Logic-In-Memory Architecture Made Real

D, Pala, G. Causapruno, M. Vacca, F. Riente, G. Turvani, M. Graziano and M. Zamboni
Politecnico di Torino, Department of Electronics and Telecommunications, Corso Duca degli Abruzzi, 24, I10129 Torino, Italy
Email: {giovanni.causapruno, marco.vacca, fabrizio.riente, giovanna.turvani, mariagrazia.graziano, maurizio.zamboni}@polito.it

***Abstract*—The current trend for intensive computational architectures is to adopt massive parallelism, with several concurrent tasks performed simultaneously, as done for example in GPUs. This approach has many advantages, such as the reduced design time given by circuit replication and an increasing in computational speed without the need of higher frequency. It has however evidenced an important bottleneck in data exchange between memory and processor.**

We envisage a revolutionary path for the future relation between memory and logic in parallel processors, where a new type of architecture exploits the principle of caching to the limit. Our Logic-in-Memory (LIM) architecture mixes logic and memory in the same device, removing the bottleneck of other existing parallel solutions. The architecture we propose, here in its preliminary version, has an array organization and each element in the array is based on three blocks: a logic unit for processing, a smart memory block and a routing structure for inter block communication. In this article we show the benefits of this approach with an application example in the image processing field. We can achieve a 4X computational time reduction for an image processing algorithm (Summed Area Table) with respect to the best architecture present in the literature, even with a preliminary and not optimized version.

Besides the adoption of massive parallelism to increase performance, new technologies to open the post-CMOS era are explored. Among them NanoMagnet Logic (NML) is particularly interesting for its ability to mix logic and memory in the same device. We present here the preliminary results of the NML implementation of the LIM architecture. We thus demonstrate that it is not only a good solution for a standard CMOS technology but can also exploit the potential of an emerging technology as NML.

I. Introduction

With the growing sizes of database and files, most algorithms used nowadays are parallel algorithms that work on a huge amount of data. Running these algorithms on general purpose processors would require a long computational time, often not compliant with the stringent performance requirements by these tasks. The only way to execute these algorithms in a reasonable amount of time is to process as many data as possible in parallel. Moreover, while in last decades CMOS scaling allowed remarkable increase in operating frequency processing data in shorter time, this trend has suddenly decayed in last years for intrinsic technological limitations [1]. For these reasons parallel architectures, that allow to process several data at the same time, are recently gaining unprecedented interest.

Current parallel structures comprise both architectural solutions (GPUs, Systolic Arrays) and specific physical organization (FPGAs) [2]. They are employed in different fields

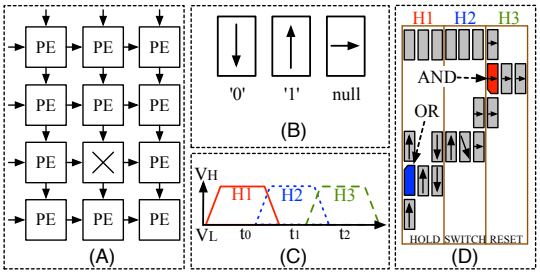

Fig. 1. (A) Array Processor with marked central cell. (B) NanoMagnet Logic (NML) principle: magnetization represents logic values. (C) Clocking scheme for magnetic QCA circuits. (D) An example of signal propagation in a NML circuit. The circuit is split in 3 different clock zones. Nanomagnets with one cut corner implement logic functions AND, OR.

and for different applications, but they all share some peculiar characteristics: 1) The amount of memory they embed is extremely limited, requiring continuous communication between the Array Processor and the external memory. 2) The communication between the processor and external elements is provided through a limited number of communication ports. In a rectangular processor as in Fig. 1.A, the communication ports are placed on boundary cells. If an input data must be provided to one central cell (labeled with X in Fig. 1.A), a certain delay is required to transmit this data through several cells.

These characteristics highlight that both the mechanism to provide inputs and the access to memory during algorithm execution are inefficient. A change of perspective is required to overcome these main limitations. We propose then a revolutionary architecture that mixes logic and memory in the same device, exploiting the Logic-in-Memory (LIM) concept [3]. This is a layered architecture, with a "smart" memory plane, a computational plane to execute complex algorithms on stored data and a routing plane to transfer data from cell to cell. Though still in its preliminary form, it already allows to solve the abovementioned problems and we demonstrate it with an application case frequently adopted in image processing. We have designed at RTL level this architecture to demonstrate the feasibility of this solution and its effectiveness, and we compared our preliminary results to previous solutions proposed in literature.

We have also designed a NanoMagnetic Logic (NML) implementation in order to demonstrate that its characteristics could be further exploited if an emerging beyond CMOS technology could be used. While the results here presented are just a preliminary work, our long term aim is to optimize

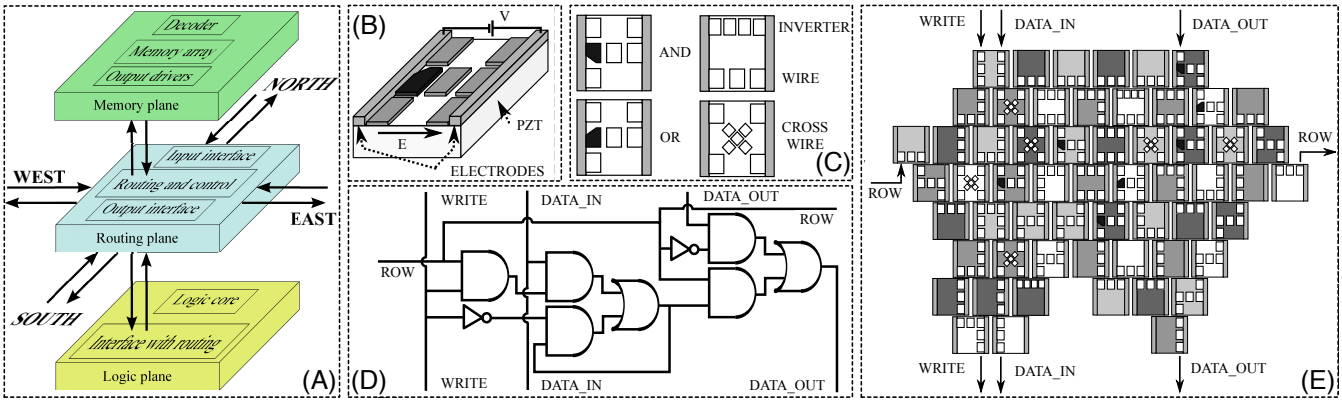

Fig. 2. (A) The LIM cell layered architecture. The routing plane communicates with memory plane above, logic plane below and with near cells. (B) Magnetoelastic NML. Magnets are placed on a piezoelectric substrate. When an electric field is applied the substrate strain induces a magnetization rotation on magnets. (C) Magnetoelastic NML logic gate set.(D) Memory cell schematic. Common buses and a distributed multiplexer architecture is used to reduce interconnections overhead. (E) Memory cell layout.

the architecture for NML, fully exploiting the peculiar characteristics of this technology.

In this article we lay the foundations for this radical change in perspective, introducing the new LIM architecture and showing an application example in the image processing field. We underline that results are only preliminary. However, the reader can perceive the remarkable potential of this structure and the way it is expected to change the relationship between logic cores and memories that represent the bottleneck of today's systems.

In the following some backgrounds on magnetic logic are given in section II and a description of the LIM architecture is discussed in section III. Then hints on the particular application and the related results are reported in section IV and V, respectively.

II. NanoMagnet Logic

The previously mentioned limits of CMOS scaling have induced, besides an increasing interest in parallel architectures, a new research path towards nanotechnologies that could replace the current technology in the future [4]. QCA [5] represents one of the most promising emerging technologies because it can guarantee low power consumption [6], in the so-called NanoMagnet Logic (NML) implementation [7]. This technology is based on nanomagnets whose magnetization represents an encoding binary value (Fig. 1.B). Nanomagnets align their state according to neighbor elements, thanks to magnetic interaction, thus propagating information through the circuit. A 3-phase clock mechanism, represented in Fig. 1.C-D, is used to guarantee correct propagation of signals. Circuits are divided in small areas called clock zones. At every clock zone one of 3 clock signals is applied, which forces the magnets in different states. During the RESET state, magnets are placed in an intermediate unstable state. Magnets in the SWITCH state are influenced only by those in HOLD state, propagating the information correctly. The "RESET" field can be generated either from a current induced magnetic field [8], or with a mechanical stress induced by the strain of a piezoelectric substrate [9][10] (Magnetoelastic NML). The use of a clock mechanism leads to an intrinsic pipelined behavior with delays depending on the circuits layout [11].

Due to the intrinsic memory ability of NML technology, the LIM architecture appears as an ideal target for this technology. The regularity of LIM architecture, coupled with the local memory, leads also to a low interconnection overhead. Wasted area for interconnections is one of the most serious problems of NML technology [12]. It is therefore important to design architectures that limit interconnection wires length. For the NML implementation we choose the Magnetoelastic NML [10], which is the NML subtype with the lowest power consumption. The working principle of this technology is outlined in Fig. 2.B. Magnets are deposited on a piezoelectric substrate. A voltage is applied to two electrodes placed beside the magnets. The electric field generated induces a strain in the substrate. The mechanical stress on the above magnets forces a magnetization rotation, driving the magnets in the RESET state. The set of logic gates available is highlighted in Fig. 2.C. The main logic functions are implemented using AND, OR and inverters [13], while a crosswire block [8] is required to cross two wires on the same plane, since NML is a single layer technology. We have envisioned the LIM architecture as a multi-dimensional structure, where logic, memory and interconnections are placed on different planes. This is however a logical structure, the physical implementation depends on the constraints of the target technology. In the Magnetoelastic NML case, the entire structure is therefore mapped on a single physical layer. As a consequence, the effectiveness of LIM architecture mapped on NML is for now limited. With further technological developments, however, we will be able to fully exploit the potential of LIM circuits.

III. The Logic-In-Memory Architecture

The LIM architecture is composed by an array of cells, each one containing a plane dedicated to logic, routing and memory (Fig. 2.A). The division in planes is from the logical point of view. The planes can be physically located in the same layer or in different layers, the choice depends on the technology. The cells can operate independently one from the

other. Each cell is connected only to the adjacent ones through local connections, completely avoiding global wires, therefore improving considerably the routing of the signals, with all the advantages in terms of area, power and speed. While most of the other architectural models, like NoC, consists of separated logic, memory and network elements, LIM integrates the three aspects in each cell. Computational power is proportional to the number of parallel elements. Cells should be small to increase the ratio between the number of cells and area.

The routing plane connects the logic plane to the memory plane of the cell. Moreover, it allows inter-cell communication among routing planes of adjacent cells. It receives memory access requests from the logic plane and from the routing planes of neighbor cells, delivering them to the final destination. The logic plane is the part of the cell devoted to the execution of the algorithm to be implemented in the LIM array. All its accesses to memory must pass through the routing plane. The memory plane represents the local memory of each cell, accessible only through the routing plane. The physical proximity of logic and memory allows a very fast access. Inter-cell and inter-plane communication is divided in packets, which may have different lengths depending on the type of operation involved. The list of possible operations is the following: *Local Write (LW), Local Read (LR), Toc Toc Write (TTW), Toc Toc Read (TTR), Remote Write (RW), Remote Read (RR).*

Local Write (LW) and Local Read (LR) are the fastest ones because there is no interactions with other cells, signals travel only through the routing plane of the same cell. On the other hand, Toc Toc access allows read/write operations in a memory location of neighbor cells. Signals travel through the logic and routing plane of the source cell and the routing and memory plane of the destination cell. Therefore the execution time of TTW e TTR operations is two times the execution time of local operations. Finally, Remote Write (RW) and Remote Read (RR) enable communication among any arbitrary couple of cells in the grid. They overcomes the limitations of local accesses to memory, but at the cost of reduced performance. Indeed, the delay of a Remote Write is N times bigger than local operations, where N is the distance between the target and the source cell. A Remote Read is instead 2N times slower, since the read data must travel back to the origin cell.

The LIM structure is described using a parametric VHDL code, that allows the flexibility necessary to implement a wide range of algorithms. The most relevant parameters are grid dimension, cell memory size and data width. The logic plane is the only one requiring a custom design for each implemented algorithm. Conversely, routing and memory planes provide communication and memory services that are intrinsically part of the architecture, not peculiar to the algorithm. Therefore, they can be shared among the implementations. This design reduces development time and effort.

IV. Application example: Summed Area Table

The test application for the LIM architecture is the "Summed area table" calculation algorithm. First introduced by Crow [14] in computer graphics domain in 1984, it became popular after the Viola-Jones object detection algorithm [15] included it as first step, renaming it integral image. The algorithm provides a fast way to detect features in two-dimensional rectangular images. At the beginning, the entire image is loaded in the LIM memory plane. Each row of the image is mapped in a cell and consecutive rows are placed in adjacent cells. The image becomes an array of rows that is then compacted in a grid in a snake-like configuration, as in Fig. 3, to optimize circuit layout. The number of cells is therefore equal to the height of the image.

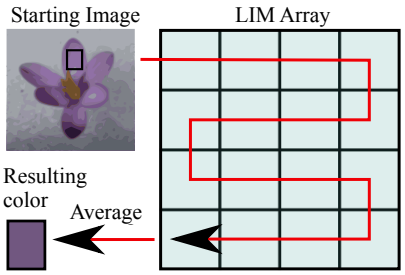

Fig. 3. Summed Area Table application example. Each cell stores an entire column. The array is linear but is arranged in a matrix structure to optimize the layout.

The computation of the Summed Area Table algorithm is organized in two phases. First, the pixels' values are added column by column. Secondly, the results are added row by row, as in [16]. Since every column is in a different cell in the array, during the first phase cells cannot work in parallel. The second phase can be executed completely in parallel, because an entire row is stored inside a cell. Resource utilization and performance reach therefore their maximum in this second phase. The locality of memory accesses, the amount of parallelism, the high memory bandwidth are all motivations that make this architecture suitable for the implementation of the Summed Area Table calculation.

V. Results

A. Results based on CMOS technology

In order to have an initial performance estimation, a preliminary synthesis on a single LIM cell has been run with a 28nm CMOS technology library, using Design Compiler Synopsys software. Memory has been synthesized using registers to achieve the maximum possible clock frequency. The area and power consumption of a single cell are $83296\,\mu m^2$ and $119\,mW$, respectively. The obtained clock frequency is $1.6\,GHz$. The total execution time of the algorithm was evaluated and compared with results presented in literature. The comparison is shown in Table I.

The gain in speed, though the structure is not optimized, is very high, highlighting the potential of this architecture in terms of timing. Delays due to traffic load and resource conflicts are already considered in the timing results. Area and power can be reduced using SRAM cells for the memory instead of registers, increasing however the execution time. In the future, we plan to embed dedicated memories in the

TABLE I
COMPARISONS WITH THE RESULTS OF PREVIOUS WORKS ON SUMMED AREA TABLE COMPUTATION

Reference	Image size	Time [ms]
Zhang[16]	640x480	0.19
This work	**640x480**	**0.051**
Huang-CPU[17]	1024x1024	7.96
Huang-GPU[17]	1024x1024	2.53
Bilgic[18]	1024x1024	2
This work	**1024x1024**	**0.096**

architecture, analyzing the trade-off between area/power and speed.

B. Results based on NML

We implemented the whole LIM architecture using Magnetoelastic NML technology. The schematic of the core part, one memory cell of the memory array, is depicted in Fig. 2.D. The design uses shared bus lines for control signals and input/output data, controlled by the ROW signal, which is the output of the selection decoder. This design was chosen because it allows to greatly reduce interconnections overhead. The memory cell layout can be seen in Fig. 2.E. It is very compact and modular: The memory array can be built simply assembling the desired number of memory cells together.

The entire circuit was modeled and simulated with an RTL model, described with VHDL language [19]. The circuit works with a frequency of $100\,MHz$. The total area of a single LIM cell is $227000\,\mu m^2$, nearly three times the area of the CMOS cell. The power consumption of one cell is $4.8\,mW$, a value much lower than the $119\,mW$ obtained with CMOS technology. However, reducing the operating frequency of the CMOS implementation, the power consumption obtained by the NML implementation is slightly higher than the CMOS one. While this is not the expected result, it can be easily explained. The LIM architecture is conceived as a three layer structure, one for logic, one for routing and one for memory. Since, for now, multilayer structures are not possible in NML, this leads to a sub-optimal result. Moreover the present LIM architecture represents a preliminary implementation of the logic-in-memory principle. It was not thought specifically for NML technology and its technological constraints. Finally, while NML technology has intrinsic memory ability, the clock mechanism negate this advantage. Magnets are periodically forced in the RESET state, eliminating therefore the memory mechanism. The most simple way to exploit the intrinsic memory ability of this technology is to embed Magnetic RAMS inside NML circuits. Doing this will result in a considerable reduction in both area and power consumption.

VI. CONCLUSIONS

We have successfully designed a Logic-In-Memory architecture, that leads to considerable advantages, whenever the access to memory is a critical factor. The circuit was synthesized on a 28nm CMOS technology, using as a benchmark the Summed Area Table algorithm. Results show that this architecture provides a considerable advantage over other hardware implementations presented in literature. We have also presented the first results of the NML implementation of the LIM architecture, which are encouraging.

We are now focusing on the NML implementation, trying to exploit the intrinsic memory ability of this technology. We plan to reach this result in two ways. First, redesigning the architecture considering the NML layout constraints. Secondly, integrating Magnetic RAMS inside the circuit.

REFERENCES

[1] D. Rairigh, "Limits of CMOS Technology scaling and technologies beyond-CMOS," 2005.

[2] H. Kung, C. Leiserson, and C.-M. U. D. of Comput. Science, *Systolic Arrays for VLSI*, ser. CMU-CS. Carnegie-Mellon University, Department of Comput. Science, 1978.

[3] W. H. Kautz, "Cellular logic-in-memory arrays," *IEEE Trans. Comput.*, vol. C-18, no. 8, pp. 719–727, Aug. 1969.

[4] "Int. Technol. Roadmap for Semiconductors (ITRS)," http://www.itrs.net, 2007.

[5] C. Lent, P. Tougaw, W. Porod, and G. Bernstein, "Quantum cellular automata," *Nanotechnology*, vol. 4, pp. 49–57, 1993.

[6] C. Augustine, X. Fong, B. Behin-Aein, and K. Roy, "Ultra-Low Power Nano-Magnet Based Computing: A System-Level Perspective," *IEEE Trans. Nanotechnol.*, vol. 10, no. 4, pp. 778–788, 2011.

[7] A. Imre, L. Ji, G. Csaba, A. Orlov, G. Bernstein, and W. Porod, "Magnetic logic devices based on field-coupled nanomagnets," in *Int. Symp. Semiconductor Device Research*, Dec. 2005.

[8] M. Niemier and al., "Nanomagnet logic: progress toward system-level integration," *J. Phys.: Condens. Matter*, vol. 23, Nov. 2011.

[9] M. Vacca, M. Graziano, A. Chiolerio, A. Lamberti, M. Laurenti, D. Balma, E. Enrico, F. Celegato, P. Tiberto, L. Boarino, and M. Zamboni, "Electric Clock for NanoMagnet Logic Circuits," in *Field-Coupled Nanocomputing*, ser. Lecture Notes in Computer Science, N. G. Anderson and S. Bhanja, Eds. Springer Berlin Heidelberg, 2014, pp. 73–110.

[10] M. Vacca, M. Graziano, L. Di Crescenzo, A. Chiolerio, A. Lamberti, D. Balma, G. Canavese, F. Celegato, E. Enrico, P. Tiberto, L. Boarino, and M. Zamboni, "Magnetoelastic Clock System for Nanomagnet Logic," *IEEE Trans. Nanotechnol.*, vol. 13, no. 5, pp. 963–973, Sep. 2014.

[11] M. Vacca, J. Wang, M. Graziano, and M. Zamboni, "Feedbacks in QCA: a Quantitative Approach," *IEEE Trans. Very Large Scale Intgr. (VLSI) Syst.*, vol. In publishing, 2014.

[12] M. Awais, M. Vacca, M. Graziano, M. R. Roch, and G. Masera, "Quantum dot Cellular Automata Check Node Implementation for LDPC Decoders," *IEEE Trans. Nanotechnol.*, vol. 12, no. 3, 2013.

[13] M. Vacca, M. Graziano, J. Wang, F. Cairo, G. Causapruno, G. Urgese, A. Biroli, and M. Zamboni, "NanoMagnet Logic: An Architectural Level Overview," in *Field-Coupled Nanocomputing*, ser. Lecture Notes in Computer Science, N. G. Anderson and S. Bhanja, Eds. Springer Berlin Heidelberg, 2014, pp. 223–256.

[14] F. C. Crow, "Summed-area tables for texture mapping," *SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 207–212, Jan. 1984.

[15] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2001, pp. 511–518.

[16] Y. Zhang, S. Yin, P. Ouyang, L. Liu, and S. Wei, "A parallel hardware architecture for fast integral image computing," in *IEEE Int. Symp. Circuits and Systems (ISCAS)*, June 2014, pp. 2189–2192.

[17] W. Huang, L.-D. Wu, and Y.-G. Zhang, "GPU-Based Computation of the Integral Image," in *Int. Conf. Virtual Reality and Visualization (ICVRV)*, Nov 2011, pp. 243–246.

[18] B. Bilgic, B. Horn, and I. Masaki, "Efficient integral image computation on the GPU," in *IEEE Intelligent Vehicles Symp. (IV)*, June 2010, pp. 528–533.

[19] D. Giri, M. Vacca, G. Causapruno, W. Rao, M. Graziano, and M. Zamboni, "A standard cell approach for MagnetoElastic NML circuits," in *EEE/ACM Int. Symp. Nanoscale Architectures (NANOARCH)*, Jul. 2014, pp. 65–70.