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

[1]Said Hamdioui [1]Lei Xie [1]Hoang Anh Du Nguyen [1]Mottaqiallah Taouil [1]Koen Bertels [2]Henk Corporaal [2]Hailong Jiao

[1]Computer Engineering
Delft University of Technology
Delft, the Netherlands
First_Letter_First_Name.LastName@tudelft.nl

[2]Electronic Systems group
Eindhoven University of Technology
Eindhoven, The Netherlands
{h.corporaal, H.Jiao}@tue.nl

[3]Francky Catthoor [3]Dirk Wouters [4]Linn Eike [5]Jan van Lunteren

[3]IMEC, Kapeldreef 75, B-3001
Leuven, Belgium
<catthoor@imec.be>

[4]RWTH Aachen University
Aachen, Germany
linn@IWE.RWTH-Aachen.de

[5]IBM Research Laboratory
Zurich, Switzerland
jvl@zurich.ibm.com

***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 the highly useful information hidden in the huge volume of data, which in turn can be used to increase e.g., the productivity. Storing and analysing such data is posing major challenges as the data volume already surpassed the capability of today's computers which suffer from e.g., communication and memory-access bottlenecks due to limited bandwidth [2-lahiri, 3-somavat]. For instance, a transfer of 1 petabyes data at a rate of 1000MB/second will take 12.5 days! Memory size and memory access do not only kill the performance, but also severely impact energy/power consumption [2, 3 ,4]. In addition, CMOS technology used to implement today's architectures contributes to such consumption due to high leakage currents; not to mention other challenges the technology is facing such as limited scalability, reduced reliability [30-34], etc. In conclusion, today's CMOS based architectures are not able to provide the computation capability needed for data-intensive applications. *New* architectures based on *new* technologies are urgently required.

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, 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.

978-3-9815370-4-8/DATE15/©2015 EDAA

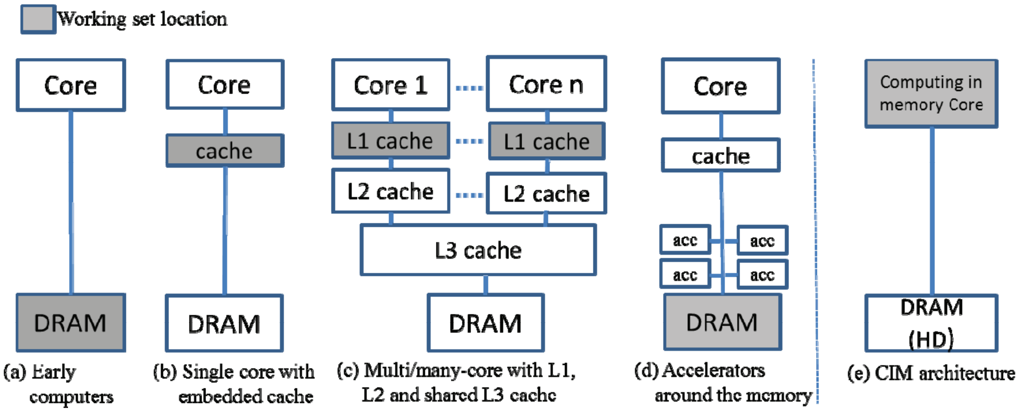

Figure 1: Classification of computing systems based on working set location

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.

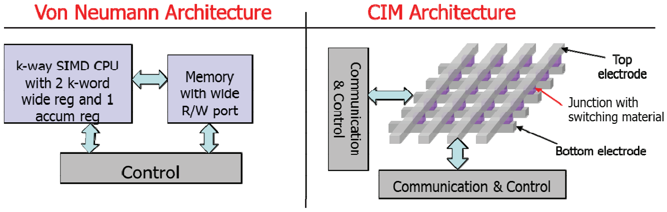

Figure 2: Traditional versus proposed architecture

- *An architecture with practically zero leakage:* Today's architectures heavily rely on SRAMs as caches. These are required to have a very fast R/W access, leading to increasingly high leakage with technology scaling. Hence, the memristor crossbar architecture solves also the leakage bottleneck, at least in the memory.
- *Significant performance improvement at lower energy and area:* Given the nature of the architecture (supporting massive parallelism), the non-volatile technology in the crossbar, and the small feature size of the memristor, the architecture has the potential of improving the overall performance at extremely low power consumption and smaller area. The next section illustrates this potential for two different applications.

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% coverages [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;

Table 1: Assumptions made for conventional and CIM architectures

Assumption for conventional architecture	Assumptions for CIM architecture

Generic assumptions
• FinFET 22nm multi-core implementation
o Gate delay = 14 ps [53, 54]
o Area per gate: 0.248 μm^2 [30]
o Power consumption per gate: 175 nW [54]
o Leakage power: (a) Leakage power consumption per gate: 42,83 nW [30], (b) Leakage duration: cycle time – delay per gate
o Operating frequency: 1 GHz
• The architecture consist of a certain number of clusters of processing units, each cluster shares an 8kB L1 cache.
For Healthcare example
• Typically, the DNA reference sequence must be covered 50 times by short reads. The length of the short reads are assumed to be 100 characters.
• 200GB of DNA data is compared to a healthy reference of 3GB.
• Number of short reads no_short_reads = coverage * 3 * giga/ short_read_len; coverage=50, short_read_len=100
• Number of comparisons no_comparsions = 4*no_short_reads, for each A, C, G, T nucleotides
• Number of clusters is 18750, each contains 32 comparators.
o Limited with the state-of-the-art chip area
• Each cluster shares 8 kB Cache (per cluster)
o Area: 0.0092 mm^2 [57]
o Hit ratio = 50%; Hit cycle time = 1 cycle
o Miss penalty = 165 cycle [55];
o Write cycle time = 1 cycle;
o Static power: 1/64 Watt [56]
For Mathematics example
• Fully scalable *reusing* clusters; each has 8 kB shared cache.
• Additions are performed by 32 adders per cluster:
o Adder architecture: Carry Look Ahead (CLA)
o Number of gates per adder: 208 [52]
o Number of gate delay: 18;
o Adder latency: 252ps = 18*14ps
o Energy per 32-bit adder
• Shared 8 kB Cache (per cluster): the same as for healthcare except with 98% hit rate,
	Generic assumptions
	• Memristor 5nm crossbar implementation [30]
	o Memristor write time: 200 ps [60]
	o Area per memristor: 1x10^{-4} um^2 [30]
	o Dynamic energy per write operation: 1 fJ [30]
	• The memory capacity of the CIM architectures is assumed to be equal to the sum of all caches for the CMOS based computer.
	For Healthcare example
	• Each comparison is performed by a comparator
	o Comparator: 2 XOR and a NAND implemented by implication logic [58]
	o Number of memristors per comparator: 13 (XOR: 5, NAND: 3)
	o Area per comparator: 1.3 * 10^{-3} um^2 [58]
	o Number of steps per comparator: 16 steps (Two XOR works in parallel, an XOR takes 13 steps, and an NAND takes 3 steps, step takes a memristor write time). [58]
	o Comparator latency: 3.2 ns
	o Dynamic energy per comparator: 45fJ [58]
	o Static energy per comparator: 0 fJ [30]
	• The crossbar size equals to total cache size of CMOS computer
	o Size= 18750*8kB = 1.536*10^8 memristors
	o Date hit rate = 50%, Hit cycle time = 1 cycle
	o Miss penalty = 165 cycle
	For Mathematics example
	• The crossbar is scalable to support the 10^6 adders
	• Additions are performed by memristors
	o Adder architecture: TC-adder [59]
	o Number of memristors per adder: 34 (N+2, N=32) [59]
	o Area per adder: 3.4x10^{-3} um^2
	o Number of steps per 32-bit addition: 133 (4N+5, N=32, each step takes a memristor write time). [59]
	o Adder latency: 16600 ps (133 * 200 ps)
	o Dynamic energy per 32-bit adder: 246 fJ (8 (operations per bit) *32 (bits) * 1 fJ [59])
	o Static energy per 32-bit adder: 0 fJ [30]
	• The memory hit rate is assumed to be 98%, remaining parameters are the same as for the healthcare example.

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 technology, and the high parallelism are enabling these improvements.

Table 2: Huge potential of CIM architecture

Metric	Archit.	DNA Sequencing	10^6 additions
Energy-delay/ operations	Conv.	2.0210e-06	1.5043e-18
	CIM	2.3382e-09	9.2570e-21
Computing efficiency	Conv.	4.1097e+04	6.5226e+09
	CIM	3.7037e+07	3.9063e+12
Performance area	Conv.	5.7312e+09	5.1118e+09
	CIM	5.1118e+09	4.9164e+12

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. Examples of its use are replacement of RAM, flash and even disk drives, complex self-learning neural networks, advanced artificial neural brains, and many more [61]. It may play a significant role in advancing Exascale 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 (HfO_x) and ECM (Ag-chalcogenide) devices a feature size of $F = 10$ nm was reported [62, 63]. A minimum switching time of < 200 ps was shown for TaO_x-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 TaO_x-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] (TaO_x-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:

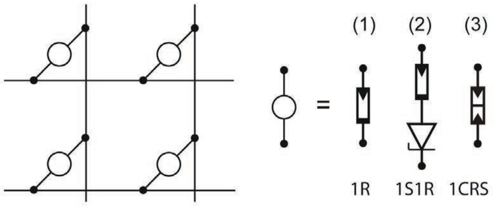

Figure 3: Illustration of complementary resistive switch

- *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.

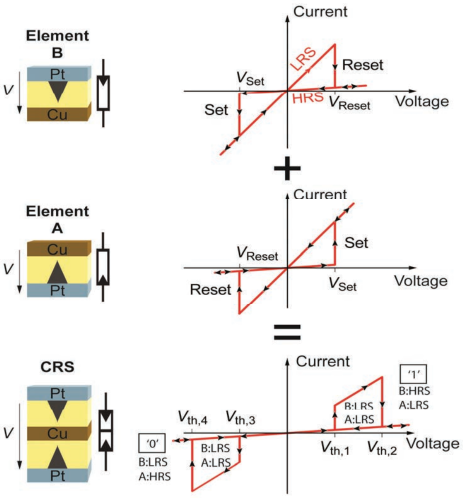

Figure 4: Left: Zoom in a passive nano-crossbar array. Right: possible cross point junctions

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 next .

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* IMP *q* is conducted as follows [49]:

1. Set device p to *p* ($V_P = \pm V_{Write}$)
2. Set device Q to *q* ($V_Q = \pm V_{Write}$)
3. *q*' = *p* IMP *q* ($V_P = V_{COND}$ and $V_Q = V_{write}$)
4. Read *q*'

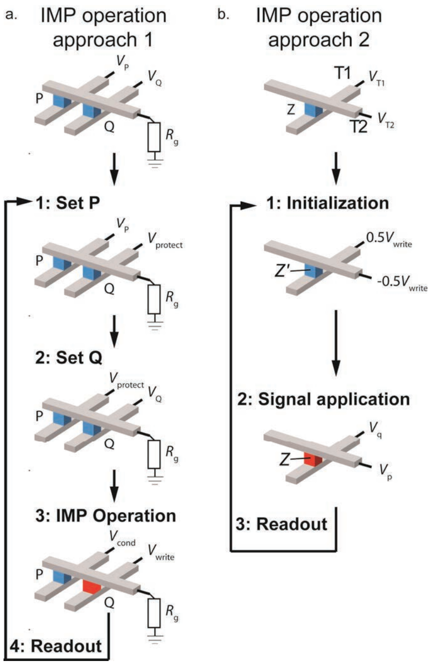

Figure 5: Two ways to implement IMP. Blue cube represents state '0' and the red cube state '1'

An alternative approach to implement *p* IMP *q,* with superior performance, is suggested in [93], as shown in Figure 5(b). The input signals $Vp = \pm½V_{Write}$ and $Vq = \pm½V_{Write}$ are applied at the terminals T1 and T2 of the memristor. The final result is stored as resistive state Z. For Z = p IMP q the following steps are performed:

1. Init device Z to '1' ($VT1 = +½V_{Write}$, $VT2 = -½V_{Write}$)
2. Z' = p IMP q (VT1 = Vq, VT2 = Vp)
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.

ACKNOWLEDGMENT

We would like to thank all people who contributed to the discussion of CIM architectures including Zaid Al-ars from delft University of Technology, Jan van Dalfson from Eindhoven University of Technology, Luca Benini from ETHZ, Shahar Kvatinsky from Technion, Lotfi Mhamdi from Leed University, Albert Cohen from INRIA, Stephan Menzel and Vikas Rana from RWTH, and Andrea Fantini from IMEC.

REFERENCES

[1] P. Chen and C-Y Zhang, 'Data-intensive applications, challenges, techniques and technologies: A survey on Big Data', Information Sciences, v 275, pp. 314-347, 2014

[2] K. Lahiri, A. Raghunathan, 'Power analysis of system-level on-chip communication architectures', International Conference on Hardware/Software Codesign and System Synthesis, CODES + ISSS, pp 236-241, 2014

[3] P. Somavat and V. Namboodiri. 'Energy consumption of personal computing including portable communication devices', Journal of Green Engineering, 1(4):447–475, 2011

[4] M. Horowitz, 'Computing's Energy Problem and what we can do about it', slides of the keynote at ISSCC 2014

[5] A.W. Burks, et al., 'Preliminary discussion of the logical design of an electronic computing instrument', 1946

[6] H. Esmaeilzadeh, et al., 'Dark silicon and the end of multicore scaling', in Proceedings of the 38th annual international symposium on Computer architecture, pp. 365-376, 2011

[7] S. Bampi, and R. Reis, 'Challenges and Emerging Technologies for System Integration beyond the End of the Roadmap of Nano-CMOS', 'VLSI-SoC: Technologies for Systems Integration' (Springer Berlin Heidelberg), pp. 21-33, 2011

[8] T. Yuan, 'CMOS design near the limit of scaling', IBM Journal of Research and Development, 2002, 46, (2.3), pp. 213-222

[9] X.-J. Yang et al., 'Progress and Challenges in High Performance Computer Technology', J. Comp. Sci. Tech., 2006, 21, (5), pp. 674-681

[10] S. A. McKee, 'Reflections on the Memory Wall', CF'04, pp. 162, 2004.

[11] M.V. Wilkes, 'The Memory Wall and the CMOS End-point', SIGARCH Comput. Archit. News, 1995, 23, (4), pp. 4-6

[12] R.E. Bryant, 'Data-Intensive Scalable Computing for Scientific Applications', Computing in Science & Engin., 2011, 13, (6), pp. 25-33

[13] W.A. Wulf and S.A. McKee, 'Hitting the memory wall: implications of the obvious', SIGARCH Comp. Archit. News, 23, pp. 20-24, 1995

[14] D. Patterson et al. 'A case for intelligent RAM', IEEE Micro, 1997, 17, (2), pp. 34-44

[15] C. Hernandez et al., 'Energy and Performance Efficient Thread Mapping in NoC-Based CMPs under Process Variations', in International Conference on Parallel Processing, 2011, pp. 41-50

[16] Y. Kang et al., 'FlexRAM: Toward an advanced Intelligent Memory system', in IEEE 30th International Conference on Computer Design, pp. 5-14, 2012

[17] J. Draper et al., 'The architecture of the DIVA processing-in-memory chip', In Proceedings of the 16th international conference on Supercomputing, pp. 14-25, 2002

[18] M. Gokhale et al., 'Processing in memory: the Terasys massively parallel PIM array', Computer, vol. 28, pp. 23-31, 1995

[19] P. M. Kogge, 'EXECUBE-A New Architecture for Scaleable MPPs', in International Conference on Parallel Processing, Vol. 1, 1994, pp. 77-84

[20] P. M. Kogge et al., 'PIM architectures to support petaflops level computation in the HTMT machine', in International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, 1999, pp. 35-44

[21] D. G. Elliott et al., 'Computational RAM: implementing processors in memory', IEEE Design Test of Computers, vol. 16, pp. 32-41, 1999

[22] Z. Wang et al., 'DSP-RAM: A logic-enhanced memory architecture for communication signal processing', in IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 1999, pp. 475-478

[23] M. Ken et al., 'Smart Memories: a modular reconfigurable architecture', in Proceedings of the 27th International Symposium on Computer Architecture, 2000, pp. 161-171

[24] T. L. Sterling and H. P. Zima, 'Gilgamesh: A Multithreaded Processor-In-Memory Architecture for Petaflops Computing', in ACM/IEEE Conference Supercomputing, 2002, pp. 48-48

[25] T. Sterling and M. Brodowicz, 'Continuum computer architecture for nano-scale and ultra-high clock rate technologies', in Innovative Architecture for Future Generation High-Performance Processors and Systems, 2005, pp. 1-9

[26] P. Dlugosch et al., 'An Efficient and Scalable Semiconductor Architecture for Parallel Automata Processing', IEEE Transactions on Parallel and Distributed Systems, 2014, vol. 99, pp. 3088- 3098

[27] N. Venkateswaran et al., 'Memory in Processor: A Novel Design Paradigm for Supercomputing Architectures', 2003, pp. 19-26

[28] L. W. Tucker and G. G. Robertson, 'Architecture and applications of the Connection Machine', Computer, vol. 21, pp. 26-38, 1988

[29] H. Plattner, 'SanssouciDB: An In-Memory Database for Processing Enterprise Workloads', Datenbanksysteme in Büro, Technik und Wissenschaft (German Database Conference), 2011

[30] ITRS ERD report. [Online]. Available: http://www.itrs.net/Links/2010ITRS/Home2010.htm, 2010

[31] B. Hoefflinger, 'Chips 2020: A Guide to the Future of Nanoelectronics', The Frontiers Collection, Springer Berlin Heidelberg, 2012, pp. 421–427

[32] J. McPherson, 'Reliability trends with advanced CMOS scaling and the implications for design', in IEEE Custom Integrated Circuits Conference, 2007, pp. 405–412

[33] S. Borkar, 'Design perspectives on 22Nm CMOS and beyond', in Proceedings of the 46th Annual Design Automation Conference, 2009, pp. 93-94

[34] G. Gielen, et al., 'Emerging yield and reliability challenges in nanometer CMOS technologies', in Proceedings of the Conference on Design, Automation and Test in Europe, pp. 1322-1327, 2008

[35] F. Schwierz, 'Graphene transistors', Nature nanotechnology, vol. 5, no. 7, pp. 487–496, 2010

[36] G. Agnus et al., 'Two-terminal carbon nanotube programmable devices for adaptive architectures', Advanced Materials, vol. 22, no. 6, pp. 702–706, 2010

[37] K. Boucart and A. M. Ionescu, 'Double-gate tunnel fet with high-gate dielectric', IEEE Transactions on Electron Devices, vol. 54, no. 7, pp. 1725–1733, 2007

[38] L. Chua, 'Memristor-the missing circuit element', IEEE Transactions on Circuit Theory, vol. 18, no. 5, pp. 507–519, 1971

[39] D. B. Strukov et al., 'The missing memristor found', Nature, vol. 453, no. 7191, pp. 80–83, May 2008

[40] D. B. Strukov et al., 'Hybrid cmos/memristor circuits', in IEEE International Symposium on Circuits and Systems, 2010, pp. 1967–1970

[41] H. Lee et al., 'Low-power and nanosecond switching in robust hafnium oxide resistive memory with a thin ti cap', IEEE Electron Device Letters, vol. 31, no. 1, pp. 44–46, 2010

[42] A. C. Torrezan et al., 'Sub-nanosecond switching of a tantalum oxide memristor', Nanotechnology, vol. 22, no. 48, pp. 1-7, 2011

[43] M.-J. Lee et al., 'A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta2O5x/TaO2x bilayer structures', Nature Materials, vol. 10, no. 8, pp. 625–630, 2011

[44] T. Prodromakis and C. Toumazou, 'A review on memristive devices and applications', in 17th IEEE International Conference on Electronics, Circuits and Systems, 2010, pp. 934–937

[45] W. Zhao et al., 'Nanodevice-based novel computing paradigms and the neuromorphic approach', in IEEE International Symposium on Circuits and Systems, 2012, pp. 2509–2512

[46] Y. S. Chen et al., 'Highly scalable hafnium oxide memory with improvements of resistive distribution and read disturb immunity', in IEEE International Electron Devices Meeting, 2009, pp. 1–4

[47] J. J. Yang et al., 'High switching endurance in tao memristive devices', Applied Physics Letters, vol. 97, p. 232102- 232103, 2010

[48] J. J. Yang et al., 'Memristive devices for computing', Nat. Nanotechnol., vol. 8, pp. 13-24, 2013

[49] J. Borghetti et al., ''Memristive' switches enable 'stateful' logic operations via material implication', Nature, 464, pp. 873-876, 2010

[50] S. Hamdioui et al., 'Memristor based memories: Technology, design and test', IEEE 9th International Conference on Design & Technology of Integrated Systems In Nanoscale Era, pp. 1-7, 2014

[51] E. A. Worthey, 'Analysis and annotation of whole-genome or whole-exome sequencing-derived variants for clinical diagnosis', Current Protocols in Human Genetics, 2001

[52] B. Parhami, Computer arithmetic: algorithms and hardware designs. Oxford University Press, Inc., 2009

[53] A. Muttreja et al., 'CMOS logic design with independent-gate FinFETs', 25th International Conference on Computer Design, pp. 560–567, 2007

[54] C. Meinhardt et al., 'FinFET basic cells evaluation for regular layouts', in IEEE Fourth Latin American Symposium on Circuits and Systems, pp. 1–4, 2013

[55] D. Levinthal, Cycle Accounting Analysis on Intel R CoreTM2 Processors. Intel Corp, 2008

[56] C.Y. Lee et al., 'CACTI-FinFET: An integrated delay and power modeling framework for FinFET-based caches under process variations', in 48th Design Automation Conference, pp. 866–871, 2011

[57] E. Karl et al., 'A 4.6 GHz 162Mb SRAM design in 22nm tri-gate CMOS technology with integrated active V MIN-enhancing assist circuitry', in IEEE International Solid-State Circuits Conference Digest of Technical Papers, pp. 230–232, 2012

[58] S. Kvatinsky et al., 'Memristor-Based Material Implication (IMPLY) Logic: Design Principles and Methodologies', IEEE Transactions on VLSI Systems, vol. 22, no. 10, pp. 2054–2066, 2014

[59] A. Siemon et al., 'A Complementary Resistive Switch-based Crossbar Array Adder', arXiv:1410.2031, 2014

[60] J. Walker, Memristor and the future, [Online]. Available: http://www.nobeliefs.com/memristor.htm, 2010

[61] R. Waser et al., 'Redox-Based Resistive Switching Memories - Nanoionic Mechanisms, Prospects, and Challenges', Advanced Materials, vol. 21, pp. 2632-2663, 2009

[62] B. Govoreanu et al., '10x10nm2 Hf/HfOx Crossbar Resistive RAM with Excellent Performance, Reliability and Low-Energy Operation', IEEE International Electron Devices Meeting, pp. 31.6.1-31.6.4, 2011

[63] K. Terabe et al., 'Quantized conductance atomic switch', Nature, vol. 433, pp. 47-50, 2005

[64] M. Meier et al., 'A Nonvolatile Memory With Resistively Switching Methyl-Silsesquioxane', IEEE Electron Device Letters, vol. 30, pp. 8-10, 2009

[65] M. N. Kozicki et al., 'Nanoscale memory elements based on solid-state electrolytes', IEEE Transactions on Nanotechnology, vol. 4, pp. 331-338, 2005

[66] Z. Wei et al., 'Retention model for high-density ReRAM', 4th IEEE International Memory Workshop, pp. 1-4, 2012

[67] M. Kund et al., 'Conductive bridging RAM (CBRAM): an emerging non-volatile memory technology scalable to sub 20nm,' IEEE International Electron Devices meeting Technical Digest, pp. 754 – 757, 2005

[68] S. Ferch et al., 'Simulation and Comparison of two Sequential Logic-in-Memory Approaches Using a Dynamic Electrochemical Metallization Cell Model', Microelectronics Journal, vol. 45, pp. 1416-1428, 2014

[69] R. S. Williams et al., 'Physics-based memristor models', IEEE International Symposium on Circuits and Systems, pp. 217-220, 2013

[70] E. Linn et al., 'Applicability of Well-Established Memristive Models for Simulations of Resistive Switching Devices', IEEE Transactions on Circuits and Systems, vol. 61, pp. 2402 - 2410, 2014

[71] M. D. Pickett et al., 'Switching dynamics in titanium dioxide memristive devices', Journal of Applied Physics, 2009

[72] J. P. Strachan et al., 'State Dynamics and Modeling of Tantalum Oxide Memristors', IEEE Transactions on Electron Devices, vol. 60, pp. 2194-2202, 2013

[73] D.B. Strukov et al., 'Prospects for Terabit-scale Nanoelectronic Memories', Nanotechnology, vol. 16, no. 1, pp. 137–148, 2005

[74] H.D. Lee et al., 'Integration of 4F2 selector-less crossbar array 2Mb ReRAM based on transition metal oxides for high density memory applications', Symposium on VLSI Technology, pp. 151-152, 2012

[75] N. Ramaswamy, 'Challenges in Engineering RRAM technology for high density applications', IEEE International Workshop On Integrated Reliability, pp. 1-5, 2012

[76] A. Flocke et al., 'Fundamental analysis of resistive nano-crossbars for the use in hybrid Nano/CMOS-memory', 33rd European Solid-State Circuits Conference, pp. 328-331, 2007

[77] H. Manem et al., 'Design considerations for variation tolerant multilevel cmos/nano memristor memory', in Symposium on Great Lakes Symposium on VLSI, pp. 287-292, 2010

[78] E. Linn et al., 'Complementary Resistive Switches for Passive Nanocrossbar Memories', Nature Materials, vol. 9, pp. 403-406, 2010

[79] J.J. Yang et al., 'Engineering nonlinearity into memristors for passive crossbar applications', Applied Physics Letters, 2012

[80] M. A. Zidan et al., 'Memristor-based memory: The sneak paths problem and solutions', Microelectronics Journal, 44 (2), pp. 176-183, 2013

[81] M. Qureshi et al., 'AC sense technique for memristor crossbar', Electronics Letters, vol. 48, pp. 757-758, 2012

[82] M. R. Stan et al., 'Molecular electronics: from devices and interconnect to circuits and architecture', Proceedings of the IEEE, vol. 91, pp. 1940-1957, 2003

[83] M. Liu et al., 'Application of nanojunction-based RRAM to reconfigurable IC', Micro Nano Letters, vol. 3, pp. 101-105, 2008

[84] L. Nielen et al., 'An Experimental Associative Capacitive Network based on Complementary Resistive Switches for Memory-intensive Computing', 2014 IEEE Silicon Nanoelectronics Workshop, 2014

[85] E. Lehtonen et al., 'Stateful Implication Logic with Memristors', IEEE/ACM International Symposium on Nanoscale Architectures, pp. 33-36, 2009

[86] A. Dehon, 'Nanowire-Based Programmable Architectures', ACM Journal on Emerging Technologies in Computing Systems, vol. 1, pp. 109-162, 2005

[87] K. K. Likharev et al., 'CMOL: Devices, Circuits, and Architectures', Introducing Molecular Electronics, vol. 680, pp. 447-477, 2006.

[88] S. Paul et al., 'Computing with Nanoscale Memory: Model and Architecture', pp. 1-6, 2009

[89] S. Paul et al., 'A Scalable Memory-Based Reconfigurable Computing Framework for Nanoscale Crossbar', IEEE Transactions on Nanotechnology, vol. 11, pp. 451-462, 2012

[90] K. Eshraghian et al., 'Memristor MOS Content Addressable Memory (MCAM): Hybrid Architecture for Future High Performance Search Engines', IEEE Transactions on VLSI Systems, vol. 19, pp. 1407-1417, 2011

[91] S.J. Lee et al., 'Complementary Resistive Switch (CRS) Based Smart Sensor Search Engine', 8th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing, pp. 485 - 490, 2013

[92] G. Snider, 'Computing with hysteretic resistor crossbars', Applied Physics A, vol. 80, pp. 1165–1172, 2005

[93] E. Linn et al., 'Beyond von Neumann-logic operations in passive crossbar arrays alongside memory operations', Nanotechnology, vol. 23, pp. 305205/1-6, 2012

[94] L.O. Chua, 'Resistance switching memories are memristors', Applied Physics A, vol. 102, pp. 765-783, 2011

[95] H Aziza, et. al, 'A novel test structure for OxRRAM process variability evaluation', - Microelectronics Reliability, Vol. 53, N. 9, pp.1208-1212, 2013