circuit, whereas negative vector elements can be applied as negative voltages.

Once the matrix $A$ is programmed in the crossbar array and the measurements $y$ are obtained, the AMP algorithm is run in a dedicated processing unit, while the computation of $q^t{=}Ax^t$ and $u^t{=}A^*z^t$ is performed using the (same) crossbar array. The vector $q^t$ is computed by applying $x^t$ as voltages to the rows and reading back the resulting currents on the columns, and $u^t$ by applying $z^t$ as voltages to the columns and reading back the resulting currents on the rows.

In the AMP algorithm, ignoring the $\eta_t(\cdot)$ and $\eta'_t(\cdot)$ functions, the main computational cost comes from the matrix-vector multiplications $Ax^t$ and $A^*z^t$ which both require O($MN$) operations for dense $A$. The other operations in the AMP algorithm are vector additions and multiplications which require O($N$) operations. Thus, one could potentially reduce the complexity of AMP from O($MN$) to O($N$) by performing $Ax^t$ and $A^*z^t$ in memristive arrays, assuming that $\eta_t(\cdot)$ and $\eta'_t(\cdot)$ involve only O($N$) or less operations. The expectation is that in a memristive crossbar, matrix-vector multiplications can be performed with constant time complexity O($\gamma$), where $\gamma$ is independent of the crossbar size.

#### 3.2.3 Analysis of the potential

To quantify the potential energy gains of the CIM implementation over a conventional design, based on the figures currently achieved with a prototype phase-change memory (PCM) chip [22], we made an FPGA design that operates at the same speed and the same precision at which we expect a PCM-based crossbar to perform. In the AMP algorithm, the matrix-vector multiplications are the most expensive operations, so we compared the memristive crossbar analog multiplier with a 4-bit FPGA multiplier design. We focus in this analysis on the energy drawn by the computational units and disregard the time and power consumption of the data transfers.

The time to compute one dot-product is equal to the vector size divided by 8, plus 5 cycles to complete the pipeline. For a $1024 \times 1024$ matrix-vector product using the 1024-unit design, each dot-product unit stores one of the matrix row of 1024 elements encoded with 4-bit per value in the local 32Kbit BlockRAM. To read the row vector from memory and to perform the dot-product operation takes a total of 133 clock cycles. Hence, it takes 665 ns to complete one matrix-vector multiplication at a clock frequency of 200 MHz. Considering a dynamic power consumption of 26.6W, one matrix-vector multiplication consumes 17.7$\mu$J on the FPGA.

In a memristive crossbar of size $1024 \times 1024$ based on PCM devices, the dynamic power dissipation in the devices for one READ operation is expected to be on the order of 0.21W, assuming an average READ current of 1$\mu$A per device and average voltage of 0.2V. In order to operate this crossbar at 1$\mu$s cycle time, 8 analog-to-digital converters (ADCs) operating at 125MSps are needed to read the currents from all 1024 columns in approximately 1$\mu$s. The power consumption of 8-bit ADCs in 90nm technology is estimated to be around 12 mW/GSps, thus 12.3mW for 1024 reads per microsecond. Therefore, the total power consumption of the crossbar and ADCs is estimated to be around 222mW, which is 120 times lower than the 4-bit FPGA design. The energy per READ is 222nJ, which is 80 times lower than the FPGA. Assuming 90nm technology and 25F$^2$ 1T1R PCM cells (F = 90nm), the area occupied by a 1024x1024 crossbar and 8 ADCs (each of size 50$\mu$mx300$\mu$m) would be on the order of 0.332mm$^2$.

Table 1: FPGA resource utilization, frequency and estimated dynamic on-chip power consumption

| LUT | FF | BRAM | f[MHZ] | $P_{static}$[W] | $P_{dynamic}$[W] |
|---|---|---|---|---|---|
| 307908 | 180368 | 1024 | 200 | 4.04 | 26.4 |
| [46.4%] | [13.6%] | [47.4%] | (utilization on the xckul 15 FPGA device) | | |

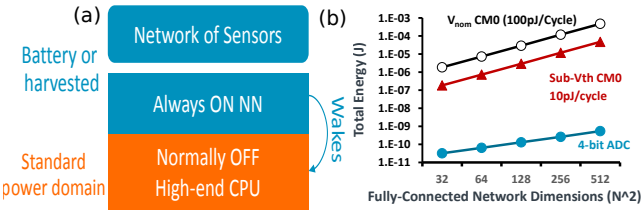



Figure 7: Inference on IoT sensory devices

## 4 CIM FOR MACHINE LEARNING

In this section, we will investigate the application domain of machine learning in particular applications where the training or inference has to be performed in highly energy/area constrained environments. First, we will present the application domain of deep learning for internet-of-things (IoT) and subsequently, we will present the emerging machine learning paradigm of hyperdimensional computing.

### 4.1 Deep learning inference for IoT sensory applications

#### 4.1.1 Targeted problem

Computing systems with CIM architectures could play a key role in the Internet of things (IoT) sensory domain. When deployed in edge-devices, always ON deep learning inference applications require minimum power consumption, and therefore, CIM architectures particularly suit these hard requirements. Examples include Human Activity Recognition (HAR), Key Word Spotting (KWS) and online Electro-cardiograph (ECG)

event detection and classification. As shown in Figure 7(a), the always ON CIM architectures can process the data coming from a network of sensors in an efficient manner, and either work as the main computing element on the IoT device, or on the other hand, sparsely wake up a higher-end CPU should a specific condition be met.

#### 4.1.2 Implementation with CIM architecture

Similar to the compressed sensing application, the computational primitive is matrix-vector multiplication using a memristive crossbar array. Deep neural networks are just a cascade of matrix-vector multiply units and activation functions. The multiple layers of a standard fully connected neural network (FCNN) or convolutional neural network (CNN) can be mapped to CIM cores comprising memristive crossbar arrays. Even though the matrix-vector multiplications are performed in the analog domain using Ohms law and Kirchhoffs current summation law, DACs are used to input the data to each crossbar array and ADCs are used to digitize the resulting current. A key challenge is the lack of precision associated with the analog multiplication as well as the quantization of the input and activations as dictated by the DAC/ADC resolution. However, it has recently been demonstrated that it is possible to perform deep learning inference with limited precision. It is shown that one can achieve comparable classification accuracy as networks operating with floating point precision [23].

#### 4.1.3 Analysis of the potential

Preliminary comparative study of implementations of the DL algorithms was conducted. First, we analyzed the effects that low precision layers have on the overall NN accuracy, determining the quantization characteristics of the different layers. Second, the CIM approach was compared with implementations using low-power near threshold Cortex-M processors [24]. The study shows the significant potential for energy gains with the use of a low precision CIM architecture (see Figure 7(b)).

### 4.2 Brain-inspired hyper-dimensional computing

#### 4.2.1 Targeted problem

We present another application space for CIM namely hyperdimensional (HD) computing suitable for various learning and classification tasks using memristive devices [25]. HD computing is a brain-inspired computing paradigm where information is represented in hypervectors: $d$-dimensional holographic (pseudo)random vectors with independent and identically distributed (i.i.d.) components. When the dimensionality is in the thousands, e.g. $d > 1000$, there exist a very large number of quasiorthogonal hypervectors. This lets HD computing combine such hypervectors into a new hypervector using well-defined vector space operations. These mathematical operations are bitwise and ensure that the resulting hypervector has the same dimensionality—i.e., fixed-width. The resulting hypervectors can then be directly used to not only classify but also to bind, associate, and perform other types of "cognitive" operations in a straightforward manner.

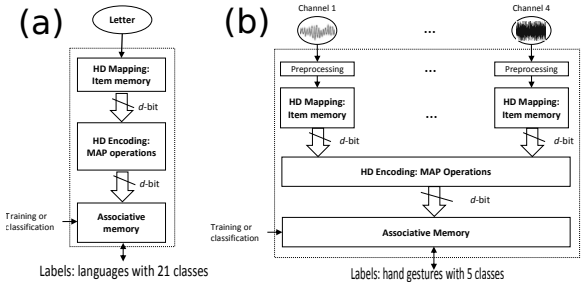



Figure 8: General and scalable HD computing for various learning and classification tasks

HD computing uses three operations to combine binary hypervectors: addition (which can be weighted), multiplication, and permutation (more generally, multiplication by a matrix) that are collectively called as MAP operations. "Addition" and "multiplication" are meant in the abstract algebra sense where the sum of binary hypervectors $[A+B+\dots]$ is defined as the componentwise majority function with ties broken at random, the product is defined as the componentwise XOR (addition modulo 2, $\oplus$), and permutation ($\rho$) shuffles the components. All these MAP operations produce a $d$-bit hypervector.

HD computing has been used in various applications such as language recognition [26] (Figure 8(a)) and biosignal processing (Figure 8(b)) including electromyography (EMG) [27], electroencephalography (EEG) [28], and electrocorticography (ECoG) [29] with up to 100 electrodes. These learning and classification tasks are based on the same hardware construct: 1) mapping to the HD space, 2) encoding with the MAP operations, and 3) associative memory (see Figure 8). During training, the associative memory updates the learned patterns with new hypervectors, while during classification it computes distances between a query hypervector and learned patterns. Hence, it is possible to build a CIM engine based on these operations to cover a variety of tasks.

#### 4.2.2 Implementation with CIM architecture

The CIM primitives used for HD computing implementation are dot-product and bitwise operations. The dot-product is performed using binary input values, binary memristor states, and analog output. The bitwise operations are performed using binary input values, binary memristor states, and binary output. The memristor values are written only once before the execution of the HD algorithm and are never modified again. Additional digital computations and memory buffers are needed in order to implement the entire HD algorithm.

#### 4.2.3 Analysis of the potential

Simulation studies were conducted using a CIM unit based on realistic models of phase-change memory devices. It was shown the CIM architecture can deliver comparable accuracies to the ideal software simulations for the task of language recognition. Preliminary results were also obtained comparing the energy efficiency of a potential CIM-based implementation over 65nm digital CMOS implementation. A cycle-accurate RTL model that has equivalent throughput to that of the proposed CIM HD processor was developed. The RTL model was synthesized in UMC 65nm technology node using *Synopsys Design Compiler*. Energy estimation was carried out in *Synopsys PrimeTime* by providing the netlist and the activity file as the inputs. A best area improvement of 9× and an energy improvement of 5× is expected with the CIM HD processor architecture compared to CMOS counterpart. By utilizing more efficient ADCs the performance numbers could be improved further. Nevertheless if only the replaceable module in the architecture are considered vast improvements can be expected which are eclipsed by the current energy budget of the non-replaceable modules. When only replaceable modules are considered, energy efficiency can be two to three orders of magnitude higher in the case of a CIM architecture.

## 5 Conclusion

Computation-in-memory using memristive devices is an emerging computing paradigm that tries to address the challenge of memory wall posed by the conventional von Neumann architecture. Although the extent of improvement in terms of energy/time efficiency is application and problem-size dependent, the CIM architecture clearly has the potential to outperform the traditional von Neumann architecture due to many reasons. For instance, it uses non-volatile memristive technology which reduces the static power. In addition, it performs computation within the memory core, meaning that data movement is significantly reduced; this results both in energy saving and performance improvement. Moreover, given the nature of the CIM core, the time complexity of some primitive function such as matrix-vector multiplication could be reduced from $O(N^2)$ to $O(1)$, resulting in further performance improvement. In this paper, we presented concrete examples from the domains of data analytics, signal processing and machine learning that could significantly benefit from this new architecture. We presented how a CIM architecture could tackle these problems and in many instances presented a detailed study on the potential area/energy/time benefits.

## References


[1] D. A. Patterson, "Future of computer architecture," in *Berkeley EECS Annual Research Symposium (BEARS), US*, 2006.

[2] H. Jones, "Whitepaper: semiconductor industry from 2015 to 2025," *International Business Strategies*, 2015.

[3] S. Hamdioui, L. Xie, H. A. D. Nguyen, M. Taouil, K. Bertels, H. Corporaal, H. Jiao, F. Catthoor, D. Wouters, L. Eike *et al.*, "Memristor based computation-in-memory architecture for data-intensive applications," in *DATE'15*, 2015, pp. 1718–1725.

[4] A. Sebastian, T. Tuma, N. Papandreou, M. Le Gallo, L. Kull, T. Parnell, and E. Eleftheriou, "Temporal correlation detection using computational phase-change memory," *Nature Communications*, vol. 8, no. 1, p. 1115, 2017.

[5] A. Sebastian, M. Le Gallo, G. W. Burr, S. Kim, M. BrightSky, and E. Eleftheriou, "Tutorial: Brain-inspired computing using phase-change memory devices," *Journal of Applied Physics*, vol. 124, no. 11, p. 111101, 2018.

[6] "https://www.research.ibm.com/ibm-q/."

[7] ITRS, "Beyond cmos white paper," ITRS, Tech. Rep., 2014.

[8] P. Hosseini, A. Sebastian, N. Papandreou, C. D. Wright, and H. Bhaskaran, "Accumulation-based computing using phase-change memories with fet access devices," *IEEE Electron Device Letters*, vol. 36, no. 9, pp. 975–977, 2015.

[9] H. A. Du Nguyen, J. Yu, L. Xie, M. Taouil, D. Fey, and H. Said, "Memristive devices for computing: Beyond cmos and beyond von neumann," in *VLSI-SoC'17*. IEEE, 2017.

[10] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *DAC'16*. IEEE, 2016.

[11] T. P. P. Council, "Tpc-h, a decision support benchmark," 2015.

[12] K. Wu, W. Koegler, J. Chen, and A. Shoshani, "Using bitmap index for interactive exploration of large datasets," in *SSDBM'03*. IEEE, 2003, pp. 65–74.

[13] J. Yang, L. Gao, and Y. Zhang, "Improving memory encryption performance in secure processors," *IEEE Trans. on Comp.*, vol. 54, no. 5, pp. 630–640, 2005.

[14] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

[15] L. Xie, H. Du Nguyen, J. Yu, A. Kaichouhi, M. Taouil, M. AlFailakawi, and S. Hamdioui, "Scouting logic: A novel memristor-based logic design for resistive computing," in *ISVLSI'17*. IEEE, 2017, pp. 176–181.

[16] H. A. Du Nguyen, L. Xie, M. Taouil, R. Nane, S. Hamdioui, and K. Bertels, "On the implementation of computation-in-memory parallel adder," *IEEE TVLSI*, 2017.

[17] P.-Y. C. PS.Yu, "merging memory technologies," *SPRING 2016 Solid-sate circuits magazine*, vol. 8, no. 2, pp. 43–56, 2016.

[18] V. Seshadri, K. Hsieh, A. Boroum, D. Lee, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Fast bulk bitwise and and or in dram," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 127–131, 2015.

[19] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE transactions on pattern analysis & machine intelligence*, no. 6, pp. 1397–1409, 2013.

[20] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18 914–18 919, 2009.

[21] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, "Compressed sensing with approximate message passing using in-memory computing," *IEEE Transactions on Electron Devices*, vol. 65, no. 10, pp. 4304–4312, 2018.

[22] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou, "Mixed-precision in-memory computing," *Nature Electronics*, vol. 1, no. 4, p. 246, 2018.

[23] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights," pp. 1–14, Feb 2017. [Online]. Available: http://arxiv.org/abs/1702.03044

[24] J. Myers, A. Savanth, P. Prabhat, S. Yang, R. Gaddh, S. O. Toh, and D. Flynn, "A 12.4pJ/cycle sub-threshold, 16pJ/cycle near-threshold ARM Cortex-M0+ MCU with autonomous SRPG/DVFS and temperature tracking clocks," in *Symposium on VLSI Circuits*. IEEE, jun 2017, pp. C332–C333.

[25] A. Rahimi, S. Datta, D. Kleyko, E. P. Frady, B. Olshausen, P. Kanerva, and J. M. Rabaey, "High-dimensional computing as a nanoscalable paradigm," *IEEE TCAS I*, vol. 64, no. 9, pp. 2508–2521, Sept 2017.

[26] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A robust and energy efficient classifier using brain-inspired hyperdimensional computing," in *Symposium on Low Power Electronics and Design*, August 2016.

[27] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey, "Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition," in *IEEE International Conference on Rebooting Computing*, October 2016.

[28] A. Rahimi, P. Kanerva, J. del R Millán, and J. M. Rabaey, "Hyperdimensional computing for noninvasive brain–computer interfaces: Blind and one-shot classification of EEG error-related potentials," *BICT'17*, 2017.

[29] A. Burrello, K. Schindler, L. Benini, and A. Rahimi, "One-shot learning for iEEG seizure detection using end-to-end binary operations: Local binary patterns with hyperdimensional computing," in *BioCAS'18*, 2018, pp. 1–4.