Memristor Based Computation-in-Memory Architecture for Data-Intensive Applications

Abstract—One of the most critical challenges for today’s and future data-intensive and big-data problems is data storage and analysis. This paper first highlights some challenges of the new born Big Data paradigm and shows that the increase of the data size has already surpassed the capabilities of today’s computation architectures suffering from the limited bandwidth, programmability overhead, energy inefficiency, and limited scalability. Thereafter, the paper introduces a new memristor-based architecture for data-intensive applications. The potential of such an architecture in solving data-intensive problems is illustrated by showing its capability to increase the computation efficiency, solving the communication bottleneck, reducing the leakage currents, etc. Finally, the paper discusses why memristor technology is very suitable for the realization of such an architecture; using memristors to implement dual functions (storage and logic) is illustrated.

I. INTRODUCTION

Today’s applications are becoming extremely data intensive; healthcare, social media, large scientific/engineering experiments, and security are just couple of examples. As the speed of information growth exceeds Moore’s Law, since the beginning of this new century, excessive data is posing major challenges [1] and a new scientific paradigm is born: data-intensive scientific discovery, also known as Big Data problems. The primary goal is to analyse and increase the understanding of both data and processes in order to extract so much potential and highly useful information hidden in the huge volume of data, and hence, dealing with Big Data problems. Big Data is extremely valuable to generate productivity in businesses and evolutionary breakthroughs in scientific disciplines, which give us a lot of opportunities to make great progress in many fields [1]. The primary goal is to increase the understanding of processes in order to extract so much potential and highly useful values hidden in the huge volumes of data, and therefore, it comes with many challenges, such as data capture, data storage, data analysis, and data visualization.

This paper discusses a new architecture, Computation-In-Memory (CIM Architecture), for specific data-intensive applications; it is based on the integration of storage and computation in the same physical location (crossbar topology) and the use of non-volatile resistive-switching technology (memristive devices or memristors in short) [30, 38, 39, 94] instead of CMOS technology.

The rest of the paper is organized as follows. Section II highlights the Big Data problem and shows how the conventional computers based on CMOS technology are incapable to deal with such problems; and motivates the need for a new architecture. Section III discusses CIM architecture, including its concept and its potential; the section puts that in perspective by taking couple of application examples and comparing the performance of CIM architecture with the state-of-the-art. Section III shows why memristor is the key enabler for CIM architecture by illustrating how the device, in crossbar architecture, can perform a dual function (storage and computation). Section IV concludes the paper.

II. DATA-INTENSIVE APPLICATIONS VS CMOS COMPUTERS

A. Big Data and Data Intensive Applications

No one can deny the fact that a large number of fields and sectors, ranging from economics and business activities to public administration, from national security to many scientific research areas, involve data-intensive applications; it is based on the state-of-the-art. Section III shows why memristor is the key enabler for CIM architecture by illustrating how the device, in crossbar architecture, can perform a dual function (storage and computation). Section IV concludes the paper.
Performing data analysis within economically affordable time and energy is the pillar to solve big data problems.

**B. Today’s Computers**

The computing systems, developed since the introduction of stored program computers by John von Neumann in the forties [5], can be classified based on the location of the so-called “working set” (loosely defined as the collection of information referenced by a program during its execution) into four classes (a) to (d) as shown in Figure 1. In the early computers (typically before the 80s), the working set was contained in main memory. Due to the gap between the core (CPU) speed and the memory, caches were introduced to reduce the gap and increase the overall performance, where the caches have become the location of the working set. Today’s computing systems for data-intensive applications are still based on Von Neumann (VN) architectures and still rely on many parallel (mini-)cores with a shared SRAM cache (parallel CPUs, GPUs, SIMD-VLIWs, vector processors); see Figure 1(c). Clusters of cores can be replicated many times, each having their own L1 cache, but it is far from realistic to assume a distributed reasonable sized L1 cache in every mini-core; too much area and leakage power overhead is incurred in that case. Such solutions suffer from major limitations such as a decreased performance acceleration per core [6], increased power consumption [7, 8], and limited system scalability [6, 9]. These are mainly caused by the processor-memory bottleneck [10, 11]. As current data-intensive applications require huge data transfers back and forth between processors and memories through load/store instructions [12], the maximal performance cannot be extracted as the processors will have many idle moments while waiting for data [10-14]. Computation, which is the main activity of a system, by far consumes less energy and chip area, and has lower execution time compared to communication and memory access (e.g., L1 cache), especially for data intensive applications [15]. The energy consumption of the cache accesses and communication makes up easily 70% to 90% [2,3,4]; not to mention the rest of the memory hierarchy. In addition, programmability in conventional processors also comes at a substantial energy cost: for example, [4] reports that executing a multiply instruction on a simple in-order core in 45nm technology consumes about 70 pJ, whereas the actual operation itself consumes less than 4 pJ. The overhead is due to instruction fetching and decoding and other control.

Triggered by these issues, the design of high-performance computing systems is starting to move away from a conventional computation-centric model towards a more data-centric approach. The latter concept intends to improve performance and power efficiency through reduction of data movement by performing the actual processing closer to where the data resides in the memory system. Several alternative architectures are proposed that fall into this category. One alternative is called “Processor-in-memory” as shown in Figure 1(d); additional processing units (accelerators) are put around one or more memories which are the working set location; examples are FlexRAM [16], DIVA [17], TeraSys [18], EXECUBE [19], HTMT [20], Computational RAM [21], DSP-RAM [22], Smart memories-based architecture [23], Gilgamesh [24], Continuum computer architecture [24], and MICRON’s architecture for automata processing [26]. The second alternative architecture is called “Memory-in-processor”, which is an extension of what is shown in Figure 1(c), where extra addressable memories are put close to the cores; examples are Data Arithmetic SRAM [27] and Connection machine [28]. The third is called “In memory computing/database” (mainly for database management), which primarily relies on the storage of the complete database working set in the main memory of dedicated servers rather than relying on complicated relational databases operating on comparatively slow disk drives [29].

**C. CMOS Technology**

Today’s computers are manufactured using the traditional CMOS technology, which is reaching the inherent physical limits due to down-scaling. Technology nodes far below 20nm are presumably only practical for limited applications due to multiple challenges [30-34], such as high static power consumption, reduced performance gain, reduced reliability, complex manufacturing process leading to low yield and complex testing process, and extremely costly masks. Many novel nano-devices and materials are under investigation to replace the CMOS technology in next IC generations. Among the emerging devices, such as graphene transistor [35], nanotube [36], tunnel field-effect transistor (TFET) [37], etc., memristor [38, 39] is a promising candidate.
Its advantages are CMOS process compatibility [40], lower cost, zero standby power [41], nanosecond switching speed [42], great scalability and high density [43], and non-volatile nature [44, 45]. It offers a high OFF/ON resistance ratio [46] and it is promising to have a good endurance and retention time [47]. More importantly, the memristor is a two-terminal resistive-switching device that can be used to build both storage and information processing units [48, 49, 50].

D. The Need of New Architecture

The speed at which data is growing has already surpassed the capabilities of today’s computation architectures suffering from communication bottleneck (due to limited bandwidth), energy inefficiency (due to CMOS technology), and programmability overhead. Therefore, there is a need for a new architecture using new device technology, being able to (a) eliminate the communication bottleneck and support massive parallelism to increase the overall performance, (b) reduce the energy inefficiency to improve the computation efficiency. This can be done by taking the data-centric computing concept much further by integrating the processing units and the memory in the same physical location and therefore moving the working set into the core as shown in Figure 1 (e).

III. CIM ARCHITECTURE- BEYOND VON NEUMANN

This section presents first the concept of the CIM architecture as an alternative of today’s architectures. Thereafter the potential of the architecture will be illustrated by selecting couple of data-intensive applications and making an analysis of different performance metrics and comparing the results with the state-of-the art. Finally, major open questions related to the implementation of the CIM architectures will be highlighted.

A. CIM Architecture Concept

To tackle the big data computation problems and solve the todays computers bottlenecks, we propose a memristor-based architecture paradigm where both the computation and the storage take place at the same physical location (the crossbar array). The approach intends to provide solutions based computing-in-memory architectures using non-volatile devices. Figure 2 shows the traditional versus the proposed CIM architecture; note that in CIM architecture the storage and computation are integrated together in a very dense crossbar array where memristors are injected at each junction of the crossbar (top electrode and bottom electrode). The communication and control from/to the crossbar can be realized using CMOS technology. CIM architecture addresses important challenges and has huge potentials which go substantially beyond the current state-of-the-art.

- **Tightly integrated computation-in-memory crossbar architecture supporting massive parallelism:** as the storage and computation are integrated together, the communication bottleneck is significantly reduced. In addition, because the memristor technology is highly scalable (~5nm [30]), huge crossbar architectures allowing massive parallelism are feasible.

B. CIM Architecture Potential

To illustrate how CIM architecture significantly advances the state-of-the-art, the performance of CIM and the conventional architectures for two applications will be estimated.

1) **Healthcare:** using genomics in diagnosing/treating diseases: the continuously dropping price of DNA sequencing has shifted the challenge from acquiring genetic information to the actual processing and analysis of this information [51]. Despite its computational simplicity, the huge amount of genetic data (hundreds of GBs per experiment) that needs to be processed makes the analysis rather time consuming and even not practical due to communication/memory access bottleneck. A practical solution used today for comparing two DNA sequences is based on the creation of a sorted index of the reference DNA that can be used to identify the location of matches and mismatches in another sequence rapidly. This approach, however, results in eliminating available data locality in the reference and causing huge number of cache misses with high memory access penalty and high energy cost. To quantify this effect, we assume we have 200 GB of DNA data to be compared to a healthy reference of 3GB for 50% coverage [51] and evaluate the DNA sorted-index sequencing algorithm on both the conventional architecture and CIM architecture; see Figure 2. The conventional architecture is assumed to scalable multi-core architecture, consisting of a number of clusters, each with 32 cores. Table 1 presents further made assumptions for both architectures. Three metrics are used for the evaluation: (a) the energy-delay product per operations, (b) the computation efficiency defined as the number (#) of operations per required energy, and (c) performance (#operations) per area;

![Figure 2: Traditional versus proposed architecture](image-url)
2) Mathematics: Here we assume 10^6 parallel addition operations and make similar assumptions for the two architectures as those done in the previous example; see Table 1 for the details.

Table 2 shows the obtained results for the two applications for conventional (Conv.) and CIM architectures; both applications clearly show that the improvements are orders of magnitude. Reducing/eliminating memory accesses, the non-volatile properties of the materials and assuring reliability, the high parallelism are enabling these improvements.

C. CIM Architecture Challenges

Although we mentioned that CIM architecture targets data-intensive applications, especially applications that require massive parallelism and huge data working sets to be continuously kept in the memory, the proposed concepts can be adapted to any computation-in-memory (CIM) architecture for high computation efficiency. This architecture paradigm shift, based on memristor technology, changes the traditional system design, compiler tools, manufacturing processes, etc., to facilitate its “industrialization”. In fact, it is well recognized that memristor technologies are very promising. Although understanding of its capabilities and limitations is still evolving, the technology is expected to rule the computer world, from material science (understanding and proving the properties of the materials and assuring reliability), to design methods, tools, operating systems and its potential applications. Exascale

<table>
<thead>
<tr>
<th>Metric</th>
<th>Archit. DNA Sequencing</th>
<th>10^6 additions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy-delay/operations</td>
<td>Conv. 2.0210e-06</td>
<td>1.5043e-18</td>
</tr>
<tr>
<td>Computing efficiency</td>
<td>CIM 2.3622e-09</td>
<td>9.2570e-21</td>
</tr>
<tr>
<td>Performance area</td>
<td>Conv. 5.7312e+09</td>
<td>5.1118e+09</td>
</tr>
<tr>
<td></td>
<td>CIM 5.1118e+09</td>
<td>4.9164e+12</td>
</tr>
</tbody>
</table>
computing, ‘computer on a chip’ capabilities, as well as driving developments in neural and analogue computing. Next section will elaborate more on memristor technology.

IV. MEMRISTOR - THE KEY ENABLER FOR CIM ARCHITECTURE IMPLEMENTATION

This section reviews first the memristor technology. Thereafter its suitability for the realization of both storage and logic functions is discussed and illustrated.

A. Memristor Technology

Memristors or memristive devices, also referred to as resistive memory devices, are very broad groups of memory technologies; they can be classified based on their dominant physical operating mechanism into three classes [30]: Phase Change Memories, Electrostatic/Electronic Effects Memories, and Redox memories. The redox-based resistive switching devices (ReRAMs) are attracting most attention due to their excellent scaling, endurance, and retention properties [30, 95]; their physical mechanism for switching is based on reduction/oxidation (Redox)-related chemical effects. The category of “Redox RAM” encompasses a wide variety of Metal-Insulator-Metal (MIM) structures; the electrochemical mechanisms driving the resistance state (from high to low or vice versa) can operate in the bulk I-layer, along conducting filaments in the I-layer, and/or at the I-layer/metal contact interfaces in the MIM structure. The ReRAMs consist of three types, two bipolar and one unipolar [30, 50, 61]; the rest of the section will focus on the two bipolar devices and show the best available properties for both device types; these are the Valence Change Memory (VCM) and the Electrochemical metallization (ECM) devices.

For both VCM (HfO₂) and ECM (Ag-chalcogenide) devices a feature size of $F = 10 \text{ nm}$ was reported [62, 63]. A minimum switching time of $< 200 \text{ ps}$ was shown for TaOₓ-based VCM devices [42], whereas for ECM devices (Ag-MSQ) switching times below 10 ns were realized [64]. In terms of endurance, more than $10^{12}$ cycles are feasible for TaOₓ-based VCM cells and more than $10^{10}$ for Ag-GeSe ECM cells [65]. Extrapolated retention of $> 10$ years was, for example, shown in [66] (TaOₓ-based VCM cells) and [67] (Ag-chalcogenide).

In ECM devices a conductive metallic filament (Cu or Ag) is established during switching, thus, the filament length can be considered the state variable [68]. For a memristive ECM model, both electronic and ionic currents must be considered, and the strong non-linearity of the switching kinetics must be reflected by the model. VCM modelling is even more challenging due to the versatile device physics [69]. Since simple memristor models fail to predict the correct device behaviour [39, 70], more complex empirical and physics-based models were developed recently [71, 72].

B. Memristor for Crossbar Memories

The primary driver for ReRAM research is the semiconductor industry seeking for novel energy-efficient non-volatile and highly scalable memory elements [30]. A straightforward implementation of the ReRAM array is realised using a passive crossbar architecture, resulting in the highest density [73, 74]. However, this architecture suffers from undesired paths for current called sneak paths [75]; due to the low resistive current paths, the maximum array is limited to small arrays [76]. To overcome this issue, three classes of solutions are proposed:

- **Selector devices**, which are separate devices in connection with the RRAM cell such as a diode or a transistor (1S1R) [77, 78].

- **Switching device modification**, where the resistive devices is modified; E.g., serially connecting of two anti-serial memristive devices (bipolar switches) resulting into a “complementary resistive switcher” (CRS) being able to block the current at low voltage irrespective of the state of the device [78], or the deployment of a high nonlinear memristive device (due to current-controlled negative differential resistance) to overcome sneak path [79].

- **Bias schemes**, where the voltage bias applied to non-accessed wordlines and bitlines are set to values different from those applied to accessed wordline and bitlines in order to minimize the sneak path current; examples are multistage reading [80] and use of AC signal instead of DC for sensing the data stored in the desired cell [81].

Figure 3 sketches the concept of the the crossbar array and some junction options to deal with sneak paths, while Figure 4 illustrates the $I$-$V$ characteristic of a CRS cell which consists of two memristive ECM devices A and B. The states '0' and '1' are the logical storage states and the state 'LRS/LRS' occurs only when reading the memory state. The internal memory states '0' and '1' of a CRS cell are indistinguishable at low voltages because state '0' as well as state '1' show a high resistance. Therefore, no parasitic current sneak paths can arise. To read the stored information of a single CRS cell, a read voltage must be applied to the cell. If the CRS cell is in state '0', then it switches to state 'ON'; if the cell is in state '1' then it remains in its state. In case conventional crossbar (with resistive current paths), reading ON state is a destructive operation, therefore, it is necessary to write back the previous state of the cell after reading it. In general, the writing of state '0' requires a negative voltage ($V < V_{th,4}$) and for writing '1' a positive voltage $V > V_{th,2}$ is required.
C. Memristor for Logic Functions

Memristive devices are well suited for the implementation of logic functions including: (a) programmable interconnects [82], (b) look-up tables (LUTs) [83] or content addressable memories (CAMs) [84], and (c) sequential 'stateful' logic operations [49, 58, 85].

Programmable logic arrays based on resistive switching junctions were suggested first in [82] and later also applied to FPGAs [86]. Typically, the CMOS overhead is relatively large since the array size is small. A next step was the CMOL FPGA concept [87], where a sea of elementary CMOS cells is connected to a small crossbar part-array. In this approach the elementary CMOS cells are connected via resistive switches (1S1R) enabling wired-or functionality. In general, reconfigurable on-chip wiring enables new options for memristive chip design and can also be combined with the functionalities as those described above.

Resistive memories can be either used to implement small LUTs for FPGAs (as suggested in [83]) or LUTs can be mapped to large-scale crossbar arrays [88, 89] to reduce the crossbar array overhead. Moreover, CAMs based on memristors are feasible with different flavors [90, 91]; e.g., a CRS-based CAM is recently demonstrated [84].

Memristors are also used to design (sequential) logic operations based on Boolean functions [92] or (material) implication logic (IMP) [49, 58, 85]; the latter seems to be more popular. Figure 5 uses two examples to illustrate the concept of IMP. Figure 5(a) gives a basic logic function using two memristors. Together with a load resistor $R_G$, the operation $p \text{ IMP } q$, the operation $p \text{ IMP } q$, with superior performance, is suggested in [93], as shown in Figure 5(b). The input signals $V_P = \pm \frac{1}{2}V_{\text{Write}}$ and $V_Q = \pm \frac{1}{2}V_{\text{Write}}$ are applied at the terminals T1 and T2 of the memristor. The final result is stored as resistive state $Z$. For $Z = p \text{ IMP } q$ the following steps are performed:

1. Init device $Z$ to ‘1’ ($V_{T1} = +\frac{1}{2}V_{\text{Write}}$, $V_{T2} = -\frac{1}{2}V_{\text{Write}}$)
2. $Z' = p \text{ IMP } q$ ($V_{T1} = V_Q$, $V_{T2} = V_P$)
3. Read $Z'$

IMP can be used to design arithmetic operations such as adders [58, 56]; hence, it paves the path to more complex memristive in-memory-computing architectures.

V. CONCLUSION

This paper discusses data storage and analysis as one of the most critical challenges for today's and future data-intensive and big-data problems. It shows how the increase of the data size has already surpassed the capabilities of today's computation architectures suffering from the limited bandwidth, energy inefficiency and limited scalability. Thereafter, the paper proposes a new architecture based on the integration of the storage and computation in the same physical location (using a crossbar topology); the architecture is driven by non-volatile resistive-switching technology (memristors) instead of traditional CMOS technology. Therefore, it has the potential to reduce both the memory wall and energy consumption with orders of magnitude, and enables massive parallelism. Hence, significantly improving the performance and enabling the solution of big-data problems. The details and many aspects of the architecture still need to be worked out.
We would like to thank all people who contributed to the discussion of CIM architectures including Zaid Al-ars from del! University of Technology, Jan van Dallion from Eindhoven University of Technology, Luca Benini from ETHZ, Shahar Kvatinsky from Technion, Lotfi Mhamdi from Leed University, Albert Cohen from INRIA, Stephon Menzel and Vikas Rana from RWTH, and Andrea Fantini from IMEC.

REFERENCES


[29] H. Platter, ‘SannsouciDB: An In-Memory Database for Processing Enterprise Workloads’, Datenbanksysteme in Büro, Technik und Wissenschaft (German Database Conference), 2011


