Memristor: The Enabler for Processing-in-Memory

Said Hamdioui

Computer Engineering, Delft University of Technology, the Netherlands
S.Hamdioui@tudelft.nl

Abstract

This paper briefly discusses a new architecture, Computation-In-Memory (CIM Architecture), which performs “processing-in-memory”. 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) instead of CMOS technology. The architecture has the potential of improving the energy-delay product, computing efficiency and performance area by at least two orders of magnitude.

The need of new architecture

Computing systems, developed since the introduction of stored program computers by John von Neumann in the forties, can be classified based on the location of the so-called “working set” (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. Caches were introduced to reduce the gap between the core (CPU) and the memory speed, and increase the overall performance; the caches have become the location of the working set. Today’s many/multi core (parallel CPUs, GPUs, SIMD-VLIWs, vector processors) computing systems are still based on Von Neumann (VN) architectures; see Figure 1(c). Recently, the design of high-performance computing systems based on data-centric approach (i.e., memory closer the processing unit and reducing the memory bottleneck) rather than conventional computation-centric model is attracting a lot of attention, although the concept is more than 40 years old; see Figure 1(d). In 1969, Logic-In-Memory (LIM) was originally introduced as a memory accelerator; i.e., add some processing units close to main memory. In 1992, LIM concept re-appeared and named computational RAM, and typically uses the same accelerator concept where these are supposed to perform operations needed by the memory such as address translations. In the late 1990s and early 2000s, Processor-In-Memory (PIM) was proposed and manufactured. PIM is based on splitting the main memory in different parts, each with surrounded computing units to bring the computation near to the memory; the architecture has a master CPU that takes care of the overall control. In 2004, Memory-In-Logic (MIL), which provides massive addressable memory on the processor, was proposed for supercomputer systems.

All mentioned above efforts have tried to close the gap between processor and memory speed. However, as the computation and the storage are kept separately, they fundamentally use the von Neumann stored-program computer concept and therefore suffer from a memory bottleneck, which negatively impacts the performance [1,2,3,4]. The situation becomes even worse considering increasing size of data-intensive applications and big data-problems. Clearly, the speed at which data is growing has already surpassed the capabilities of today’s computation architectures.

Very important is the fact that today’s computers are manufactured mainly using CMOS technology. Such technology is reaching inherent physical limits due to down-scaling, and is suffering from major limitations; high static power, reduced reliability, reduced performance gain, and higher production cost due to increased number masks and manufacturing tolerances are just couple of examples [3,4]. Hence the need of new device technology that can be combined with - or replace CMOS to sustain the probability of technology scaling is a must.

Memristor based CIM architecture

To overcome/ mitigate one or more of the disadvantages of the prior art, recently *Computing-In-Memory (CIM) architecture* [5,6,7] has been introduced; CIM takes the data-centric computing concept much further by interweaving 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).

Figure 2 illustrates the concept of CIM architecture; the storage and computation are integrated together in a dense crossbar array where memristors are injected at each crossbar junction (top electrode and bottom electrode). The communication is realized within the crossbar and/or with the support of CMOS block (communication and control); the latter is responsible for the overall control. Figure 3 shows the different levels of the control circuits that maps an algorithm on the architecture. Algorithms are compiled into macro-instructions, each comprises a set of micro-instructions. Micro-instructions are primitive operations such as single add/subtract; they are translated nanoinstructions, which are electrical signals (generated by CMOS control part) that independen tly control the columns and the rows of the memristor crossbar tile. A tile is a flexible unit performing a function (micro or macro-instruction) determined by the compiler to minimize communications in memristor crossbar array. The controller is a state machine which enables and distributes required functions to each tile at a time.

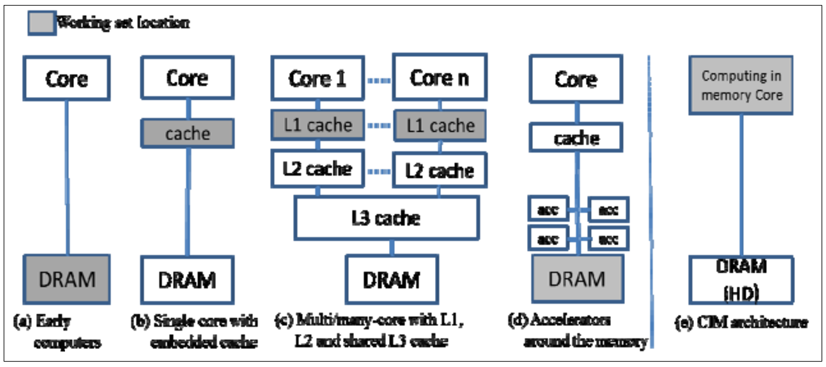

Fig.1: Classification of computing systems based on working set location

The preliminary results based on simulation for CIM architecture and compared with state-of-the art show that depending on the application, at least two order of magnitude improvements can be realised with regards to the energy-delay product per operations, the computation efficiency (defined as the number of operations per required energy), and performance (#operations) per area [5,6]; CIM performance is evaluated against a conventional architecture (Conv.) which is assumed to be a fully scalable multi-core architecture (22nm FinFET technology), consisting of clusters, each having 32 ALU's [5,6].

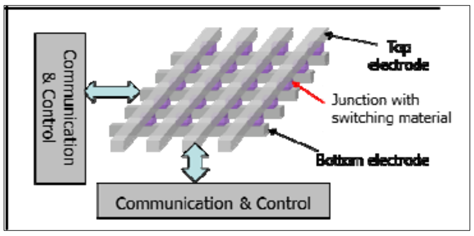

Fig. 2: CIM concept

The results clearly show an increase computing energy and area efficiency by orders of magnitude; *this enables the computation of currently infeasible big data applications, fuelling important societal change!*

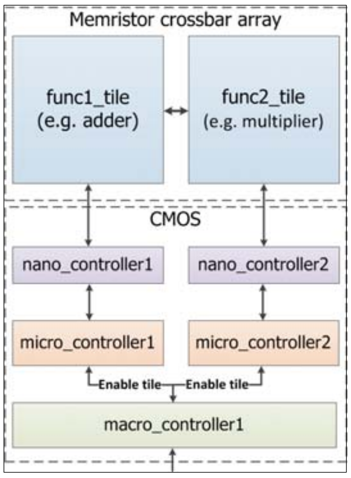

Fig. 3: Control circuit levels

References

[1] S. A. McKee, 'Reflections on the Memory Wall', CF'04, pp. 162, 2004.
[2] M.V. Wilkes, 'The Memory Wall and the CMOS End-point', SIGARCH Comput. Archit. News, 1995, 23, (4), pp. 4-6.
[3] Horowitz, 'Computing's Energy Problem and what we can do about it', slides of the keynote at ISSCC 2014
[4] K. Lahiri, A. Raghunathan, 'Power analysis of system-level on-chip communication architectures', CODES + ISSS, pp 236-241, 2014.
[5] S. Hamdioui, et. al, 'Memristor Based Computation-in-Memory Architecture for Data-Intensive Applications', Proceedings of the 2015 Design, Automation & Test in Europe & Exhibition, pp. 1718-1725, 2015.
[6] H.A. Du Nguyen, et.al, 'Computation-In-Memory Based Parallel Adder', IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), pp 57-62, 2015.
[7] S. Hamdioui, M. Taouil, K. Bertels, Scalable Computation Architecture in a Memristor-Based Array, Patent Application P6057067NL, July 2015.