CIM100x

Computation in-Memory Architecture Based on Resistive Devices

Hamdioui, Said; Taouil, Mottaqiallah; Du Nguyen, Hoang Anh; Haron, Adib; Xie, Lei; Bertels, Koen

Publication date
2016

Published in
Proceedings of CNNA 2016

CIM100x: Computation in-Memory Architecture Based on Resistive Devices

Said Hamdioui Mottaqiallah Taouil Hoang Anh Du Nguyen Adib Haron Lei Xie Koen Bertels

Computer Engineering, Delft University of Technology, the Netherlands
S.Hamdioui@tudelft.nl
Phone: +31- 15 278 3643; Fax: +31- 15 278 4898

1. Abstract

In this paper, we briefly discuss a new proposed architecture, Computation-In-Memory (CIM) architecture, that targets specifically data-intensive applications. The architecture consists of the interwoven placement of computing and storage units, which are physically tightly integrated together inside a non-volatile memristor crossbar. The architecture has the potential of improving the energy-delay product, computing efficiency and performance per area by at least two orders of magnitude with respect to conventional CMOS architectures.

2. The Need for a New Architecture

One of the most critical challenges of today's and future data-intensive and big-data problems (ranging from economics and business activities to public administration, from national security to many scientific research areas) is data storage and analysis. The required amount of data to process has already surpassed the capabilities of today's computation architectures, which suffer from the limited bandwidth [1,2,3] (due to memory-access bottlenecks), energy inefficiency and limited scalability [3,4,5] (due to CMOS technology).

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., having memory closer to the processing units) rather than conventional computation-centric model is attracting a lot of attention, although the concept is more than 40 years old [6]; see Figure 1(d).

Several efforts [6,7] 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. The situation becomes even worse when the size of data-intensive applications and big-data problems increases. Clearly, the speed at which data is growing has already surpassed the capabilities of today's computation architectures. Having supercomputers to solve big-data problems (as it is today for limited applications) is not affordable due to the cost (hundreds millions of US$) and power consumption (Megawatts).

Today's computers are manufactured mainly using CMOS technology. Such technology is reaching its 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 an increased number of masks and manufacturing tolerances are just a couple of examples [3,4,5]. Hence, the need of new device technology that can be combined with CMOS or even replace it, are needed to sustain technology scaling.

3. Memristor Based CIM Architecture

To overcome/ mitigate one or more of the disadvantages of the prior art, recently *Computing-In-Memory (CIM) architecture* [8,9] 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 map 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 into nano-instructions, which are electrical

ISBN 978-3-8007-4252-3

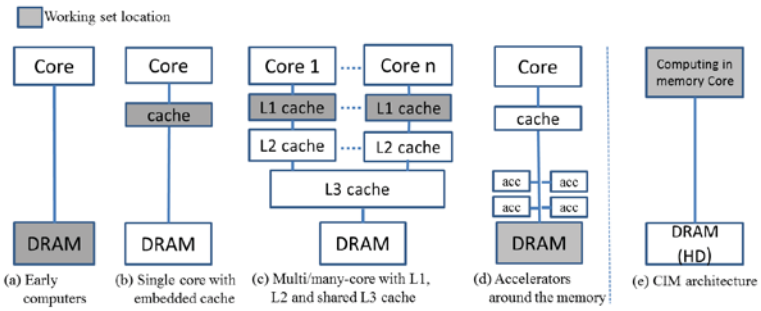

Fig.1. Classification of computing systems based on working set location

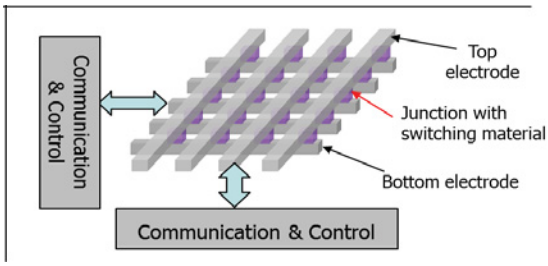

Fig. 2. CIM Concept

signals (generated by CMOS control part) that independently 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.

CIM architecture has huge potentials which go substantially beyond the current state-of-the-art:

- *Non program stored based-computer*: CIM interweaves the storage and computing units, which significantly reduces the memory bottleneck. Instructions are performed on data by surrounding computing elements. Therefore, traditional cache misses are not applicable.
- *Flattened data-memory*: all the traditional memory hierarchies are combined in a single distributed memory across the crossbar.
- *Practically zero leakage*: Today's architectures heavily rely on SRAM caches. These are required to have a very fast R/W access, leading to higher leakage with technology scaling. Hence, the memristor crossbar architecture solves the leakage bottleneck, at least in the memory.
- *Near zero communication*: Fully configurable flattened data-memory enables a communication optimization between storage and computing units.
- *Full configurability and flexibility*: the crossbar can be optimized for storage, computation and communication at each location within the crossbar. The architecture can be configured statically at design stage (as an accelerator) or dynamically at run-time (general purpose). An algorithm (program) can be compiled into a set of macro-instructions; no restrictions are put on the size and the functionality of the macro-instruction.
- *Explore maximum parallelism*: The interweaved nature of the architecture and the size of the crossbar enable massive parallelism. The maximum parallelism of each algorithm (program) is explored by compiling it into macro-instructions (functions); these are performed in parallel within the crossbar.

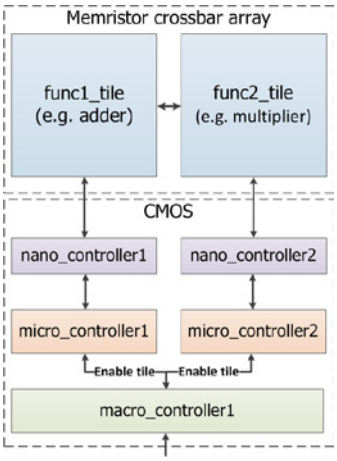

Fig. 3. Control Circuits

The preliminary results based on simulation for CIM architecture show that depending on the application, at least two order of magnitude improvements can be realised with respect to state-of-the art regarding the energy-delay product per operations, the computation efficiency (defined as the number of operations per required energy), and performance (#operations) per area [8,9]. The results clearly show an increased computing energy and area efficiency by orders of magnitude; this enables the computation of currently infeasible big data applications, fuelling important societal change!

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, 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, 'Reliability Challenges of Real-Time Systems in Forthcoming Technology Nodes', DATE, pp. 129-134, 2013.
[6] H. Stone, "A logic-in-memory computer," TC, vol C-19, 1970.
[7] N. Venkateswaran et al., "Memory in Processor: A Novel Design Paradigm for Supercomputing Architectures," in MEDEA '03, 2003.
[8] S. Hamdioui, et. al, 'Memristor Based Computation-in-Memory Architecture for Data-Intensive Applications', DATE, pp. 1718-1725, 2015.
[9] H.A. Du Nguyen, et.al, 'Computation-In-Memory Based Parallel Adder', NANOARCH, pp 57-62, 2015

