Reconfigurable Systolic Array: From Architecture to Physical Design for NML

Reconfigurable Systolic Array: From Architecture to Physical Design for NML

G. Causapruno, F. Riente, G. Turvani, M. Vacca, M. Ruo Roch, M. Graziano, *Member,* and M. Zamboni

***Abstract*—NanoMagnet Logic (NML) is among the emerging technologies that might replace CMOS in next decades. According to its physical characteristics, to better exploit the potential of this technology – and of other similar ones – the use of parallel architectures with regular layout that avoid long interconnection signals is advised. Systolic Arrays are among these architectures, being composed of a grid of equal Processing Elements locally interconnected. However, they are usually implemented to execute only a small set of algorithms, and for this reason throughout the years they have not been an appealing solution for CMOS.**

To seriously analyze the potentials of NML, complex architectures must be conceived, and their physical implementation explored taking into account realistic technological constraints. With the increasing complexity of NML circuits, two issues, then, are noticed: 1) The need for a regular structure arises, that at the same time helps to reduce the intrinsic pipelining nature of NML and can be configured to be used for several applications without developing a dedicated design for each algorithm. 2) The capability to synthesize, place and route NML circuits is fundamental to demonstrate the feasibility of the architecture in two important conditions: efficiently managing the complexity of the design and sticking to the characteristics that are technologically feasible at the time of writing. In this article we address these issues presenting a new Reconfigurable Systolic Array, that can be programmed to execute different algorithms, and we provide two examples to show its working principle. Moreover the Array is synthesized and simulated with the aid of the first real tool for nanotechnology circuits that we have conceived, ToPoliNano. The joint contribution at both architectural and physical design level gives a relevant step forward to the state of the art in the demonstration of this emerging technology potential.

***Index Terms*—NanoMagnet Logic, Parallel Architectures, CAD for nanocomputation**

I. Motivation

During last years the successful trend of CMOS scaling, predicted by Moore in 1965, has suddenly slowed down due to technological limitations such as the increasing leakage current and minimum fabrication sizes achievable [1]. Emerging technologies, that could replace CMOS in next several years, are currently under study. These technologies will be able to process data at an extremely high operating frequency [2] or with a significant reduction of consumed power [3]. The ITRS report [4] summarizes several possible technological solutions, among which carbon nano-tubes and graphene based devices, Quantum-dot Cellular Automata (QCA) and silicon nanowire based nanoarrays. Our attention is focused on Quantum-dot Cellular Automata since different studies envisage this technology as a promising alternative to CMOS [5]. In particular we consider in this article NanoMagnet Logic (NML) implementation (Fig. 1.A), which has been already experimentally demonstrated [6] and is then an excellent case study to explore how emerging technologies can find their way into the future with the accompanying cumbersome inheritance of the CMOS success story.

Authors are with the Department of Electronics and Telecommunications, Politecnico di Torino, TO, I10129 Italy e-mail: mariagrazia.graziano@polito.it. M. Graziano is also with the London Center for Nanotechnology (UCL), m.graziano@ucl.ac.uk

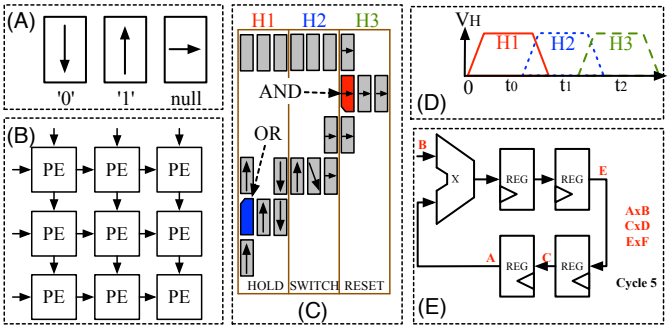

Fig. 1. **(A)** (NML) principle: magnetization represents logic values. **(B)** Systolic Array structure. **(C)** An example of signal propagation in a NML circuit. The circuit is split in 3 different clock zones. Nanomagnets with one cut corner implement logic functions AND, OR. **(D)** Clocking scheme for magnetic QCA circuits. **(E)** Pipeline interleaving technique: input data are interleaved to execute 3 operations exploiting the feedback cue.

One important aspect to be underlined, and valid for all the emerging technologies conceived for computation, is the unavoidable need to tackle the design of circuits of a reasonable complexity. We strongly believe this is an essential point for two reasons. The first is that it is not at the device level that a realistic evaluation of the potential of a new technology can be really compared to CMOS performance and success history. The second relies on the fact that only with complex architectures real problems and needs are understood and proper countermeasures and directions at the architectural, the design and the technological levels can be found.

Having these motivating principles in mind we addressed our attention to *architectures* for nanocomputation of a certain complexity. We tried to analyze how architectural solutions can at the same time exploit these technologies' remarkable characteristics and solve critical aspects. These critical aspects are partially related to their intrinsic nature and partially due to the immaturity of technological processes. With the same motivation we do not stop at the architectural level, but also consider the feasibility of these architecture at the *physical level* by tackling their physical design down to the single device element. This physical design phase should then take into

account the technological characteristics and constraints and, at the same time, tackle the design of complex architectures to clearly demonstrate the feasibility of the design. Without this physical design step, that we consider fundamental, the architectural evaluation would not be reliable and its actual feasibility would not be demonstrated at the design phase. These two stages of the design for nanocomputation are at the basis of this work.

The rest of the paper is organized as follows: in section II we start from physical characteristics of NML to introduce a new architecture called Reconfigurable Systolic Array. This addresses the first point of our analysis: the identification of a complex architecture tailored for NML technology. The successive section III deals with the physical layout phase of the design flow: in this section we present ToPoliNano, the first existing tool able to tackle a complete top-down design flow for nanotechnologies, showing its working principle and main steps of the design flow. Section IV presents in detail the architecture of the Reconfigurable Systolic Array. The reconfigurability mechanism is described in section V. NML implementation with ToPoliNano is reported in section VI. Finally, conclusions are provided in section VII.

II. Background

In this section we focus on the first stage of the design flow for nanotechnologies, focusing on the design of an architecture able to exploit the remarkable characteristics of NanoMagnet Logic and at the same time addresses its critical aspects.

NML and the majority of other nanotechnologies are intrinsically pipelined, and the pipeline level is usually extremely high [5]. While in combinational circuits this is not a problem [7], in sequential circuits, where the result of one operation depends on previous ones, this represents a big issue [8]. Feedback signals, that require many clock cycles to propagate back, limit the throughput. Actually a new operation cannot be started at every clock cycle, and the input data rate is reduced according to the length of the longest loop in the circuit [9] [10]. To solve this problem and exploiting the deep pipeline of the circuit at its best, *pipeline interleaving* can be introduced: It requires providing several *unrelated* input sequences, so as to fill the pipeline queue present in the loop and maximize the throughput [8][11] (Fig. 1.E).

Due to this intrinsic pipelining proper architectures should be adopted: Global interconnections should be avoided, because their routing would heavily complicate the design and the delay of the circuit [7]; Regular structures that can be replicated should be preferred to simplify synchronization of signals across clock zones [12]. Systolic Arrays (SAs) [13] are among the architectures that meet the constraints. Indeed SAs are regular structures based on Processing Elements (PEs) locally interconnected (Fig. 1.B). This means that each PE can communicate with its neighbors. For these peculiar characteristics SAs have already been exploited for general QCA implementation [14] [15]. In this article we present a step further tailoring the design for NML technology, taking into account also physical constraints of this implementation. In the following paragraph we give an introduction to SAs.

Systolic Arrays

SAs were first introduced by Kung and Leiserson in 1978 [13], but they did not emerge as a valid architectural solution in past decades because designers have relied on performance improvements provided by technological scaling rather than on parallel solutions to increase the throughput. SAs are gaining great interest in the last years again because they represent an ideal structure for emerging nanotechnologies and their parallel nature is now exploited also in CMOS to increase throughput. Indeed, their structure is ideal for the so called “Embarrassingly Parallel Algorithms”, where several computations must be executed in parallel to achieve a reasonable throughput. SAs, given their highly parallel structure, are also an ideal target for the so called ”Logic-In-Memory” applications [16][17]. Logic-In-Memory architectures embed logic and memory in the same device; as a consequence SAs are ideal for this application since memory can be easily embedded in each processing element.

In CMOS, SAs are usually designed as dedicated co-processors to accelerate a given task. This has been done with ad-hoc designs that cannot be reused for other operations, i.e. they are algorithm-dependent. For this reason SAs have been adopted only for a small subset of problems requiring a high number of calculations: signal processing [18], video processing [19], biological sequence comparison [20][21]. In other cases they are mapped to FPGA[22], so that the hardware could be reused if a different algorithm must be implemented. The first case (ad-hoc ASIC) can be used only for a small set of algorithms, the second (implementation on FPGA) limits the operating frequency and the number of Processing Elements that can be mapped.

The solution to this disadvantageous duality can be found in Reconfigurable SAs, that can be reprogrammed to execute different algorithms. In this article we introduce our Reconfigurable Systolic Array (R_SA): each Processing Element can execute a different operation and can use previous results, input data or results of neighbor PEs as operands depending on the chosen configuration. In this way the same architecture can be used for several applications, making it more attractive for a real implementation, not only for nanotechnologies but also in CMOS.

Some reconfigurable SAs have been proposed in recent years: the work in [23] presents a reconfigurable architecture for VLSI implementation of BP neural networks with on-chip learning, while in [24] a general purpose architecture for the parallelization of nested loops in reconfigurable architectures is described. Both SAs propose reconfigurability to address a specific problem’s scale. A fully reconfigurable architecture has been proposed in [25], with a systematic design approach to map two or more algorithms into a single SA. This study however lacks real data and physical implementation, and it has been conceived for CMOS implementation only. In this article we have developed our R_SA with particular attention to exploit as much as possible the positive features of emerging nanotechnologies.

NanoMagnet Logic

Quantum-dot Cellular Automata (QCA) is an emerging technology of particular interest for its high frequencies achievable, low energy and small area requirements [26].

Literature presents two main implementations of the QCA principle: Molecular QCA, where molecules are used as the base element, which is studied for the high theoretical clock frequency that can reach (1 THz) [27][28][29] and Nano-Magnet Logic (NML), where single domain nanomagnets are used as the base element [30][31], which is interesting for its low power consumption [32]. In this article we consider NML as a reference technology, as it is the nearest to a real implementation and remarkable experimental proofs have been already given for its feasibility. Information on the physical results and procedures also help introducing more reliable and realistic constraints and data when a physical design phase is attempted.

NML technology is based on nanomagnets whose magnetization represents an encoding binary value (Fig. 1.A). Nanomagnets align their state according to neighbor elements, through magnetic interaction, and in this way they propagate 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 HOLD state, magnets maintain their polarization hence storing the contained information. In SWITCH state, magnets change their polarization according to neighbor magnets in the HOLD state. These magnets are not influenced by the successive ones that are placed in an intermediate unstable state called RESET. To generate this clock mechanism, it is necessary to force cells in the RESET state in the unstable state through a current induced magnetic field [33], current induced spin-torque mechanism [34], or with a mechanical stress induced by the strain of a piezoelectric substrate [3] (Magnetoelastic NML) exploiting the magnetoelastic effect [35]. With micromagnetic simulations it has been proved that this technology can work at 100 MHz [36].

The use of a clock leads to an intrinsic pipelined behavior: Every group of 3 consecutive clock zones has a delay of 1 clock cycle. The pipeline level is hence intrinsically related to the technology and it depends on the circuit layout. This is the main difference with respect to CMOS circuits. Usually this pipeline level is extremely "deep" (hundreds of clock cycles). The intrinsic pipelining leads to the "layout=timing" problem. The delay propagation of a signal depends on the length of its correspondent wire. If at the input of a logic circuit the wires length is not perfectly matched, errors will occur due to bad synchronization. This problem can be solved both using asynchronous delay-insensitive logic [9] or with a proper clock zones layout [8]. Moreover, in case of sequential circuits the intrinsic pipelining leads to a reduction of throughput of N times, where N is the length in clock cycles of the longest loop in the circuit [9]. This problem can be solved applying *pipeline interleaving*[11][20], if several unrelated input sequences are available (Fig. 1.E).

As mentioned in the first section, the pressing need is to conceive and study complex QCA architectures in order for them to be comparable to CMOS as well as to reveal the real needs and advantages of these emerging technologies. With QCA circuits that are growing in complexity, two main issues can be noticed: 1) there is the need for a regular structure that can be used for several application and does not require a dedicated design; 2) a tool able to synthesize, place and route QCA circuits is fundamental to manage efficiently the complexity of the design. We address these two issues presenting our reconfigurable architecture, synthesized and simulated with the aid of the first tool for real nanotech circuits that we have conceived and presented in a first form in [37][38], here improved to manage the complexity of the architecture proposed and to adhere to the emerging trends on clock distribution type.

The R_SA structure satisfies our first goal of working with complex architectures that exploit emerging technologies' characteristics and solve their critical problems, so our overall goal can be met by implementing the R_SA physical design in NML technology. This is done using ToPoliNano, the first existing tool able to tackle a complete top-down design flow for nanotechnologies [37][38]. In particular ToPoliNano is able to synthesize, place, route [39] and simulate [40] [41] NanoMagnet Logic circuits, and evaluate metrics such as the area occupation and the dissipated power (Fig. 2). For this paper, ToPoliNano was enriched by essential new algorithms at the place and route level, that, with respect to previous versions of the tool [38], can manage big combinational circuits and use a more evolved NML technology recently presented in the literature. It is worth mentioning that nothing of this sort is present in the literature, where only examples of algorithms managing simple circuits are found, and, in all cases, only conceived for general QCA, without any realistic adhesion to really available technology. The main added value of this work is the demonstration of a new flexible architecture enhancing NML characteristics through both functional verification and actual implementation at the physical design level.

III. ToPoliNano Organization

ToPoliNano (Torino Politecnico Nanotechnology Tool), is a design and simulation tool developed to deal with new emerging technologies such as nano-array based technologies and QCA, with a particular focus on NML (for a description of NML see section VI). The aim was to design a completely new tool to allow researchers to explore these technologies exploiting the same top-down approach used for CMOS technology.

Starting from a HDL description of a circuit, which is a technology independent description, we can obtain a complete and detailed layout based on NML logic and then we can perform logical and electric simulations to evaluate the correct behavior at device level, to estimate the total occupied area and the whole power consumption, and to specify feedbacks to technologists and architects in order for them to devise the correct new directions in the research. ToPoliNano is fully developed in C++ and can currently run on three platforms:

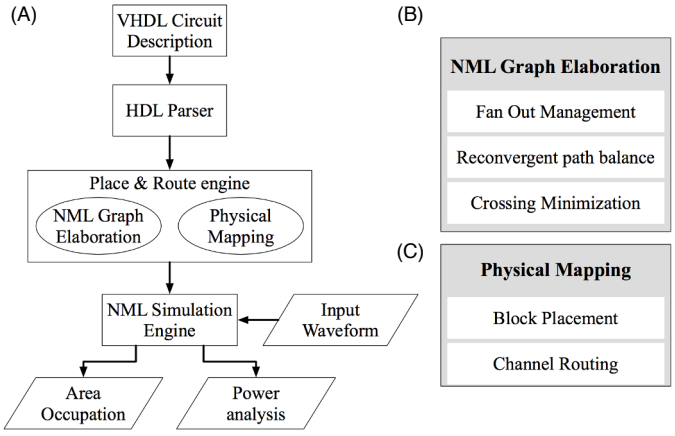

Fig. 2. **(A)** ToPoliNano design flow. **(B)** Techniques applied during graph elaboration. **(C)** Techniques applied during physical mapping

Linux, Mac OS and Windows. At the moment it counts more than 100k lines of code, excluding external libraries. With respect to other existing tools (such as QCA-LG [42]), ToPoliNano offers a complete working flow starting from the Register Transfer Level (RTL) description of any kind of architectures using a High Level Description language (HDL) to its automatic layout generation and, after the circuit is placed and routed, it also allow the logic simulation. More recently ToPoliNano has also been added a full custom layout feature of a circuit that can be either simulated with logic simulation or extracted so that a RTL description is obtained (HDL). Another important feature of ToPoliNano can be recognized in its flexibility, as it offers the possibility to work with different emerging technologies and implementations. Manifold layout engines have been specifically tailored for a few target emerging technologies. In other words the same HDL file can be used to design circuits based on silicon nanoarrays and gate-all-around transistors or circuits based on NML. In the NML case different kinds of implementations are possible (e.g. different clock mechanisms, different magnet shapes etc...). Even the logic simulation that can be performed on the designed layout can behave differently depending on the chosen simulation engine.

The main organization and flow of the application is shown in Fig. 2.A. The application allows the user to select a target technology (NML or nano-array based at the moment). Then, the CAD is opened and it is possible to create a new VHDL file using the integrated editor or to load an existing one from a specific library. As it happens in commonly used tools, the VHDL netlist is parsed and compiled, and at this point the circuit, represented through a graph, is logically linked to an existing library of elementary NML logic sub-blocks. ToPoliNano generates the physical layout: first it elaborates the NML graph generated by the parsing process according to NML technological constraints, then it performs the physical mapping of the circuit. These two steps might resemble the classical procedures used in CMOS technology. However, it is fundamental to remark that these are completely new algorithms only inspired to existing ones, but totally reinvented to tackle new problems and constraints derived by NML technology. The difficulty that here is on top of the normal problems in physical design algorithms relies on the fact that for QCA the layout has a direct impact on the correctness of logic behavior, not only on performance like in standard technologies. This imposes a new and critical point of view on the way the methods normally used are applied.

The NML graph elaboration is achieved in three steps according to Fig. 2.B-C: 1) The Fan Out Management modifies the graph taking into account NML limitations of the fan-out that each magnetic cell can support (which in this case is not related to a maximum load as in CMOS, but to the maximum number of magnets that can be correctly driven in the correct magnetization state). 2) The Reconvergent path balance phase guarantees signals synchronization, it ensures that each path is composed by the same number of nodes (to solve automatically the "layout=timing" problem mentioned in section II). 3) In the Cross Wire Minimization phase the graph is elaborated to reduce the number of these components starting from the principles of different algorithms such as Barycenter, Fan-out duplication, Simulated Annealing and Kernighan-Lin [38], elaborated and modified for better exploiting the potentials of NML.

During the physical mapping phase, each node of the elaborated graph is mapped into the corresponding magnetic logic gate and it is placed in order within the circuit respecting the strict constraints that this technology imposes for the correct signal propagation behavior. The final position of each gate is obtained during the routing phase; here the aim is to maximize the circuit compaction, therefore reducing the length of interconnection wires. Again it is worth underlining that in QCA an interconnection is a sequence of basic cells (i.e. of nanomagnets in NML) and it is thus a logic element, not just a wire as in standard technologies. This changes the point of view on the routing algorithms. At this point the result produced by the place and route engine – represented through a graphical interface – are the starting point for the logical and the electrical (magnetic in NML) simulations. The tool translates the internal representation of the layout into a regular matrix in which each node represents a single magnet. In this way with a specific visiting algorithm, conceived ad-hoc for this technology, the state of each element can be evaluated. This kind of approach is unique in literature, where only physical level micromagnetic finite element simulators are available which do not allow the simulations of more than tens of magnets. At the end, ToPoliNano produces the simulations waveforms (both in graphical and textual format) describing the behavior of inputs, outputs and clock signals. In section VI specific description of the NML constraints and implementation are given.

IV. Reconfigurable Systolic Array

The Reconfigurable Systolic Array (R_SA) is composed of a square array of Reconfigurable Processing Elements (R_PEs). The structure of an R_PE is shown in Fig. 3. It is composed of a Reconfigurable ALU, a CTRL block, registers and multiplexers. CTRL block redirects the **ctrl_in**

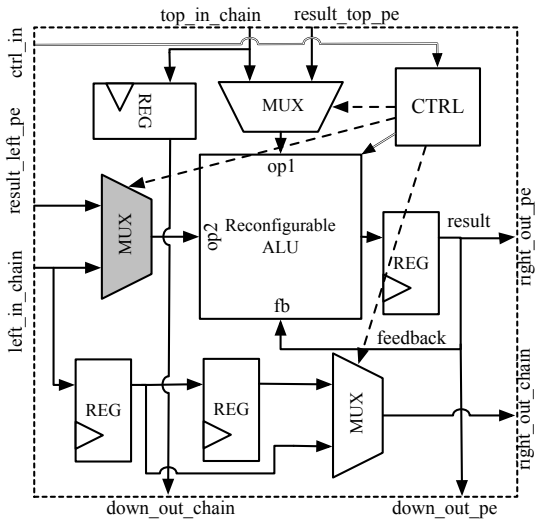

Fig. 3. Reconfigurable Processing Element (R_PE): CTRL block redirects the **ctrl_in** signal to all the configurable elements of the R_PE, i.e. the 2 input multiplexers, the horizontal chain multiplexer, and the Reconfigurable ALU. The transmission of **ctrl_in** to R_PE below is not represented for sake of simplicity.

signal to multiplexers and to the Reconfigurable ALU. This signal defines the behavior of each PE, and it is transmitted locally from left to right in each column (not represented in Fig. 3). Each R_PE receives 4 input signals: **top_in_chain** is a signal provided from the boundaries of the SA and transmitted through each PE with a direct propagation via a register to **down_out_chain**; the same behavior can be observed for **left_in_chain**, but in this case the propagation can occur through 1 or 2 registers depending on the control signal of the multiplexer; the other two inputs, **result_top_pe** and **result_left_pe** are the values calculated in the above and left PE respectively (Fig. 4.A).

Input multiplexers are used to choose between the two inputs for each side (the one evaluated in previous PE and the one that arrives from the outside). The Reconfigurable ALU can implement a given set of operations, always working on 3 input data: they can be chosen among input data coming from multiplexers, stored values '0' and '1', and feedback signal shown in Fig. 3. In our proposal the Reconfigurable ALU can implement addition, multiplication, Multiply and ACcumulate (MAC), and logic left shifting (Fig. 4.B). The adder is implemented as a classical Ripple Carry Adder, while the multiplier is an Array Multiplier composed of AND gates to perform multiplications and Ripple Carry Adders to sum partial products. The MAC is actually a multiply and add structure that can use the **fb** (feedback) signal as input to implement a MAC. Depending on the available area, the designer can decide to enhance the ALU providing hardware for other arithmetic or logic functions.

Each R_PE can be programmed independently from the others (while in classical SA all PEs execute the same function). In this way a given PE in a custom SA that implements several operations can be mapped to a set of R_PEs in the R_SA. R_PEs are programmed sending a set of **Ctrl** signals at the first column of the array (in Fig. 4.A local transmission from one PE to the successive is not represented). Each PE has a configuration register that stores the **Ctrl** signal. One bit of **Ctrl** signal is used to select between programming and normal operation mode (it represents the Write Enable of the configuration register).

V. Reconfigurability

To demonstrate reconfigurability of the R_SA we propose two different applications: matrix multiplication and a set of FIR filters.

A. Matrix multiplication

Given two rectangular matrices $A = (a_{ik})$ and $B = (b_{kj})$ of order $N_1 \times N_3$ and $N_3 \times N_2$ respectively, their product, matrix $C = A \times B$, $C = (c_{ij})$, of order $N_1 \times N_2$, can be obtained according to the equation (1):

$$c_{ij} = \sum_{k=1}^{N_3} a_{ik} \cdot b_{kj}, \quad i = 1, 2, \ldots, N_1 \quad j = 1, 2, \ldots, N_2 \quad (1)$$

This can be mapped to a SA of $N_1 \times N_2$ cells, each performing Multiply and Accumulate operations to store partial results $c_{ij}(k)$ at each iteration k (Fig. 4.C) [13]. Therefore, the R_SA must be configured in order to: use the MAC resource in the Reconfigurable ALU, with **op1** and **op2** as inputs of the multipliers and **fb** signal as second input for the adder. The other multiplexers must select **top_in_chain** and **left_in_chain** as input data, and one single register in the left-to-right transmission of **left_in_chain**.

B. FIR filters

The following explanation refers to Fig. 4.D describing the logic organization of FIR filters, to Fig. 4.E describing the implementation in the reconfigurable architecture and to Fig. 5 discussing the simulations.

Given a discrete-time FIR filter (Fig. 4.D), the output sequence $y[n]$ can be expressed in terms of input sequence $x[n]$ and weights b_j with equation (2):

$$y[n] = \sum_{j=0}^{N} b_j \cdot x[n-j] \quad (2)$$

where N is the filter order.

To map FIR filters in the R_SA cells must be programmed in different ways: this is possible since we can manage one configuration signal for each PE. In Fig. 4.E, two rows of the Reconfigurable Array are used to implement a FIR filter. Other FIR filters can be mapped in other rows of the R_SA. They must all share the same weights b_j since these must be provided from the external and are locally transmitted to PEs below (through each column). To map the filter in the R_SA three different configurations must be used: MUL, ADD and top-to-right signal transmission (the bottom-left PE in Fig. 4.E), hereinafter called TRANSMIT. In the following each of these configuration is described.

1) MUL cells are configured to execute multiplications on incoming operands from top and left, therefore the Reconfigurable ALU is programmed to use **op1**, **op2** and '1'

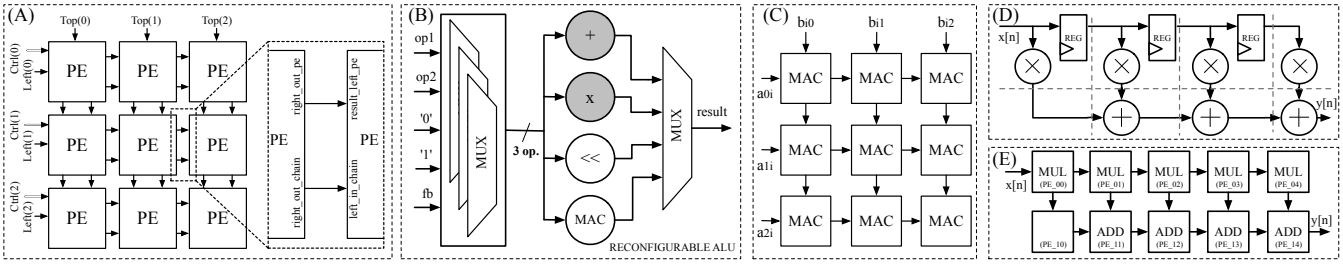

Fig. 4. **(A)** R_SA: each PE communicates with the one below and the one at its right using two signals. Ctrl(i) signals are transmitted through rows and are used to program each PE. **(B)** Reconfigurable ALU: the 4 computational blocks work on 3 input data, that are chosen configuring the three input Muxes. Each of this Mux is used to select one of the 5 operands available: op1, op2, fb, 0 and 1. The executed operation is selected configuring the output Mux. Control signals are not shown for sake of simplicity. **(C)** SA to execute matrix multiplication. Values of the resulting matrix c_{ij} are stored in each PE. **(D)** FIR filter with dashed lines to represent the mapping to the R_SA. Each line represents a cut-set for retiming, therefore PEs that represent the upper row of the FIR filter must have a 2-clock delay in left-to-right input transmission. **(E)** FIR filter on two lines of the R_SA. Note that three different configurations are required (the blank cell must be configured to transmit the value coming from top to the right). Depending on the number of rows, the array can map several FIR filters, one below the other.

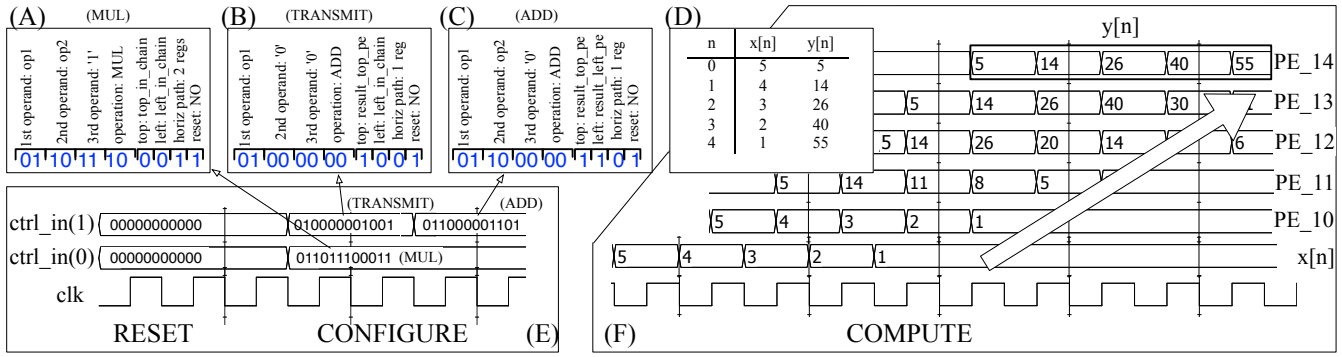

Fig. 5. Simulation of the R_SA configured to implement a FIR Filter. **(A)** Configuration word for MUL cells. **(B)** Configuration word for TRANSMIT cells. **(C)** Configuration word for ADD cells. **(D)** FIR filters input values and correspondent results. **(E)** RESET and CONFIGURE waveforms. **(F)** COMPUTE waveforms that represent inputs (x[n]) and results of PEs in the bottom row (PE_1x).

as operands. The other multiplexers select **top_in_chain** and **left_in_chain**. Notice that in this case the left-to-right transmission must follow the path with two registers, to have one clock delay difference with respect to the horizontal path between Adders, where one register only (the one that stores the result) is present. In Fig. 4.D each dashed line represents a cut-set for retiming, therefore additional registers must be inserted in each left-to-right signal transmission and in each path from multipliers to adders. Configuring word for MUL cells is shown Fig. 5.A.

2) ADD cells are configured to execute addition on incoming operands from top and left, therefore the Reconfigurable ALU is programmed to use **op1**, **op2** and '0' as operands. The other multiplexers select **result_top_pe** and **result_left_pe**. In this case the delay for the partial result to be transmitted to neighbor PE is always 1 clock cycle as expected. Configuring word for ADD cells is shown Fig. 5.C.

3) TRANSMIT: **result_top_pe** must be transmitted as result to the PE at its right. This can be done configuring the PE to execute an addition with operands **result_top_pe** (**op1** inside the Reconfigurable ALU), '0' and '0'. Configuring word for TRANSMIT cell is shown Fig. 5.B.

Fig. 5 summarizes the example of FIR filter implementation in the R_SA. Three phases are necessary: reset, to clear registers; configure, to program each PE to execute a given function (Fig. 5.E); compute, when the array is actually used for its scope, in this case FIR filtering (Fig. 5.F).

VI. NML Implementation

The R_SA was mapped on NML technology using ToPoliNano. At the time of writing the tool is not able to handle sequential circuits and tackle hierarchical floorplanning. Both functions are under development. In this work we have automatically generated the layout of all main blocks with ToPoliNano, while the floorplan customly. Fig. 6 shows the layout of a simple 2 bits multiplier. This block is not part of the processing element but is used here to highlight the basic circuits structure. Clock zones are made by parallel stripes which correspond to the wires where the current flows and therefore generates the magnetic field used as clock. This layout is based on the results shown in [33], where these wires where physically implemented. While other type of NML circuits based on different clock mechanisms can have different clock zones layout, we choose to base our design on this particular layout because it has the advantage to automatically synchronize signals without the need of asynchronous logic [8]. NanoMagnets used for this design are $50 \times 100 \times 20\,nm$ blocks of Permalloy, with $20\,nm$ spacing between magnets. It is possible to use magnets of different sizes, guaranteeing an aspect ratio of 1.2 minimum to have a bistable switch logic behavior. As shown in Fig. 6 circuits are based on a AND, OR [43] and inverters as basic logic gates. Two support structures

are also required, the coupler and the cross wire. The coupler is just a simple structure used to split a signal in two parts while the cross wire is a particular block that allows to cross two wires on the same plane [33]. The cross wire is crucial to build any working circuit in NML technology, since up to now no multilayer structures are possible.

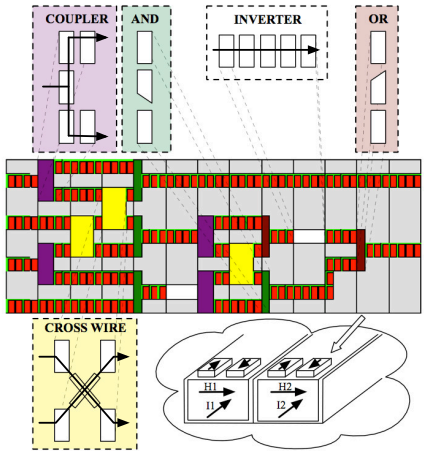

Fig. 6. Layout example of a 2-bit Multiplier. Clock zones layout are organized in parallel stripes and mirrors the physical structure of the wires where the current used to generate the magnetic field flows. The logic gate set includes AND, OR and inverters. These two gates have a different shape with respect to other magnets; this implies the presence of a preferred verse of magnetization wich can be used to realize those two logic functions [43]. Couplers are used to split a signal in two parts, while cross wires allow to cross two wires on the same plane without interferences.

Fig. 7 shows the general floor plan of the processing element, on the left the equivalent schematic. The processing element can be seen as a two stage block. The first stage is based on an 8 bits adder and an 8 bits multiplier. Their output is connected through a multiplexer to the second stage, based on an 8 bits adder and a 16 bits multiplier. This particular organization was chosen to better exploit the technology characteristics. Depending on how the multiplexers are configured we can obtain the three operations required by the programmable processing element (3-operands sum, 3-operands multiplication and multiply-and-accumulate) and an additional operation (sum and multiplication) that expands the capabilities of the SA. A shift register is also required to complete the logic functionalities of the processing element. Fig. 7 (top and bottom) shows an example of two blocks obtained using ToPoliNano, a 2to1 8 bits multiplexer in Fig. 7 (top) and an 8 bits adder in Fig. 7 (bottom). Table I reports the ToPoliNano performance for the layout creation of the main blocks of the circuit. In particular, it shows the time (expressed in ms) taken to design the entire layout and to run the crosswire reduction algorithms, which have been developed with the aim to minimize the number of crosswires present in the circuit and as a consequence able to significantly reduce the area occupation.

The floorplan organization has a particular U-shape, where the second stage is bent under the first one. In this way input signals are from the top-left side and output signals are connected to the bottom-right side. This solution was chosen

	8bit Adder	8bit Mul	8bit 2to1 Mux
Crosswire reduction [ms]	4	12	2
Total layout [ms]	60	357	18

TABLE I
TOPOLINANO PLACE & ROUTE EXECUTION TIME FOR THE MAIN BLOCKS OF THE SYSTOLIC ARRAY

considering the matrix-like structure of the SA, so that input and output signals among neighbor processing elements are perfectly matched. The estimated area of the whole processing element is 0.18 mm^2. This value is quite big but the reason is intrinsic to the limited maturity of the technology. Up to now no multilayer structures can be fabricated. Circuits are therefore constrained to one single layer and this fact leads to a huge circuit area [7]. Moreover the place&route algorithm of ToPoliNano is still under development and further improvements are possible. It is worth noticing that in literature often hypotheses on stacked magnetic layers and/or out-of-the plane crosswires can be found. Here we prefer to maintain the maximum coherency with technology that is reasonably feasible at the moment.

At this point it is important to underline a fact. In this paper we are presenting an innovative architecture designed and partially obtained through the first CMOS-like CAD for QCA circuits. This is the first time that a work like this is presented in literature. Further optimization to the place&route algorithms, to the floorplan creation and to the technology itself [8][3] are already under development and they will lead to a substantial reduction of circuit area. However, the consequence of this work are very important also at this stage of development of this nanotechnology.

ToPoliNano is also capable of simulating the generated circuits using a behavioral simulation [40] suited for circuits made by a very large number of magnets. The simulations made with ToPoliNano are at a intermediate level of abstraction, where the elementary device is modeled in terms of logic behavior and the interactions among magnets organized as logic gates reproduce the correct physical behavior. However this does not involve physical descriptions in order to reduce the simulation time. Other models [44][45] are more similar to a micromagnetic simulations, but they do not allow simulations of big circuits and the logic behavior of the circuit is correspondent to the one of ToPoliNano. For this reason the approach used for ToPoliNano is to have a simplified model for “logic” simulation, similar to the approach used in [34]. Nevertheless, the accuracy of ToPoliNano simulations have been demonstrated [30]; indeed the behavior of a nanomagnet can be assimilated to a bistable switch [46], which is the basic assumption used for ToPoliNano simulation. Of course the technological constraints used during the place and route phase are those that are coherent with the model used in the simulations, that have been previously and carefully validated using micromagnetic simulations. In case different technological constraints like magnet sizes and distances are chose, then micromagnetic simulations are run for every elementary logic component (AND, OR, MV, wires,...) that is always re-used in the circuit, so that the switch level simulator is updated and

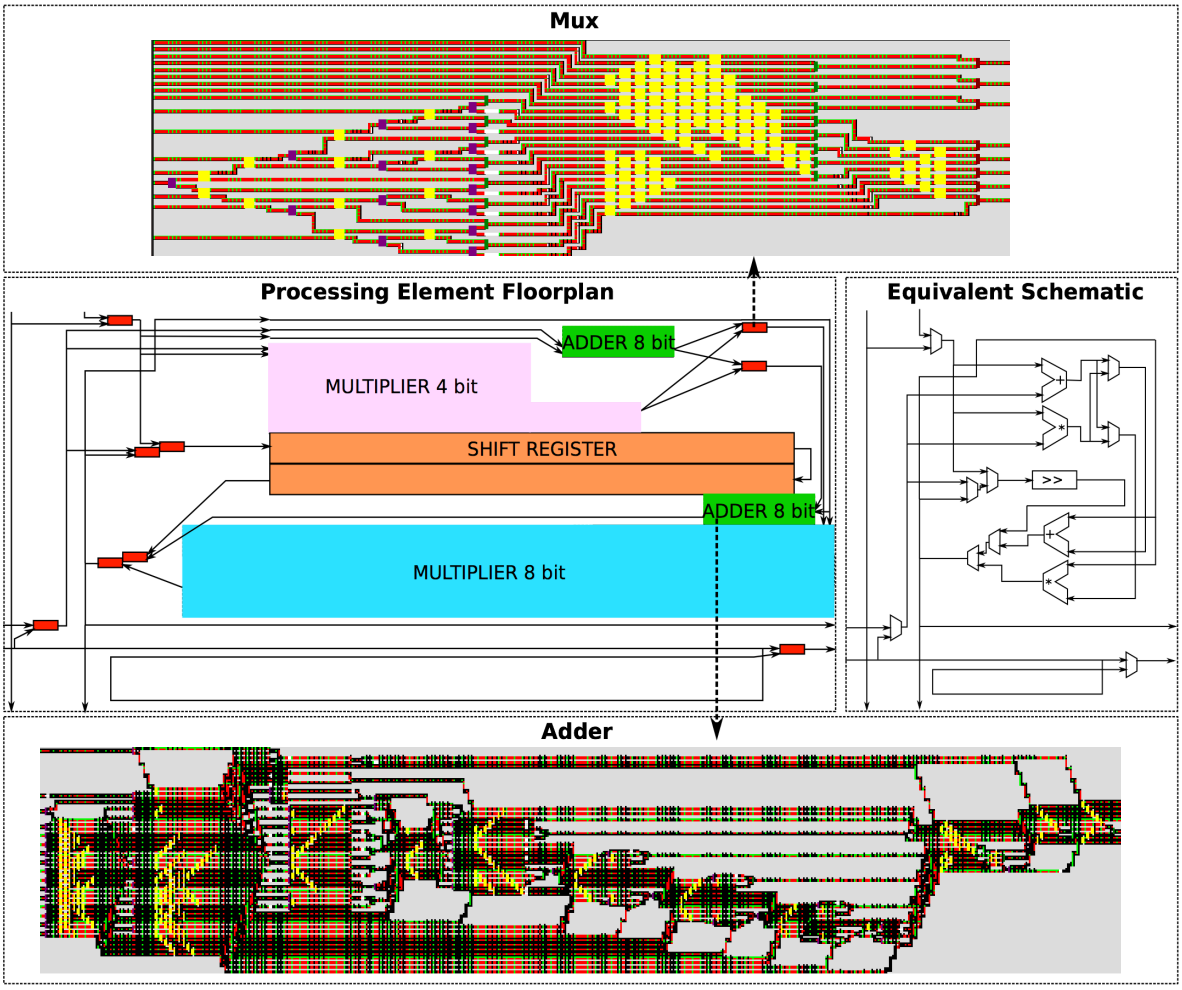

Fig. 7. Processing element floorplan and layout. The general floorplan is shown in the figure center, while on the right the equivalent circuit can be seen. On top and on the bottom an example of two circuits obtained by ToPoliNano can be seen: A 2to1 8 bits multiplexer (top) and an 8 bits adder (bottom).

validated.

As previously stated, the floorplan was manually estimated because an appropriate algorithm for top level floorplan creation is still under development. As a consequence it is not possible for now to simulate the entire processing element with ToPoliNano. We have however simulated and verified the behavior of all major blocks. In Fig. 8 we report as an example the simulation of the 2 bits multiplier of Fig. 6. While the 2 bit multiplier is not part of the processing element it was chosen for the sake of clarity of the simulation. As can be seen from Fig. 8 the multiplication switch level output value is correct (A0,A1 and B0,B1 are the values of the magnetization of the magnets on the left of the multiplier in figure 6 and Mul0 and Mul1 are the values of the magnetization of the output magnets in the same figure).

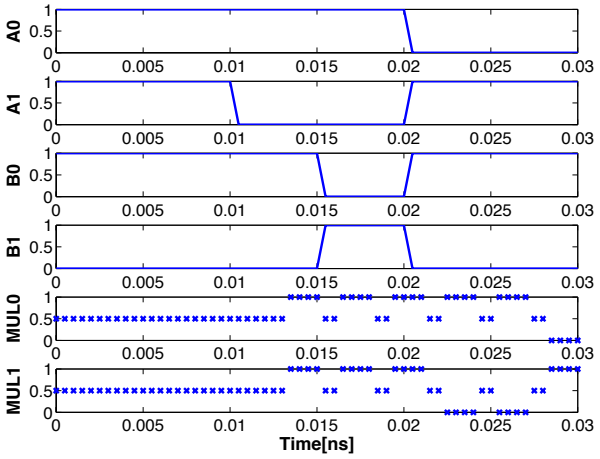

Fig. 8. Switch level magnetic simulation of a 2-bit multiplier based on NML.

Comparison with a commercial FPGA: Although the final objective of this article is to present a complete design flow for NML technology, the reconfigurable nature of the proposed architecture led us to a simple, preliminary but yet interesting, comparison with existing reconfigurable architectures. In this paragraph we provide a hint of this comparison, using as target existing architecture the Altera FPGA Stratix IV EP4SGX70 [47]. The chip area for this FPGA is $1225\,mm^2$. In an equivalent area it is possible to place about 6800 R_PEs. The number of reconfigurable resources is higher in the FPGA, since it has

29040 Adaptive Logic Modules (ALMs), each containing 2 6-input LUTs. According to [47], the FPGA can perform 153 Giga Multiply-Accumulate Operations per Second (GMACS), working at 400 MHz. Considering a clock frequency of 100 MHz for NML R_SA implementation [36], with one MAC in each PE, it would be possible to perform 680 GMACS. This particularly fits the case of DSP applications and highlights how NML, even if it is still in its infancy, can find its way against current commercial CMOS technology. We are currently upgrading the R_SA structure and we will provide a more in-depth comparison with CMOS FPGA architectures in the future.

VII. Conclusions

We have presented an innovative architecture, a Reconfigurable Systolic Array, thought to exploit the true potential of emerging nanotechnologies. It couples the regularity and absence of long interconnection wires of systolic arrays with the programmability not far from that of a FPGA, greatly enhancing the future commercial appealing of these technologies. We have completely designed the architecture and partially implemented it with ToPoliNano, a design tool conceived to emulate the top-down design methodology used in CMOS. This work represents therefore something never attempted in literature before and greatly enhances the research in the nanotechnology field.

References

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

[2] M. Liu, C. Lent, and Y. Lu, "Molecular electronics - from structure to circuit dynamics," in *6th IEEE Conf. Nanotechnology*. Cincinnati, Ohio, USA: IEEE, 2006, pp. 62–65.

[3] M. Vacca, M. Graziano, A. Chiolerio, A. Lamberti, M. Laurenti, D. Balma, E. Enrico, F. Celegato, P. Tiberto, and M. Zamboni, "Electric clock for NanoMagnet Logic Circuits ," *Anderson, N.G., Bhanja, S., Field-Coupled Nanocomputing. LNCS. Springer, Heidelberg*, 2014.

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

[5] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, "Quantum cellular automata," *Nanotechnology*, vol. 4, no. 1, pp. 49–57, 1993.

[6] 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, pp. 25–25.

[7] 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.

[8] M. Vacca, M. Jiang, J. Wang, F. Cairo, G. Causapruno, G. Urgese, A. Biroli, and M. Zamboni, "NanoMagnet Logic: an Architectural Level Overview ," *Anderson, N.G., Bhanja, S. (eds.), Field-Coupled Nanocomputing. LNCS. Springer, Heidelberg*, 2014.

[9] M. Graziano, M. Vacca, D. Blua, and M. Zamboni, "Asynchrony in Quantum-Dot Cellular Automata Nanocomputation: Elixir or Poison?" *IEEE Design Test Comput.*, vol. 28, no. 5, pp. 72 –83, sept.-oct. 2011.

[10] M. Vacca, J. Wang, M. Graziano, M. Roch, and M. Zamboni, "Feedbacks in qca: A quantitative approach," *IEEE Trans. Very Large Scale Intgr. (VLSI) Syst.*, vol. 23, no. 10, pp. 2233–2243, 2015.

[11] G. Causapruno, M. Vacca, M. Graziano, and M. Zamboni, "Interleaving in Systolic-Arrays: a Throughput Breakthrough," *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 1940–1953, 2015.

[12] M. Crocker, X. Hu, and M. Niemier, "Design and Comparison of NML Systolic Architectures ," *Nanoarch*, 2010.

[13] 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.

[14] L. Lu, W. Liu, M. O'Neill, and E. Swartzlander, Jr, "QCA Systolic Array Design," *IEEE Trans. Comput.*, vol. 62, no. 3, pp. 548–560, 2013.

[15] L. Lu, W. Liu, M. O'Neill, and E. Swartzlander, "Qca systolic matrix multiplier," in *IEEE Comput. Society Annual Symp. VLSI (ISVLSI)*, Oct. 2010, pp. 149–154.

[16] D. Pala, G. Causapruno, M. Vacca, F. Riente, G. Turvani, M. Graziano, and M. Zamboni, "Logic-in-memory architecture made real," *ISCAS*, May 2015.

[17] M. Cofano, G. Santoro, M. Vacca, D. Pala, G. Causapruno, F. Cairo, F. Riente, G. Turvani, M. R. Roch, M. Zamboni, and M. Graziano, "Logic-in-memory: A nanomagnet logic implementation," *ISVLSI*, July 2015.

[18] H. Lim and J. Swartzlander, E.E., "Multidimensional systolic arrays for the implementation of discrete Fourier transforms," *IEEE Trans. Signal Process.*, vol. 47, no. 5, pp. 1359 –1370, may 1999.

[19] H. Yeo and Y. H. Hu, "A modular high-throughput architecture for logarithmic search block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 3, pp. 299 –315, 1998.

[20] G. Causapruno, G. Urgese, M. Vacca, M. Graziano, and M. Zamboni, "Protein Alignment Systolic Array Throughput Optimization," *IEEE Trans. Very Large Scale Intgr. (VLSI) Syst.*, 2014.

[21] M. Graziano, S. Frache, and M. Zamboni, "A Hardware Viewpoint on Biosequence Analysis: What's Next?" *ACM J. Emerging Tech. Computing Syst.*, vol. 9, no. 4, 2013.

[22] C. K. Wijenayake, A. Madanayake, and L. Bruton, "FPGA-prototypes of differential-form 2D-IIR systolic-array DSP architectures for multi-beam plane-wave filters," in *IEEE Workshop Signal Processing Systems (SIPS)*, Oct 2010, pp. 58–63.

[23] Q. Wang, A. Li, Z. Li, and Y. Wan, "A Design and Implementation of Reconfigurable Architecture for Neural Networks Based on Systolic Arrays," in *Proc. 3rd Int. Conf. Advances in Neural Networks - Vol. Part III*, ser. ISNN'06. Berlin, Heidelberg: Springer-Verlag, pp. 1328–1333.

[24] I. Panagopoulos, C. Pavlatos, G. Manis, and G. Papakonstantinou, "A Flexible General-purpose Parallelizing Architecture for Nested Loops in Reconfigurable Platforms," in *Proc. 17th Int. Conf. Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 20–30.

[25] W. Jin, C. Zhang, and H. Li, "Mapping multiple algorithms into a reconfigurable systolic array," in *Canadian Conf. Electrical and Comput. Engineering (CCECE)*, 2008, pp. 001 187–001 192.

[26] A. Csurgay, W. Porod, and C. Lent, "Signal processing with near-neighborcoupled time-varying quantum-dot arrays," *IEEE Trans. Circuits Syst.*, vol. 47, no. 8, pp. 1212–1223, 2000.

[27] A. Pulimeno, M. Graziano, D. Demarchi, and G. Piccinini, "Towards a molecular QCA wire: Simulation of write-in and read-out systems," *Solid-State Electronics, Elsevier*, vol. 1, p. 7, 2012.

[28] A. Pulimeno, M. Graziano, V.Cauda, A. Sanginario, D. Demarchi, and G. Piccinini, "Bis-ferrocene molecular qca wire: ab-initio simulations of fabrication driven fault tolerance," *IEEE Trans. Nanotechnol.*, vol. 12, no. 3, 2013.

[29] M. Graziano, A. Pulimeno, R. Wang, X. Wei, M. R. Roch, and G. Piccinini, "Process variability and electrostatic analysis of molecular qca," *J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 2, pp. 18:1–18:23, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2738041

[30] M. Vacca, M. Graziano, and M. Zamboni, "Majority Voter Full Characterization for NanoMagnet Logic Circuits," *IEEE Trans. Nanotechnol.*, vol. 11, no. 5, pp. 940–947, 2012.

[31] M. Graziano, A. Chiolerio, and M. Zamboni, "A Technol. Aware Magnetic QCA NCL-HDL Architecture," in *Int. Conf. Nanotechnol.* Genova, Italy: IEEE, 2009, pp. 763 – 766.

[32] 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.

[33] M. Niemier and al., "Nanomagnet logic: progress toward system-level integration," *J. Phys.: Condens. Matter*, vol. 23, p. 34, Nov. 2011.

[34] J. Das, S. Alam, and S. Bhanja, "Low Power Magnetic Quantum Cellular Automata Realization Using Magnetic Multi-Layer Structures," *J. Emerging and Selected Topics in Circuits and Syst.*, vol. 1, no. 3, pp. 267–276, Sep. 2011.

[35] M. S. Fashami, J. Atulasimha, and S. Bandyopadhyay, "Magnetization Dynamics, Throughput and Energy Dissipation in a Universal Multiferroic Nanomagnetic Logic Gate with Fan-in and Fan-out," *Nanotechnol.*, vol. 23, no. 10, Feb. 2012.

[36] N. Rizos, M. Omar, P. Lugli, G. Csaba, M. Becherer, and D. Schmitt-Landsiedel, "Clocking Schemes for Field Coupled Devices from Magnetic Multilayers," in *Int. Workshop Computational Electronics*. Beijin, China: IEEE, 2009, pp. 1–4.

[37] S. Frache, D. Chiabrando, M. Graziano, F. Riente, G. Turvani, and M. Zamboni, "ToPoliNano: Nanoarchitectures Design Made Real," in

IEEE/ACM Int. Symp. Nanoscale Architectures, Amsterdam, The Netherlands, 2012, pp. 160–167.

[38] M. Vacca, S. Frache, M. Graziano, F. Riente, G. Turvani, M. Roch, and M. Zamboni, "ToPoliNano: NanoMagnet Logic Circuits Design and Simulation," in *Field-Coupled Nanocomputing*, ser. Lecture Notes in Comput. Science, N. G. Anderson and S. Bhanja, Eds. Springer Berlin Heidelberg, 2014, pp. 274–306.

[39] G. Turvani, A. Tohti, M. Bollo, F. Riente, M. Vacca, M. Graziano, and M. Zamboni, "Physical design and testing of nano magnetic architectures," in *9th IEEE Int. Conf. Design Technol. Integrated Syst. In Nanoscale Era (DTIS)*, May 2014, pp. 1–6.

[40] M. Vacca, S. Frache, M. Graziano, and M. Zamboni, "ToPoliNano: A synthesis and simulation tool for NML circuits ," *IEEE Int. Conf. Nanotechnol.*, pp. 1–6, Aug. 2012.

[41] G. Turvani, F. Riente, M. Graziano, and M. Zamboni, "A quantitative approach to testing in Quantum dot Cellular Automata: NanoMagnet Logic case," in *10th Conf. Ph.D. Research Microelectronics and Electronics (PRIME)*, June 2014, pp. 1–4.

[42] T. Teodosio and L. Sousa, "QCA-LG: A tool for the automatic layout generation of QCA combinational circuits," in *Norchip, 2007*, pp. 1–5.

[43] M. Niemier, E. Varga, G. Bernstein, W. Porod, M. Alam, A. Dingler, A. Orlov, and X. Hu, "Shape Engineering for Controlled Switching With Nanomagnet Logic," *IEEE Trans. Nanotechnol.*, vol. 11, no. 2, pp. 220–230, Mar. 2012.

[44] G. Csaba, W. Porod, and Ã. I. Csurgay, "A computing architecture composed of field-coupled single domain nanomagnets clocked by magnetic field," *Int. Journal Circuit Theory and Applications*, vol. 31, no. 1, pp. 67–82, 2003.

[45] S. Breitkreutz, J. Kiermaier, C. Yilmaz, X. Ju, G. Csaba, D. Schmitt-Landsiedel, and M. Becherer, "Nanomagnetic logic: compact modeling of field-coupled computing devices for system investigations," *Journal Computational Electronics*, vol. 10, no. 4, pp. 352–359, 2011.

[46] G. Csaba and W. Porod, "Simulation of field coupled computing architectures based on magnetic dot arrays," *Journal of Computational Electronics*, vol. 1, no. 1-2, pp. 87–91, 2002.

[47] Altera, "Stratix IV FPGA High-Performance DSP Features," http://www.altera.com/devices/fpga/stratix-fpgas/stratix-iv/overview/architecture/stxiv-dsp-block.html.

G. Causapruno Giovanni Causapruno received the Dr.Eng. degree in Electronics Engineering from Politecnico di Torino, Torino in 2012, where he is a PhD candidate in Electronics and Communications Engineering. He works on parallel processing architectures for nanotechnologies.

F. Riente Fabrizio Riente received his M.Sc. Degree with honors (Magna Cum Laude) in Electronic Engineering in 2012 from Politecnico di Torino where he is now a Ph.D. candidate in Electronics. His research interests are in CAD tools for COMS and beyond-CMOS circuits performance exploration.

G. Turvani Giovanna Turvani received her M.Sc. Degree with honors (Magna Cum Laude) in Electronic Engineering in 2012 from Politecnico di Torino. She is pursuing her Ph.D. in Electronics with the same university. Her interests are in CAD tools development for non-CMOS nanocomputing.

M. Vacca Marco Vacca received the Dr. Eng. degree in Electronics engineering from the Politecnico di Torino, Turin, Italy, in 2008. In 2013, he got the Ph.D. degree in Electronics and Communications engineering and he is now a Research Assistant. His research interests include QCA and others beyond-CMOS technologies.

M. Ruo Roch was born in Torino, Italy, in 1965. He achieved Dr. Ing. and the Ph.D. degrees in 1989 and 1993, respectively. Since 1989 he is a Researcher at the Department of Electronic of Politecnico di Torino. Main area of interest is on dedicated microprocessor architecture, to telecommunication, DSP and high speed ASICs and design tools for beyond-CMOS technologies.

M. Graziano Mariagrazia Graziano received the Dr.Eng. degree and the Ph.D in Electronics Engineering from the Politecnico di Torino, Italy, in 1997 and 2001, respectively. Since 2002 she is Assistant Professor at the Politecnico di Torino. Since 2008 she is adjunct Faculty at the University of Illinois at Chicago and since 2014 she is a Marie-Curie fellow at the London Centre for Nanoelectronics. She works on "beyond CMOS" devices, circuits and architectures.

M. Zamboni Maurizio Zamboni got his Electronics Eng. and the Ph.D. degrees in 1983 and in 1988 from the Politecnico di Torino, respectively, where he is now a Full Professor. His research activity focuses on multiprocessor architectures design, in IC optimization for Artificial Intelligence, Telecommunication, low-power circuits and innovative beyond CMOS technologies.