

Logic-In-Memory: A NanoMagnet Logic Implementation

Original

Logic-In-Memory: A NanoMagnet Logic Implementation / M., C., Santoro, G., Vacca, M., D., P., Causapruno, G., Cairo, F., Riente, F., Turvani, G., RUO ROCH, M., Graziano, M., Zamboni, M.. - ELETTRONICO. - (2015), pp. 286-291. (ISVLSI Montpellier July 2015) [10.1109/ISVLSI.2015.121].

Availability:

This version is available at: 11583/2616680 since: 2016-04-03T03:14:51Z

Publisher:

IEEE

Published

DOI:10.1109/ISVLSI.2015.121

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

Logic-In-Memory: A NanoMagnet Logic Implementation

M. Cofano, G. Santoro, M. Vacca, D. Pala, G. Causapruno,
F. Cairo, F. Riente, G. Turvani and M. Zamboni
Politecnico di Torino, Dep. of Electronics and Telecommunications

M. Graziano
London Centre for Nanotechnology, UCL

Abstract—In most computational systems memory access represents a relevant bottleneck for circuits performance. The execution speed of algorithms is severely limited by memory access time. An emerging technology like NanoMagnet Logic (NML), where its magnetic nature leads to an intrinsic memory ability, represents therefore a very promising opportunity to solve this issue. NanoMagnet Logic is the ideal candidate to implement the so called Logic-In-Memory (LIM) architecture. But how is it possible to organize an architecture where logic and memory are mixed and not separated entities?

In this paper we try to address this issue presenting our recent developments on LIM architectures. We originally conceived a LIM architecture without considering any technological constraints. Here we present the first adaptation of that architecture to NanoMagnet Logic technology. The architecture is based on an array of identical cells developed on three virtual layers, one for logic, one for memory and one for information routing. These three virtual layers are mapped on two physical layers exploiting all our recent improvements on NanoMagnet Logic technology, which are validated with the help of low level simulations. The structure has been tested implementing two different algorithms, a sort algorithm and an image manipulation algorithm. A complete characterization in terms of area and power is reported. The structure here presented is therefore the first step of an ongoing effort directed toward the development of truly innovative architectures.

Index Terms—NanoMagnet Logic, Logic-In-Memory, Parallel Computing, Multi-Layer Circuits.

I. INTRODUCTION

In the last decades CMOS technology has undergone a considerable evolution. Circuits have become incredibly small and fast, allowing to fit an increasing number of functionalities in a single chip. Most of this growth was due to the scaling process, with transistors constantly shrinking in size. The advantages provided by transistors scaling has shadowed other problems that affect VLSI circuits and that are not improved by the scaling process itself. A significant problem of modern computational systems is in fact the “Memory Wall” problem [1], the huge performance loss due to memory access times. Currently a processing unit can handle an enormous amount of operations, but memories are not fast enough to provide them the required data. As a consequence system performance are limited by memories. The scaling process makes things worse, because the performance gap between logic and memory further increases reducing transistor size. To attenuate this problem, current computational systems use caching systems and hierarchical memories [2]. However, this is not enough,

and a change in computational paradigm to overcome Von Neumann limitations is required. This is a particularly favorable moment of history to work on this topic, because transistors scaling is ending and new emerging technologies are being developed. The scaling process end means that the focal point of future developments is shifting from device to architecture. The emerging of new technologies, particularly magnetic-based technologies like NanoMagnet Logic (NML) [3], means that new features, impossible to achieve with CMOS transistors, can now be integrated inside VLSI circuits.

This work represents the sum of our recent efforts in trying to reach the goal of a “true” logic-in-memory (LIM) circuit. We have developed an architecture made by a big array of computational units. Each unit is made by three “virtual” layers, a logic core, a memory core and a routing plane. Since memory is locally embedded in each processing unit, this architecture provides several benefits in case of parallel algorithms with a strong local interaction among neighbor elements. The technology of choice is NanoMagnet Logic. The LIM architecture here presented does not fully exploit the potential of a magnetic technology, however we describe several enhancement that we have made toward this application. The background on NML logic and our improvements are described in Section II. The general structure of LIM architecture is instead described in Section III. To test the effectiveness of our idea we have adapted and implemented two algorithms on the LIM structure, the odd-even sort and the binomial filter, both highlighted in Section III. Finally Section IV describes the NML implementation. The circuit is designed and simulated using an RTL model. At the same time our advancement on NML technology allow us to design circuits using two physical layers, instead of planar circuits as normally happens in NML technology. The architecture is analyzed in terms of power and area. We believe that the solution here proposed represents a promising new beginning for architectures development, leading the research on circuits for computation on unexplored paths.

II. BACKGROUND ON NML

NanoMagnet Logic is an emerging technology that uses single domain nanomagnets with only two stable states to represent logic values (Figure 1.A) [4]. The innovative key point, compared to transistors based technologies, is based on a signal propagation that does not rely on voltages or currents.

Signals propagate through the circuit thanks to magnetodynamic coupling among neighbor magnets [3]. A NML wire is therefore composed simply by chaining the desired number of magnets (Figure 1.A) [5]. The advantages of this technology are related to its magnetic nature. A magnet maintains its state also without an applied power supply. The consequence is that in NML logic a magnet can act as a logic or memory element. Moreover it has no static power consumption and a potential very low dynamic power absorption [6]. The main characteristics of NML technology is the need of an external mean to switch magnets from one stable state to the other [7]. Magnets are forced in an intermediate unstable state, allowing them to overcome the energy barrier and switch to a different state depending on the value of neighbor magnets [8]. This system is called clock, because it gives to the technology a synchronous behavior similar to clocked CMOS circuits. Several techniques can be used to force magnets in the RESET state, like an external magnetic field [7] or the Spin Hall Effect [9], a spin-torque coupling with a current flowing through the magnet itself [10] or a mechanical stress applied through a voltage and piezoresistive substrate [6]. Regardless of the mechanism used a multiphase clock system must be used [11]. Circuits are divided in areas composed by a limited number of magnets called clock zones. At each clock zone a different clock signal is applied. The multiphase clock system allows to safely propagate signals in NML circuits in presence of thermal noise [12].

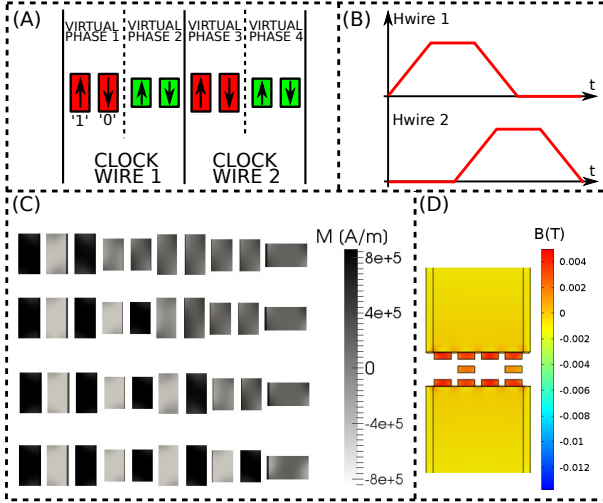


Fig. 1. NML fundamentals. A) Single domain nanomagnets encode the information. Virtual clock zones can be created with magnets of different size. B) Clock signals applied to the circuit. C) Micromagnetic simulation of a NML wire with virtual clock phases. D) Comsol simulation of 3D NML structure, with copper wires placed above and below magnets.

Among the clock solutions only the magnetic field [3] and the Spin Hall Effect [9] clock where experimentally demonstrated. Both shares the same technological frame, where the clock is generated by a current flowing through a wire placed under the magnets plane. The difference among these two clocks lies in the current value required and the material used for the wires. More details will be provided in Section IV.

We base our LIM design on these two clock solutions. In this paper we present a major improvement to NML clock, the virtual clock system. The working principle is depicted in Figure 1.A. Magnets with a bigger aspect ratio require a higher magnetic field to be reset. As a consequence we have built a wire chaining magnets with different aspect ratios ($50 \times 100 \text{ nm}^2$ and 50×80^2 in Figure 1.A). Two physical wires are used to generate the clock field, the waveforms applied are depicted in Figure 1.B). This particular solution allows us to use only two wires to create four “virtual” clock phases. Each virtual phase correspond to magnets with a different aspect ratio (Figure 1.A). In Figure 1.C the low level simulation obtained with NMAG [13] software is reported. Magnets start to switch from left to right, the two bigger magnets in the first virtual clock phase and then the two smaller magnets in the second virtual clock phase. Following the waveforms of Figure 1.B, the next magnets in the chain that switch are the bigger magnets corresponding to the third virtual clock phase and then the last two small magnets of the fourth virtual clock phase.

The complete explanation and validation of this clock system is out of the scope of this paper. However we commonly employ it in our designs because it provides several benefits to NML circuits. It simplifies the clock generation network, only two clock wires are required to generate the four clock phases necessary to assure a correct signals propagation. It enables to safely use the majority voter, the basic logic gate of this technology, also considering the constraints related to the fabrication of clock wire (more details are in Section IV). It allows us to build magnetic interconnections in NML circuits using domain walls, as we demonstrated in [14], greatly reducing interconnections overhead. Finally, placing clock wires above and under the magnets, as suggested in [3], it is possible to build multilayer NML circuits. In this case we limit our design to two logic layers, with a third separation layer between them. Figure 1.D shows a Comsol Multiphysics [15] simulation of such structure. The section view depicts two clock wires of 400nm thickness and three layer of magnets. The current value used is 3mA and the magnetic flux density varies in the range of 2-3mT, depending on the layer distance. This value of magnetic flux density is sufficient to reset magnets. As will be described in Section IV, the possibility of using two logic layers allow us to greatly reduce area thanks to lower interconnection overhead.

III. THE LOGIC-IN-MEMORY STRUCTURE

The *Logic-In-Memory* can be seen as a smart memory where data are not only stored but also elaborated. The basic element (*cell*, Figure 2.A) embeds memory and logic with an information routing component. The architecture has a matrix-like structure (*grid*, Figure 2.A) where each element works autonomously in parallel.

The cell is composed by three “virtual” layers, memory, logic and routing. With the definition “virtual” we mean that these three layers can be mapped on different physical layers, depending on the possibility offered by the target technology.

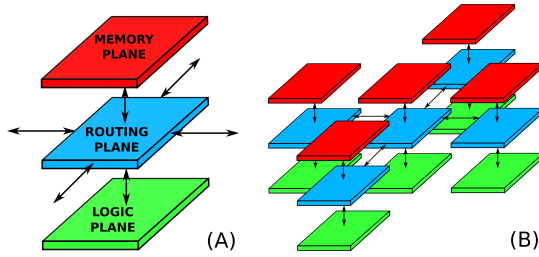


Fig. 2. (A) Logic In Memory (LIM) basic cell, three virtual layers are used, a memory plane, a logic plane and the routing plane. (B) Example of cells connection. Every LIM cell is connected with neighbor cells on 4 sides through the routing plane.

The *Logic Plane* hosts all the computational blocks required for the algorithm execution. It is a unit that must be designed ad-hoc for each implemented algorithm. The *Memory Plane* stores memory cells that can be read and written with simple operations. The number of memory locations depends on the algorithm implemented. In the case of the algorithms described in this paper, few memory locations are needed, making the memory plane size negligible compared to the other planes. The *Routing Plane* handles the communication and data exchange with neighbor cells. The routing plane is the same for all applications, while the memory plane size and the logic plane structures depends on the algorithms implemented.

Information are exchanged between cells through messages, called *words*. Each *word* is a complex bit string with many fields used to identify important information, like the operation type and cells address. Each *word* is composed by the following fields.

- TAG: it identifies the operation type.
- TCA (Target Cell Address): it is the address of the message origin cell inside the grid.
- WA (Word Address): it specifies the memory location to access;
- DATA: it is the data to be computed or stored.
- DEST: it represents the message destination cell address inside the grid

The instruction set of each is composed by several operations.

- Local write/read: local operations to allow the logic to write or read data from the memory.
- Toc-toc write/read: they allow the communication among two different cells; each cell can write or read a data of adjacent cells.
- Remote write/read: remote operations are the slowest ones and are used to allow the communication between non adjacent cells.
- Logic-logic: these operations allows to the logic planes of two adjacent cells to directly exchange data without accessing their memory planes.

This kind of structure is particularly suited to execute algorithms where a high number of memory accesses have to be performed, since the logic is embedded in the memory and therefore the access is less expensive. Moreover, since

each cell is connected to the adjacent ones by means of local interconnections avoiding global wires, the LIM architecture is particularly adapted to NML technology where interconnections overhead wastes a lot of area. In the following the routing plane and the logic planes for the two algorithms are described. The description of the memory plane is not reported because it is made simply by an array of memory cells.

A. The routing plane

The block diagram of the routing plane is depicted in Figure 3. To simplify the architecture, the routing plane has been designed with only one FSM. It manages the input requests without solving conflicts in case of simultaneous request coming from different cells. To solve this issue a priority management block has been created. The highest priority has been given to the logic and then, in descending order, to the requests coming from the north, west, south and east cell respectively. If one or more requests cannot be satisfied, an acknowledge bit is sent to the request-sender. The key role of the routing plane is to discriminate the operation type and to identify the sender and the receiver. Neighbor cells are identified by two bits (from 00 of the east cell to 11 of the north cell in clockwise order). This two bits represent a sort of relative address, that each cell uses to identify its neighbors. In conjunction to the relative address, an absolute address system is used to identify each cell. A number indicates the cell position in terms of column and row. The routing plane can be divided in four main regions. 1) The input interface is responsible for sinking all requests and managing their priorities. 2) The priority manager checks if the FSM is available to execute an operation. In the affirmative case it selects the request with the highest priority and sends it to the selection unit. If a request cannot be satisfied, acknowledge bits are sent to the senders. 3) The selection unit identifies the operation to be executed. 4) The last region is the output interface. It is responsible of sending the data to the correct destination.

B. The odd-even sort logic plane

As a first case of study the logic plane has been designed to execute the odd-even sort algorithm. It is a comparison sorting algorithm where every even and odd couple of numbers are compared. If numbers are not in the exact order, they are swapped. The main advantage of this algorithm is its ability to simultaneously compare all even or odd couples of numbers exploiting the parallel computation. The logic plane for this application (depicted in Figure 4) is composed by registers, a FSM, an adder to compare the value of the two data and two counters: one to switch from the operating mode to the stand-by mode and one to stop the execution.

Only even cells start the computation while the odd ones are in stand-by. After the algorithm execution, the even cells go in stand-by mode while the odd ones run the sorting. The process is repeated until all data are sorted. The pseudo-code of the algorithm, in order to implement it inside the LIM architecture, is reported in the following.

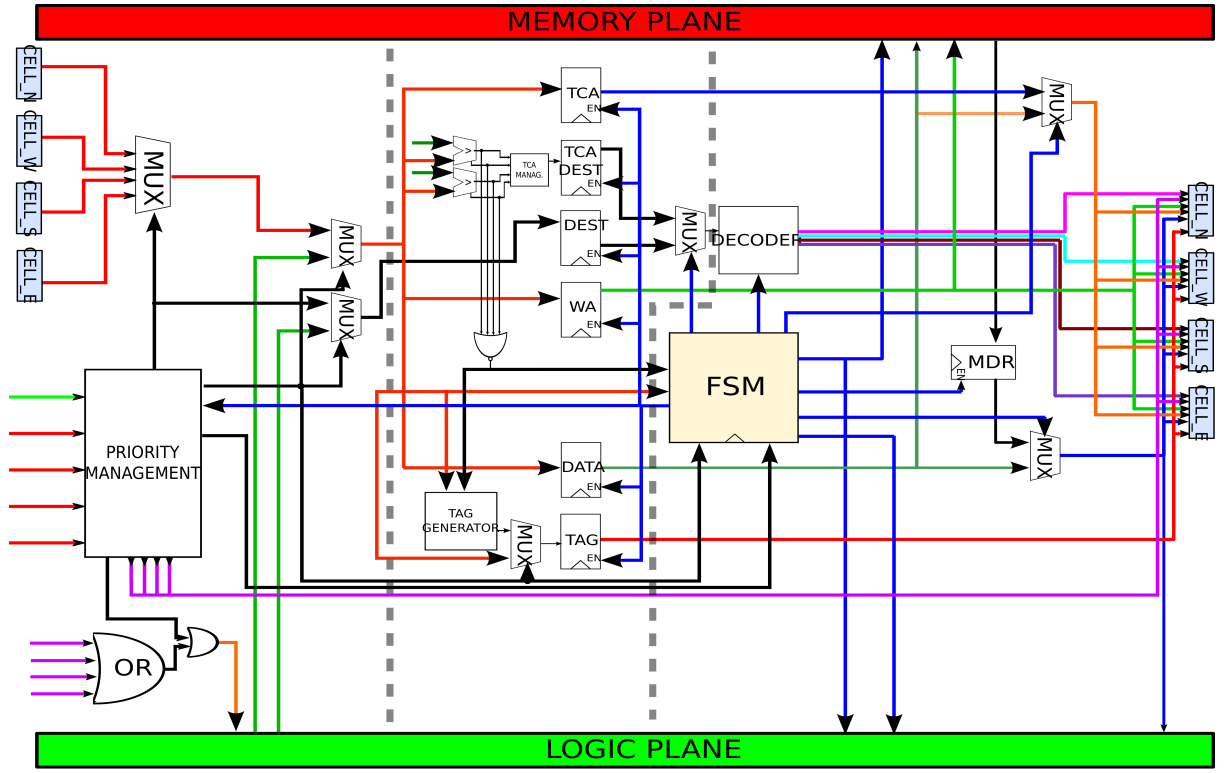


Fig. 3. Routing plane block diagram. Multiplexers, a priority management component and destination selection blocks are used to route signals to logic and memory planes and to neighbor cells. The plane behavior is controlled by a dedicated finite state machine (FSM).

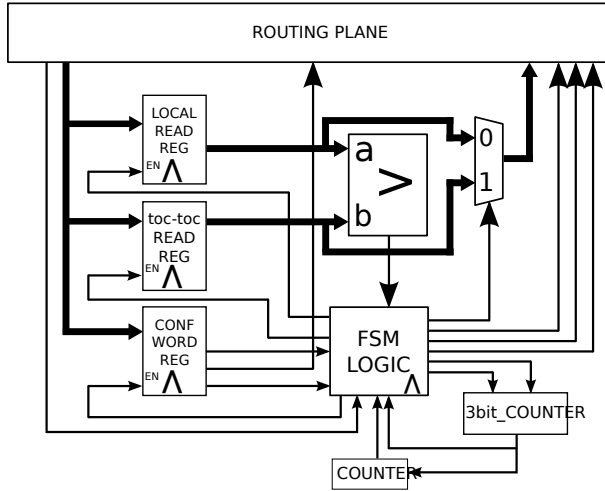


Fig. 4. Logic plane block diagram for the odd-even sort algorithm. The logic plane is composed by a comparator, a multiplexer, counters, registers and an FSM control unit.

- 1) Read the configuration word;
- 2) Read the local data;
- 3) Read the adjacent cell data;
- 4) Compare the two data;
- 5) Exchange data if necessary;
- 6) Stand-by until the execution restarts.

C. The binomial filter logic plane

As a second case of study a binomial filter, mostly used for signal processing, has been implemented. The logic plane (depicted in Figure 5) is composed of registers, a FSM and an adder to sum all read data.

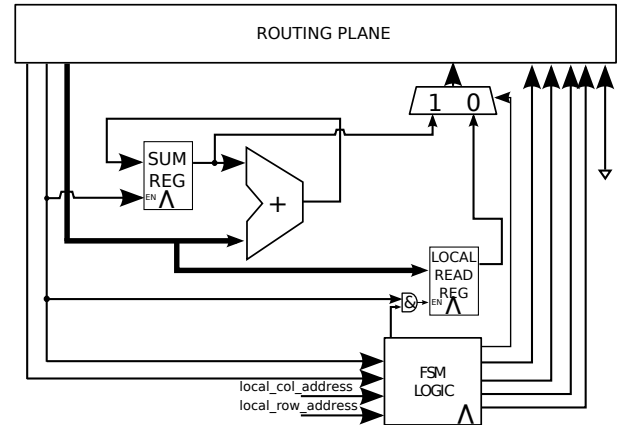


Fig. 5. Logic plane block diagram for binomial filter algorithm. The logic plane is composed by an adder, two registers, a multiplexer and an FSM control unit. The datapath is simpler compared to the odd-even sort algorithm.

The pseudo-code of the algorithm is reported in the following.

- 1) Read the local data;
- 2) Read the eight neighbouring cells data;

- 3) Sum all data and divide the result by 16;
- 4) Write the result in the local memory.

IV. THE LOGIC-IN-MEMORY ARCHITECTURE

The Logic-In-Memory architecture has been implemented using NML technology applying the enhancements described in Section I. To design and simulate the NML circuit, an RTL model has been developed [16]: as depicted in Figure 6, registers simulate the propagation delay of signals through consecutive clock zones, while ideal logic gates model the logic function. Majority voters and inverters are sufficient to represent every logic function. An inverter is obtained linking consecutively an odd number of magnets. AND/OR gates can be obtained from a majority voter fixing the polarization of one input to a constant logic '0' or logic '1', respectively. Figure 6

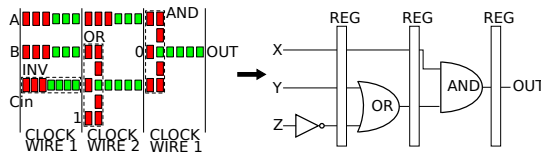


Fig. 6. RTL model of an NML circuit. Registers are used to model the propagation delay introduced by each clock zone whereas ideal gates are used to represent the logic function.

highlights the majority voter implementation using the virtual clock. The three input wires belong to the first virtual phase (bigger magnets). The central and output magnets belong to the second virtual phase (smaller magnets). As a consequence the central magnet switch only after all input signals have successfully propagate. This solution greatly improves the reliability of majority voters considering constraints related to clock wires fabrication [5].

The NML implementation of the Logic-In-Memory architecture is based on the use of two physical layers and virtual clocking. In Figure 7 is depicted, as an example, a 3D full adder. The adder develops horizontally on two physical layers (layers 1 and 3), thus, it can be thought of as a three-dimensional structure. Layer 1 and layer 3 are connected through an intermediate level on which only few magnets (those necessary to the communication between layers) are allowed. The role played by magnets on Layer 2 is the same as the one played VIAs in CMOS circuits. Thanks to virtual clocking and multilayer structures the circuit area greatly decreases. The full adder here presented is more than 4 times smaller than the full adder described in [14].

Figure 8 depicts instead the 3D layout of a ripple carry adder. The layout has been designed according to the following constraints: 1) clock wire length equal to 6 magnets and 2) maximum number of consecutive magnets belonging to the same virtual phase equal to 5 [12].

The LIM architecture is based on three virtual layers, but the technology allow us to use only two physical layers. As a consequence we found that the optimal solution was to map every plane independently on two physical layers. The cell and the array were then assembled by aligning the three virtual

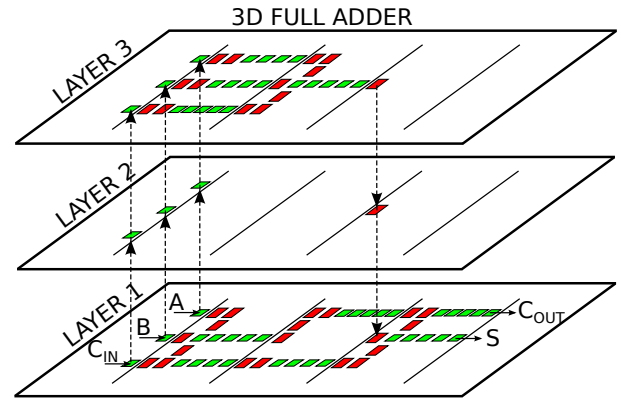


Fig. 7. Layout of a 3D full adder.

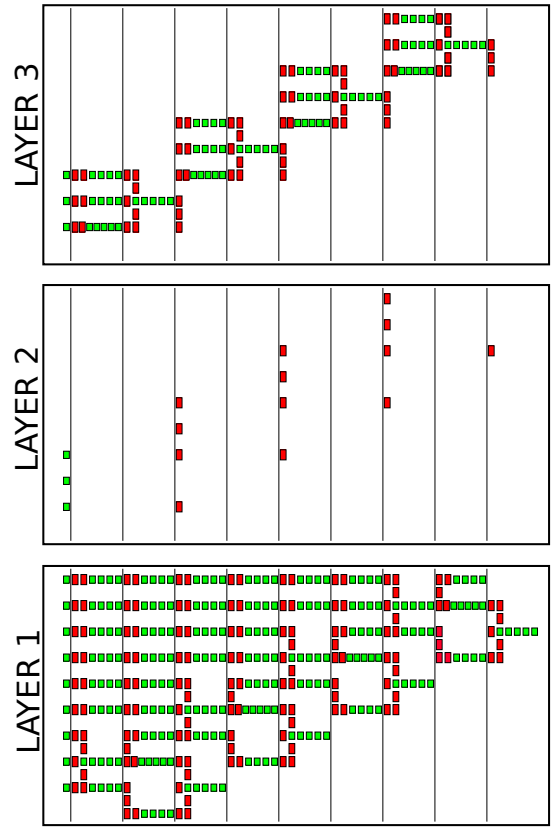


Fig. 8. NML layout implementation of a 4 bit ripple carry adder. Two physical layers and virtual clocking have been used.

planes. Figure 9 depicts the generic organization of a 4×4 grid of LIM cells with the block scheme of a cell outlined. The block scheme and the dimensions refer to a PE in which the logic plane implements the odd-even sort algorithm. It can be noticed that the routing plane occupies most of the area of the cell. This is mainly due to two reasons. First there is an high number of interconnections, that waste a considerable portion of area even if solutions (such as domain walls and two physical layers) to compact the layout are exploited. Secondly, an high parallelism is adopted, implying that a large number of NML wires travels in parallel and the area that they take is

not at all negligible. The area occupied by the “virtual” logic plane is comparable to the routing plane, while the memory is much smaller. Each processing element (PE) has an estimated height of $33.1 \mu\text{m}$ and a length of $66.4 \mu\text{m}$.

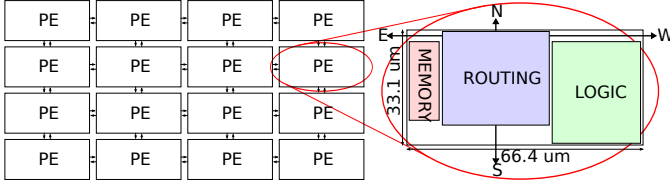


Fig. 9. 4×4 grid of LIM cells. The block scheme of each PE is outlined. Each “virtual” plane is mapped independently on two physical layers. The whole structure is then assembled by aligning the “virtual” planes together.

The area occupation and the power consumption have been estimated for both implementations. Results are summarized in TABLE I. The area was evaluated by knowing the total height and width in terms of number of magnets and considering the physical height and width of magnets as reported in Section II. Power dissipation was evaluated estimating the clock wires length starting from circuit area. More details on how to evaluate power consumption can be found in [16]. Power was evaluated considering the magnetic field clock [3], using copper wires of 400nm thickness and 420nm width and a current of 6 mA for each wire [3]. Power was also estimated in case of Spin Hall Effect clock, considering wires made by a complex heterostructure where the resistance value is dominated by the 10nm layer of Tantalum. In [9] authors use a current of 2mA for a clock wire with a width of $1\mu\text{m}$ and an height of 10nm . In this case we use wires with a width of 420nm and an height of 10nm , thus the current value that we use is 0.8mA . and a thickness equal to 400 nm .

TABLE I
AREA OCCUPATION AND POWER CONSUMPTION ESTIMATION OF A SINGLE CELL THAT EXECUTES, IN ONE CASE, THE ODD-EVEN SORT ALGORITHM AND, IN THE OTHER, THE BINOMIAL FILTER ONE.

Circuit	Area (μm^2)	Power (mW)	
		Magnetic field	Spin Hall Effect
Odd-even sort	2196.2	18.4	2.6
Binomial filter	2129.8	18.4	2.6

Considering the two algorithms the cell area occupation and power consumption is roughly the same. The two clock systems instead lead to the same circuits organization, therefore the area does not change considering one clock system instead of the other. Power consumption instead is quite different, being 7 times lower in the Spin Hall Effect clock. If further development of this solution will allow to use the Spin Hall Effect clock with lower resistance materials, power consumption will be further reduced. Regarding time, circuits works with a frequency of 100MHz , the instructions execution time varies from 79 to 131 clock cycles. This is mainly due

the intrinsic pipelined nature of this technology which reduces performance in case of feedback signals [11].

V. CONCLUSIONS

We have developed a logic-in-memory architecture to exploit the potential of emerging technologies. We have implemented the LIM architecture using NML technology as target, bringing, at the same time, several enhancements to the technology itself. Performance obtained are very good, especially in terms of area and power consumption. We are now working on technological solution that allow us to implement NML circuits with many layers. At the same time we are redesigning the architecture keeping in mind its pipelined behavior in order to improve circuit performance.

VI. ACKNOWLEDGEMENTS

We thank Alessandra Fioriti, Michele Giacobbe, Giuseppe Casuccio and Massimo Caligaris for their valuable contribution on this project.

REFERENCES

- [1] A. Nowatzky, P. Fong, and A. Saulsbury, “Missing the memory wall: The case for processor/memory integration,” *1996 23rd Annual International Symposium on Computer Architecture*, p. 90, May 1996.
- [2] P. Jacob, A. Zia, O. Erdogan, P. Belemjian, J. Kim, M. Chu, R. Kraft, J. McDonald, and K. Bernstein, “Mitigating memory wall effects in high-clock-rate and multicore cmos 3-d processor memory stacks,” *Proceedings of the IEEE*, vol. 97, no. 1, pp. 108–122, Jan 2009.
- [3] M. Niemier and al., “Nanomagnet logic: progress toward system-level integration,” *J. Phys.: Condens. Matter*, vol. 23, p. 34, Nov. 2011.
- [4] R. Cowburn and M. Welland, “Room temperature magnetic quantum cellular automata,” *Science*, vol. 287, pp. 1466–1468, 2000.
- [5] M. Vacca, D. Vighetti, M. Mascarino, L. Amaru, M. Graziano, and M. Zamboni, “Magnetic QCA Majority Voter Feasibility Analysis,” *2011 7th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pp. 229–232, 2011.
- [6] M. Vacca, M. Graziano, L. D. 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 Transaction On Nanotechnology*, vol. 13, no. 5, September 2014.
- [7] M. Niemier, X. Hu, M. Alam, G. Bernstein, M. P. W. Porod, and J. DeAngelis, “Clocking Structures and Power Analysis for nanomagnet-Based Logic Devices,” in *Int. Symp. on Low Power Electronics and Design*. Portland-Oregon, USA: IEEE, 2007, pp. 26–31.
- [8] M. Awais, M. Vacca, M. Graziano, and G. Masera, “FFT Implementation using QCA,” *2012 19th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 741–744, 2012.
- [9] D. Bhowmik, L. You, and S. Salahuddin, “Spin hall effect clocking of nanomagnetic logic without a magnetic field,” *Nature Nanotechnology Letter*, vol. 9, pp. 59–63, 2014.
- [10] J. Das, S. Alam, and S. Bhanja, “Ultra-low power hybrid cmos-magnetic logic architecture,” *Trans. on Computer And Systems*, 2011.
- [11] J. Wang, M. Vacca, M. Graziano, and M. Zamboni, “Biosequences analysis on NanoMagnet Logic,” *International Conference on IC Design and Technology*, pp. 131–134, May 2013.
- [12] G. Csaba and W. Porod, “Behavior of Nanomagnet Logic in the Presence of Thermal Noise,” in *International Workshop on Computational Electronics*. Pisa, Italy: IEEE, 2010, pp. 1–4.
- [13] T. Fischbacher, M. Franchin, G. Bordignon, and H. Fangohr, “A Systematic Approach to Multiphysics Extensions of Finite-Element-Based Micromagnetic Simulations: Nmag,” *IEEE Transactions on Magnetics*, vol. 43, no. 6, pp. Available on-line, 2007.
- [14] F. Cairo, M. Vacca, M. Graziano, and M. Zamboni, “Domain magnet logic (dml): A new approach to magnetic circuits,” in *IEEE International Conference on Nanotechnology*, 2014.
- [15] “Comsol Multiphysics,” <http://www.comsol.com/>.
- [16] M. Vacca, M. Graziano, and M. Zamboni, “Nanomagnetic Logic Microprocessor: Hierarchical Power Model,” *IEEE Transactions on VLSI Systems*, vol. 21, no. 8, pp. 1410–1420, Aug. 2012.