Parallel and Serial Computation in Nanomagnet Logic: An Overview

Davide Giri, Giovanni Causapruno, and Fabrizio Riente, *Member, IEEE*

***Abstract*— Nanomagnet logic (NML) is a promising technology beyond CMOS technology because it can guarantee an extremely low-power consumption. This technology is extremely different from CMOS in some peculiar aspects: 1) logic gates and wires have the same delay and 2) the layout of a circuit influences its timing characteristics. With these characteristics, it is clear that a simple mapping of CMOS register-transfer level circuits in NML would be inefficient. Circuit logic design must be adapted to this new technology. One interesting aspect is the opportunity to design bit-serial circuits instead of parallel ones and achieve comparable performance with less area occupation. In this paper, we explore the parallel and serial design in NML with magnetoelastic clock through a common case study: the multiply and accumulate algorithm. This is designed in three different versions (fully parallel, fully serial, and parallel–serial) and analyzed in terms of latency, throughput, area occupation, and circuit dissipation.**

***Index Terms*— Magnetoelastic (ME) clock, nanomagnet logic (NML), parallel architectures.**

I. Introduction

NANOMAGNET logic (NML) is one of the most appealing technologies to replace CMOS in the next decades, when the successful era of Moore's law will finally come to an end [1]. NML represents the magnetic implementation of the quantum-dot cellular automata (QCA) principle. In NML, the polarization of nanomagnets represents logic values "0" and "1" [Fig. 1(a)]. This technology is particularly attractive for its low-power consumption [2].

The peculiar characteristics of NML, described in Section II, define a number of constraints that are unusual for CMOS logic circuits design. The most important is the so-called "timing–layout" problem, i.e., different layouts of a given circuit may lead to different time delays [3]–[5]. The reason is that wire delays of NML technology are even more critical than those of CMOS. Only a very short and fixed-length portion of a wire can be crossed in a clock cycle. Furthermore, wires and logic gates are the same in terms of delays as they are both composed in the same way by a series of adjacent magnets. This raises a fundamental requirement for NML logic circuits: long interconnections should be avoided and regular structures with local interconnections should be preferred [6], [7]. For this reason, solutions like systolic arrays [8], composed of arrays of identical cells and locally interconnected (i.e., one cell is connected only to the near ones), seem to be an ideal architectural choice and have been widely explored for NML implementation [5], [9], [10].

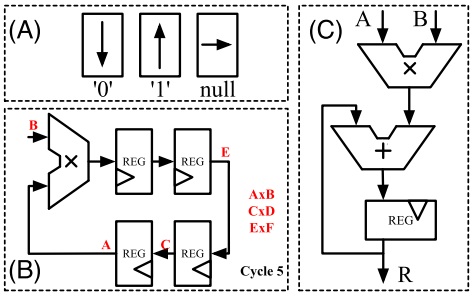

Fig. 1. (a) NML principle: polarization of nanomagnets represents logic values. (b) Pipeline interleaving technique: three operations are interleaved to fill the pipeline cue and maximize the throughput. (c) MAC architecture.

Taking into account these constraints, it is then clear that all the assumptions that are valid for the design of CMOS logic circuits must be thought again for NML technology. This is a fundamental step in the design of NML circuits, because starting from wrong assumptions derived from CMOS technology; the correspondent NML circuits can be inefficient.

It is crucial to point out that the best CMOS circuits usually maximize performances in terms of working frequency, at the cost of a higher complexity. However, such optimization does not necessarily have the same advantages when designed with NML, where the maximum working frequency is dependent on the technology itself and not on the architecture adopted. Therefore, optimization of NML cannot improve the circuit speed and it has to be aimed elsewhere: reduce area occupation and consequently power consumption, minimize internal delays reducing the pipeline stages of feedback loops.

One aspect particularly interesting for NML technology, is the choice between *serial* and *parallel* computation. Parallelism has been regularly increasing in VLSI circuits, given the performance improvement that can be achieved due to the low cost of interconnections. This has been possible also because of the multilayered structure of CMOS circuits. In NML, this choice may not be so straightforward, it is still a single layer technology at the time of writing, and its interconnections have a high cost both in terms of area occupation and latency.

CMOS allows wires to have fairly variable length without affecting the functionality, as long as the overall combinational delay between registers stays lower than the clock period.

Manuscript received November 9, 2017; revised February 1, 2018; accepted March 18, 2018. *(Corresponding author: Fabrizio Riente.)*

D. Giri is with the Computer Science Department, Columbia University, New York, NY 10027 USA.

G. Causapruno and F. Riente are with the Department of Electronics and Telecommunications, Politecnico di Torino, I-10129 Turin, Italy (e-mail: fabrizio.riente@polito.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2018.2821107

Instead, NML wires that connect two logic gates almost always need to be pipelined, because the amount of wire (i.e., the number of magnets) that can be crossed during a clock cycle is extremely short. A small change in wire's length can save up to several clock cycles, especially if in a feedback. This means that in a simple circuit with a long feedback loop as the one depicted in Fig. 1(b), most of the time is wasted in the loop without performing any computation [5]. If the loop is made as short as possible (transmitting one single bit for serial computation instead of N bits for parallel computation), and the same computation cell is reused several times, there may be the opportunity to reduce area and achieve similar performance.

In this paper, we give an overview of the serial and parallel computation in magnetoelastic NML (ME-NML), a particular NML implementation. A case study has shown that ME-NML is able to compete and even outperform a 28-nm low-power CMOS technology in terms of area and power consumption as well as the magnetic clock NML [11], therefore motivating further research. Here we use a multiply and accumulate (MAC) architecture as case study [Fig. 1(c)], which is a common structure always present in standard arithmetic logic units.

The contribution of this paper can be summarized as follows.

1) We implemented different versions of the MAC architecture: fully serial, serial–parallel, and fully serial exploring the design space of the ME-NML technology. Each architecture has been designed, modeled, and simulated in ME-NML.
2) We developed a register-transfer level (RTL) model that can be used to design any kind of ME-NML architecture by using a set of ME-NML standard cells.
3) We compared the performance of the ME-NML proposed architectures in terms of area, latency, and power consumption. Some work has been done on serial computation on generic QCA [12]–[16].
4) Understand if there is the necessity for a paradigm change from parallel to serial with the change of the technology from CMOS to ME-NML.

We want to explore the design of the MAC architecture in ME-NML, since there is not yet an automatic tool able to synthesize ME-NML circuits. Only a single ME-NML circuit has been designed so far [11], [17]. Some work has been done on serial computation in generic QCA [12]–[16], [18], [19], but from the best of our knowledge, a parallel versus serial analysis of this type has never been proposed.

The rest of this paper is organized as follows. Section II provides details about NML technology and its magnetoelastic clock implementation. Section III describes the nanomagnet cells that can be used to design any NML circuit, with a standard sells approach that we have introduced for the first time in [17]. Moreover, we introduce a VHSIC Hardware Description Language (VHDL) model to extrapolate hierarchically information about area occupation and power dissipation of a NML circuit. In Section IV, we describe three different implementations of the MAC: completely serial, partially serial and partially parallel, and completely parallel. Section V provides a comparison among the different solutions in terms of throughput, latency, area occupation, and power dissipation. Finally, conclusions are provided in Section VI.

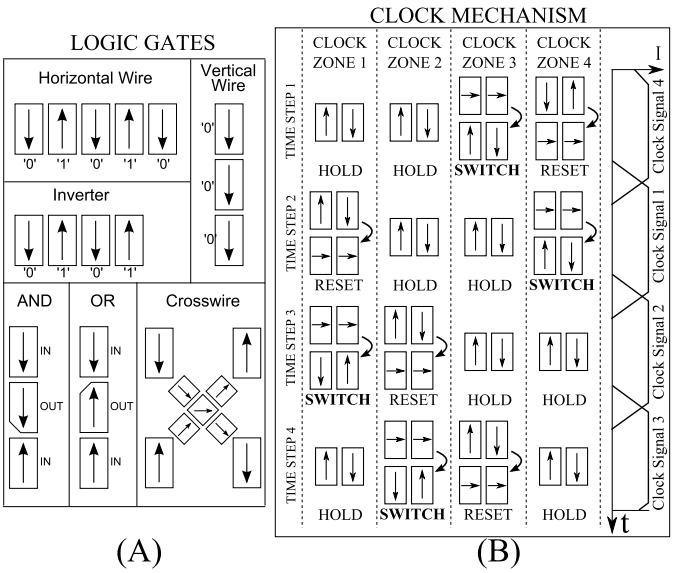

Fig. 2. (a) NML logic gates. (b) ME-NML clock mechanism applied to a horizontal NML wire and described in space (horizontally) and time (vertically). The four clock signals on the right are applied to different clock zones in space and they are periodic.

II. Nanomagnet Logic

In this section, we introduce NML technology to highlight the main differences with respect to CMOS.

NML is based on single-domain nanomagnets with dimensions between 50 and 100 nm. These magnets can have two stable states only, which are used to represent logic "0" and "1" [20]. Magnetic attraction between opposite poles creates an interaction between adjacent nanomagnets, used to propagate information in a NML circuit.

NML technology counts a small number of basic logic blocks. They are depicted in Fig. 2(a), where IN and OUT identify inputs and outputs. Horizontally, magnets align themselves antiferromagnetically, with each magnet having an inverted polarization with respect to the neighbors. The *inverter* port is, therefore, a simple horizontal wire with an even number of magnets. An odd number of adjacent magnets is instead a simple *wire*. In a vertical wire, the coupling is ferromagnetic; as a consequence no inversion is possible. A peculiarity of NML is the possibility of obtaining specific logic gates by modifying the shape of a magnet. By making magnets with a slanted edge, it is possible to give them a preferential polarization state, creating AND and OR logic functions [21]. Since NML is still a planar technology, independent wires can be crossed through the *crosswire* gate depicted in Fig. 2(a).

The electrostatic interaction between nanomagnets is not strong enough to propagate a signal through an NML wire. The switching of a cell requires as much energy as the barrier between its two stable states. For this reason, there is the need of an external mean able to control the signal propagation

acting on the energy barrier between the two stable states. The external mean should be able to lower the barrier forcing the cell in an unstable "null" state [Fig. 1(a)]. Once released, the cell will stabilize either at "0" or "1," depending on the state of neighboring cells.

This control mean is called *clock*. In principle, this technique could work with an infinite number of cascaded cells, but practically a signal cannot pass through a whole circuit at once. The cells' pattern would be too long causing propagation errors mainly due to thermal noise [22]. A spatial flow control system is mandatory.

The clock mechanism is shown in Fig. 2(b). Circuits are partitioned in small areas, each with a limited number of cascaded magnets; these areas are called *clock zones*. There are four clock signals corresponding to four clock zones. Clock signals have the same waveform but different phases. The second, third, and fourth clocks will have, respectively, 90°, 180°, and 270° phase shift with respect to the first clock.

Fig. 2(b) shows the clock waveforms on the right and the functioning of a wire divided into four clock zones on the left. As explained, we need a clock that can force cells in their unstable state before the switching phase. The clock waveform has four phases, as depicted in Fig. 2(b).

1) HOLD phase: The potential barrier is kept high by a low clock voltage. The cell cannot be influenced by neighbors.
2) RESET phase: The potential barrier is kept low, so the barrier between stable states is at its minimum. The cell is in the "null" state. This is the only phase with a high clock.
3) SWITCH phase: The clock voltage goes from high to low and so does the potential barrier. The cell will stabilize in one of the two states, depending on the neighboring cells in the HOLD phase. In this way, it also guarantees the propagation direction of the signal.

In Fig. 2(b), the signal goes from left to right through eight magnets, two per clock zone. The same wire is shown at four different times during the period of a single clock cycle. At time step 4, clock 3 is the only "high" one, therefore, the magnets in zone three switch to RESET state. The magnets in zones 1 and 4 were already stable at time step 3, so they keep staying stable in HOLD. The magnets in zone 2two were in RESET state, but their clock just switched to "low" so they immediately SWITCH to a stable state influenced by neighbor magnets in zone 1, while magnets in zone 3 cannot influence the switch as they are now in an intermediate state. This methodology assures data propagation in a specific direction, a signal traverses the clock zones in order from 1 to 4 and then 1 again.

The main difference between NML implementations is the clock system. The magnetic clock is based on a current that flows below nanomagnets in each clock zone. The current induces a magnetic field that forces the nanomagnets in the RESET state [2]. The ME-NML clock [17] is instead based on the magnetoelastic effect: the magnetization of magnetic materials undergoing mechanical stress is bonded. Applying a mechanical stress with proper intensity and direction, magnetic

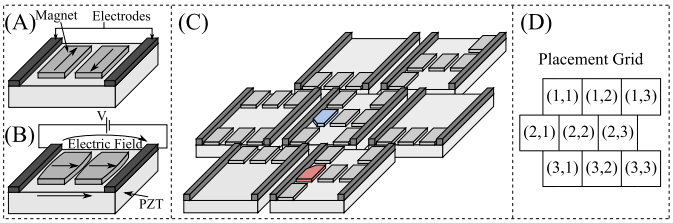

Fig. 3. (a) No voltage applied. (b) Voltage applied to the electrodes. The PZT substrate induces a strain on the nanomagnets forcing their magnetization to their intermediate state. (c) Circuit example composed by a few clock zones (cells). The blue magnet is an AND gate, the red magnet is an OR gate. (d) Placement grid for ME-NML cells.

cells can be forced into the RESET state. The magnetic cells (10-nm thick) are coupled with a 40-nm thick lead zirconate titanate (PZT) layer. The magnetic material is, then, controlled applying a voltage to the piezoelectric. When the voltage is applied, the strain induced by the piezoelectric material, forces the magnetization of the magnet's layer to the intermediate position, parallel to the short edges [see Fig. 3(b)]. The electrodes are deposited on top of the PZT, while the wires that drive the electrodes can be placed in additional layers, just as for CMOS. This makes this NML implementation compatible with the CMOS fabrication process.

Since the clock system exploits a voltage instead of a current, the power consumption is extremely low. The magnets switching represents the main source of power dissipation due to the charge and discharge of parasitic capacitances. The amount of loss depends on the voltage applied to the electrodes (CV^2) which is less than 1 V and a capacitance value in the order of hundreds of femtofarad. The overall power consumption is then around ten times lower than a 28-nm low-power transistor [23].

ME-NML circuits are composed of mechanically isolated islands, like the ones in Fig. 3(c), where colored magnets have a slanted edge to obtain AND and OR gates. Each island corresponds to a clock zone and it is driven by one of the clock signals, applied as a voltage on the platinum electrodes. Fig. 3(c) shows how to put together the clock zones to create a circuit. The communication among cells can take place only through the top and bottom corners, due to the presence of the electrodes. Cells are placed on a grid as in Fig. 3(d), where the coefficients identify row and column of the cell's positioning within the circuit. We refer to cells in Fig. 3(c) as "3 × 3," because each can contain three nanomagnets in a row and in a column. Another possible configuration with larger cells is "5 × 3."

In a N-phase clock system, signals need one clock cycle to propagate through N clock zones. As a consequence, the delay of a signal depends on how many clock zones it has to cross. This is quite different from CMOS where wires with different lengths have very similar delays. Each clock zone crossed by a signal can be modeled as a register. As a result, it is easy to understand that NML circuits are intrinsically pipelined.

The problem gets more complex when dealing with feedback signals. Note that the longer the feedback wire is,

the longer the delay will be. The input should be delayed to match the length of the feedback loop reducing the throughput. If, for instance, a circuit has a feedback of 5 cycles long, only an input of every 5 cycles can be fed. In fact, at any time, only $1/5$ of the magnets will contain useful data. A radical solution is the *data interleaving* [3], which allows to reach the maximum throughput. With interleaving, multiple noncorrelated set of operations are executed in parallel, so that the delay time between an input and the next is filled with other operations. The number of required parallel operations is equal to the delay (in terms of clock cycles) of the longest loop inside the circuit.

III. Standard Cells and RTL Model

In this section, we describe a set of standard cells developed for the design of ME-NML circuits and the RTL VHDL model that we implemented to simulate them.

In this paper, we used the 3×3 cells, which is the smallest size feasible with the current lithographic resolution. Compared to bigger cells, it has a shorter critical path (number of cascaded magnets) leading to both a higher working speed and a better signal propagation reliability. Due to the small size of this ME-NML cell, there is a limited number of possible magnets configurations. Therefore, it is possible to define a finite set of standard cells: a standard cell library [17], where each element is described in VHDL language. The result is that any digital circuit can be designed by assembling cells from the library. This standard cell approach confers to ME-NML, a propensity for design automation, making this technology very much suitable for having its own simulation and synthesis tool. The full 3×3 standard cell library is tabulated in Fig. 4.

We developed a RTL model in VHDL language that makes it possible to easily simulate any ME-NML circuit. It enables the functional analysis and hierarchical estimation of circuit performance in terms of area occupation and power consumption. Our model for ME-NML keeps consideration of all the relevant technology constraints, which means that all the components dimensions and materials choice and properties that we use have been proved and verified in the literature [2].

Logic gates must be distinguished also by layout and orientation, not only by their logic function because the library is thought in the perspective of a future automated tool for circuit design. Cells lying within the same row of Fig. 4 can be derived from each other by horizontal and/or vertical flipping. The binary numbers in the table associated with wire and inverter will be given as `generic` parameters to state the cell orientation. A VHDL `entity` has been defined for each line in Fig. 4. Each cell is modeled as a CMOS register plus, if needed, an ideal logic port.

Double wires, double inverters, and the mix of the two, together with crosswires, allow the propagation of two independent signals through a single cell. The wire with a "L" shape touches three corners and has two outputs; it is the only cell with four possible orientations.

Nanomagnets used for ME-NML have dimensions of 50×65 nm^2 with an intermagnet space of 20 nm. This size choice provides the best immunity to process variation [2], [23].

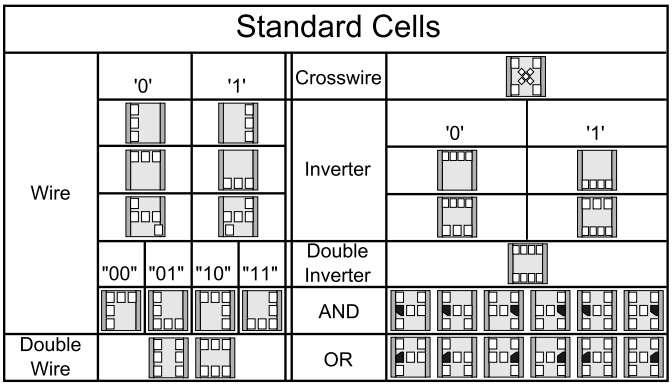

Fig. 4. Full 3×3 standard cell library for ME-NML.

The hierarchical structure of the VHDL model is shown in Fig. 5(a) where the `entity` of each standard cell outputs its own area occupation and power consumption [see Fig. 5(b)], which corresponds to the energy required for charging their parasitic capacitance. From this point of view, each cell is the equivalent of a capacitor. These information are summed and propagated up to the top entity, which collects the final area and power of the whole circuit. Therefore, the RTL model purpose is twofold: it allows functional simulation while precisely estimating power consumption and area occupation. The model also contains the exact physical mapping of the circuit. Each cell is assigned a clock phase (for ME-NML the clock phases are four) and a row and column parameters to identify the relative position within the circuit [Fig. 5(b)].

A single standard cell has an area equal to 0.059 μm^2.

The working frequency is $f_{\text{clk}} = 100$ MHz, which is close to the upper bound for NML technology. Power consumption is derived from energy consumption: $P = E \cdot f_{\text{clk}}$. There are two main sources of energy dissipation in NML circuits: magnets switching and clock generation network. The former is the intrinsic energy loss required to force magnets in the RESET state, while the clock network dissipation is due to Joule losses. Since PZT is an insulator, an ME-NML cell behaves as a capacitor. Therefore, the main contribution to clock losses (for a 100-MHz frequency) is the charge of such capacitor. The capacitance is estimated in the following equation [2]:

$$C = \frac{\epsilon_0 \cdot \epsilon_r \cdot t_{\text{PZT}} \cdot H_{\text{cell\_eff}}}{W_{\text{cell\_eff}}}. \tag{1}$$

The first three constants are the absolute dielectric constant (ϵ_0), the relative dielectric constant of PZT (ϵ_r), the thickness of the PZT substrate ($t_{\text{PZT}} = 40$ nm [2]). The other two values are the effective dimensions of a standard cell, without the inclusion of the separation between cells. Hence, $H_{\text{cell\_eff}} = 235$ nm and $W_{\text{cell\_eff}} = 250$ nm.

The following equation evaluates the voltage that should be applied to a clock zone to force it into the RESET state:

$$V = \frac{W_{\text{cell\_eff}} \cdot \sigma}{Y \cdot d_{33}}. \tag{2}$$

In this formula, we have the applied stress ($\sigma = 28$ MPa), Young's modulus for Terfenol ($Y = 80$ GPa), and the

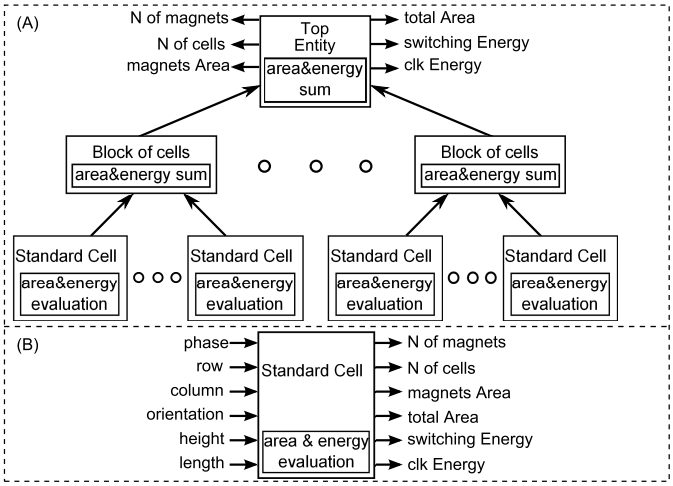

Fig. 5. (a) VHDL hierarchical model. The information on energy dissipation and area occupation are propagated hierarchically toward the top entity. (b) `Generic` inputs and outputs of a standard cell.

coefficient for strain and applied voltage coupling in the PZT substrate ($d_{33} = 150$ pm/V). Normally, for these cells, the applied voltage should be in the range of 0.7–1.3 V [2]. Finally, the energy required to charge the capacitance of one cell is listed in the following equation:

$$E_{\text{clk}} = \frac{1}{2} \cdot C \cdot V^2. \tag{3}$$

The clock period T_{clk} depends on technological constraints, not on the logic, as the critical path for signals is fixed, no matter which logic has been implemented.

The power contribution of the circuit for clock generation is negligible, as the circuit counts a limited number of transistors [24]. Therefore, this component will not be taken into account.

The half adder example in Fig. 6 helps to practically understand how ME-NML circuits are designed and how they work. Fig. 6(a) and (b) portrays the starting circuit scheme and the ME-NML circuit layout. The pattern from inputs to outputs is 5 clock zones long. The cells colors identify the clock phase of each cell, namely, the clock signal driving such cell (clock zone). The clock system choice for ME-NML is a four-phases overlapped clock, the four waveforms, with their assigned colors are listed in Fig. 6(c).

The RTL model maps each clock zone to one or two registers, plus a logic gate, if needed. The VHDL code for the ME-NML half adder describes the CMOS circuit as in Fig. 6(d). Notice that the path from input to output counts five registers (five pipeline stages), just like the five clock zones needed to pass through the ME-NML version in Fig. 6(b). The numbers marking registers define their clock phase. The timing diagram in Fig. 6(d) shows, as an example, the propagation of the inputs through the A-B-C-D pattern. It is quite clear from the timing that a signal needs one clock cycle T_{clk} to cross four clock zones (registers in the VHDL counterpart of the ME-NML circuit). Hence, a signal has a latency of $T_{\text{clk}} = 4$ to cross a clock zone.

IV. MAC Design in NML

The comparison between the parallel and the serial approach in ME-NML has been conducted choosing an algorithm and implementing it in different ways according to the way inputs are provided and outputs are given (parallel or bit-serial). These circuits have been designed using the standard cells and simulated using the RTL VHDL model presented in Section III.

A MAC unit, which implements the following equation, has been chosen as case study for the *parallel* versus *serial* approach in ME-NML:

$$R = \sum_i A_i B_i, \quad i = 0, 1, 2 \ldots \tag{4}$$

The MAC unit is generally composed of a multiplier, an adder, and an accumulator [Fig. 1(c)]. In Sections IV-A–IV-C, three different versions of the MAC are introduced: fully serial, serial-parallel, and fully parallel. Each of them has been designed, modeled, and simulated in ME-NML technology.

We adopt the following nomenclature: FA and HA are the full adder and half adder blocks, respectively; N is the number of bits of inputs A_i and B_i, while the output R has 2 N bits; T_{clk} and f_{clk} are the period and frequency of circuit clock, respectively. We refer to latency as the delay between the last input and the last output. The throughput represents instead the number of results that can be produced in a given time.

A. Serial MAC

The first implementation analyzed in this paper is the serial MAC, where inputs and output are all serial. The starting idea is to build a MAC unit counting only two 1-bit forced-air-cooled transformer (FA), one for the multiplication and one for the addition.

Fig. 7(a) shows the serial implementation of a 4-bit MAC ($N = 4$). It consists of a serial multiplier, a serial adder, and an accumulator, which is the adder feedback loop. Registers with the $\times 3$, $\times 4$, and $\times 32$ labels represent multiple cascaded registers. The multiplier accurately imitates the handmade multiplication algorithm: to generate the partial products properly, each bit of input B must be multiplied with all the input A bits. Therefore, the elapsed time to generate all the $A_i \times B_i$ products is $N^2 \times T_{\text{clk}}$. The multiplier produces one significant bit of the result every N clock cycles, therefore, the whole operation takes $2N^2 \times T_{\text{clk}}$. The adder sums up the multiplication result to the value in the accumulator, starting from the LSB and puts the result back into the accumulator. Four control signals applied to the four feedbacks assure a correct functioning of the MAC unit. Fig. 7(b) shows the ME-NML implementation of the 4-bit serial MAC.

The accumulator works as a shift registers. Its length is equal to $2N^2$ (equal to the number of clock cycles to complete the operation). Because of the circuit functioning, at any instant only $2N$ registers of the accumulator will contain useful data and only one every N additions is meaningful. A lot of space is, therefore, wasted by registers that for most of the time do not contain meaningful data.

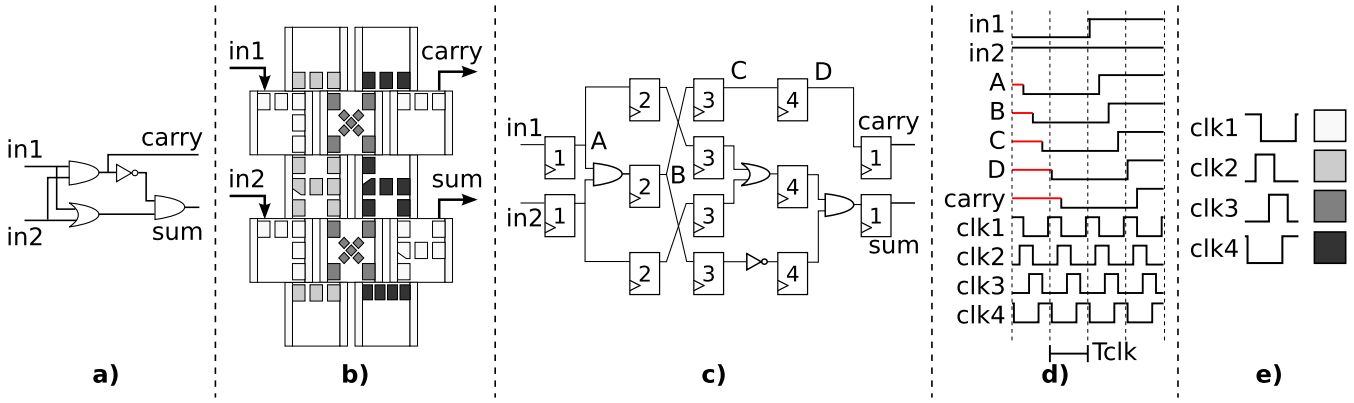

Fig. 6. Half adder example. (a) Scheme. (b) ME-NML circuit. (c) CMOS circuit described by the RTL model. (d) Timing. (e) Clocking system.

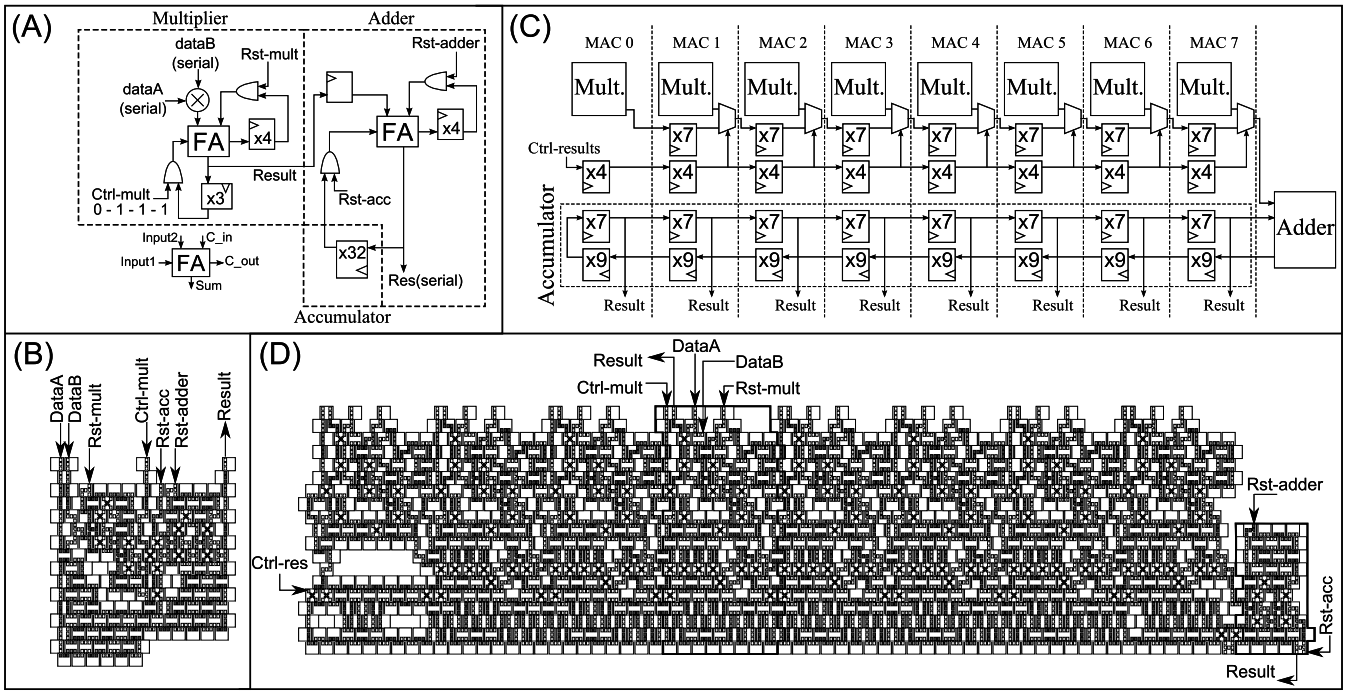

Fig. 7. Serial MAC (4 bit) and serial MAC with sharing (8 bit). (a) Serial MAC scheme. (b) Serial MAC ME-NML circuit. (c) Serial MAC with sharing scheme. (d) Serial MAC with sharing ME-NML circuit.

To address this problem, we propose to reduce the great impact of the accumulator letting multiple MAC units share the same accumulator and adder. In principle, while each multiplier works on one multiplication for all the time, the adder and the accumulator exploit the data interleaving concept.

The scheme in Fig. 7(c) and its ME-NML implementation in Fig. 7(d) contain eight 8-bit serial MAC units with shared accumulator and adder. Even though the eight MAC units are connected together, they can be treated independently of each other, with their own inputs and outputs.

The presented serial MAC, unlike the other implementations, is not modular, and it is indeed not scalable. All the feedback loops increase in length together with the number of bits, affecting radically the circuit layout. Changing the parallelism, the MAC requires to be redesigned; for this reason, we only designed and simulated the 4- and 8-bit serial MAC with shared resources. However, we were able to identify the main sources of area increase and make projections for the 16- and 32-bit serial MAC.

Since the multiplier processes a continuous flow of data, for this implementation it is not necessary to use the interleaving technique. Table I contains the main information concerning timing. The throughput is $1/(2N^2 \cdot T_{\text{clk}})$ for both implementations of the serial MAC. However, the version with shared accumulator and adder requires N MAC units to be linked together; therefore, the throughput for the entire shared-accumulator serial MAC is $1/(2N \cdot T_{\text{clk}})$. The latency for the first serial MAC implementation is equal to $9T_{\text{clk}}$. While the MAC's body does not change increasing the number of bits, the feedbacks of multiplier and adder do. Therefore, inputs and outputs need slightly more time to reach the MAC's central body, because they need to cross the area occupied by those

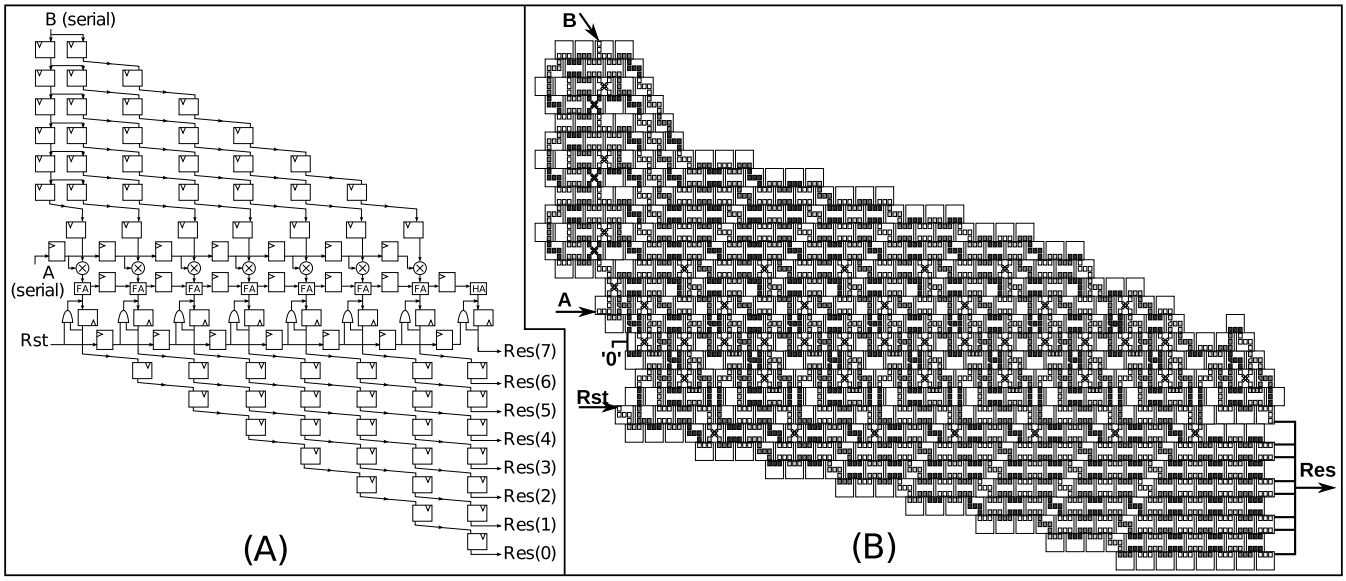

Fig. 8. Serial-parallel MAC (4 bit). (a) Scheme. (b) ME-NML circuit.

feedbacks. A simple calculation leads to the following latency as function of the parallelism: latency $= (N/4+8)\cdot T_{\text{clk}}$. The serial MAC with shared resources has the following latency: $(2N^2+9)\cdot T_{\text{clk}}$.

B. Serial–Parallel MAC

The idea for the second version of the MAC was to create a circuit organized as a 1-D systolic array of elements. We refer to this circuit as a serial–parallel MAC, because it has serial inputs and parallel output. Being a 1-D systolic array, the circuit's body itself has excellent characteristics but its input–output protocol is so unique that it would be very difficult to interface it directly with other devices. For this reason, additional interconnections are required, terribly spoiling performance.

The scheme in Fig. 8(a) depicts the 4-bit serial-parallel MAC. The preskew (top) and deskew (bottom) networks make it possible to have serial inputs and parallel outputs. The circuit counts $2N$ 1-bit adders. Each adder has its own feedback, so that the array of FA can work as an accumulator. A reset signal allows to reset the accumulator whenever necessary. The scheme is fully pipelined to imitate ME-NML behavior.

The timing protocol follows the handmade multiplication procedure, where the N partial products are evaluated one at a time and summed together. Evaluating all the partial products only requires N clock cycles. However, additional N cycles have to be spent feeding zeros to prepare the circuit for the next operation.

Fig. 8(b) depicts the ME-NML realization for the 4-bit serial–parallel MAC. The main element is a full adder with a 3 clock cycles feedback loop. Like in the earlier cases, the feedback is the critical path that decides the delay required between inputs. In this case, an input bit has to be fed every 3 clock cycles, hence, the maximum throughput can be reached with a three-operations interleaving. The circuit in Fig. 8(b) is divided into four main regions. To construct the generic MAC, each region has been treated separately. First, we isolated a set of recurrent blocks for each region, and then, we investigated how to organize them so that by combining them properly, it is possible to create a parametric MAC described by a single VHDL entity. Inputs A and B give their bits serially with a delay of 3 clock cycles between them. Then, the time required to provide all the bits is $3\times N\times T_{\text{clk}}$. After that, for another $3\times N\times T_{\text{clk}}$, the inputs are set to 0, until a new operation starts. The throughput would be equal to one operation every $3\times 2\times N$ clock cycles, but exploiting the interleaving technique, it goes up to $1/(2N\times T_{\text{clk}})$.

C. Parallel MAC

The last implementation presented is a fully parallel version of the MAC unit. It is basically composed by a parallel multiplier and an adder with feedback. The accumulator is embedded in the feedback, as ME-NML is intrinsically pipelined. The array multiplier and the ripple carry adder (RCA) have been chosen as components of the parallel MAC, because they both have a systolic array architecture.

The scheme of the 4-bit array multiplier and the 8-bit RCA composing the parallel MAC are drawn in Fig. 9(b). The two inputs A and B are given in parallel, just like the output Res. Notice how the multiplier is basically a matrix of full adders, so it is 2-D and its area grows quadratically with the circuit parallelism. The circuit arrangement and orientation imitate the ME-NML implementation [shown in Fig. 9(a)].

For the design of a generic N-bit parallel MAC, we defined a set of basic blocks for multiplier, adder, and interconnections. They can be assembled to create a MAC of any parallelism ($\geq$4 bits). The interconnection regions assure inputs and outputs synchronization: bits of the same signal can be fed and acquired simultaneously, guaranteeing the easiest possible interface protocol with other devices.

The array multiplier is composed by a matrix of $N\times(N-1)$ base blocks. By increasing the circuit parallelism, the matrix

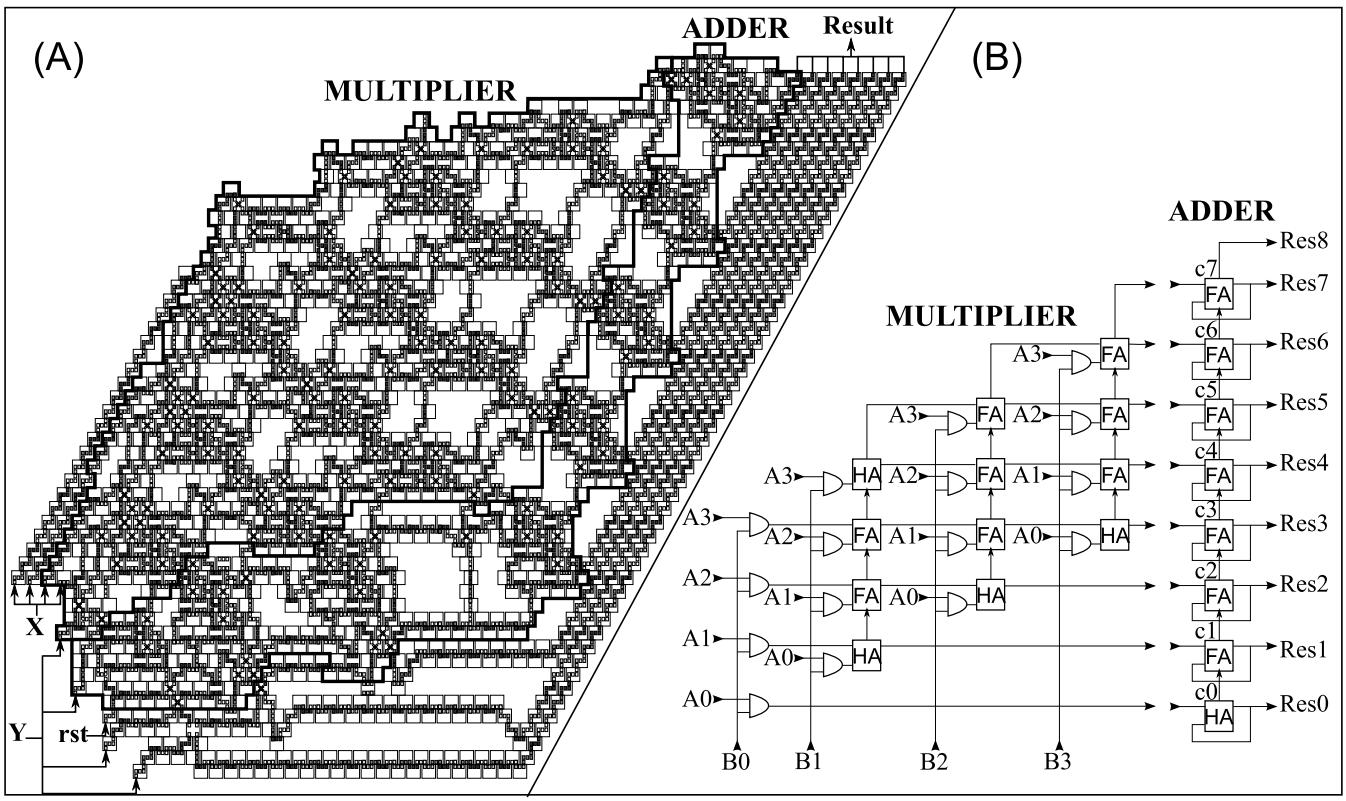

Fig. 9. 4-bit parallel MAC composed by array multiplier and RCA. (a) ME-NML circuit. (b) Scheme.

gets bigger affecting the overall circuit latency. On the other hand, for any number of bits, the RCA is always one column thick, having a constant impact on the latency. Every block of the multiplier requires 5 clock cycles to be crossed horizontally (signal A) and 2 vertically (signal B). Therefore, the inputs (bottom left) need $(5(N-1)+2N+5)\cdot T_{\text{clk}}$ to reach the result. In a MAC, each multiplication result is added to the value in the accumulator. Since each block of the adder has a 5 clock long feedback loop, the operations cannot be fed to the MAC in a continuous flow. Two operations must be fed with 5 cycles delay in order for them to be added to each other. Therefore, to reach the maximum throughput (one operation per clock cycle), five uncorrelated operations must be interleaved.

TABLE I
COMPARISON OF THE MAC IMPLEMENTATIONS IN TERMS OF MAXIMUM THROUGHPUT, MAXIMUM INTERLEAVING, AND LATENCY

Latency and Throughput			4 bit	8 bit	N bit
Latency (T_{clk})		Serial	9	10	$N/4+8$
		Serial with sharing	45	85	$2N^2+9$
		Serial-Parallel	36	76	$6(N-1)+4N+2$
		Parallel	28	56	$5(N-1)+2N+5$
Throughput ($N_{outputs}/T_{clk}$)	With Interleaving	Serial	1/32	1/128	$1/2N^2$
		Serial with sharing	1/32	1/128	$1/2N^2$
		Serial-Parallel	1/8	1/16	1/2N
		Parallel	1	1	1
	No Interleaving	Serial	1/32	1/128	$1/2N^2$
		Serial with sharing	1/32	1/128	$1/2N^2$
		Serial-Parallel	1/24	1/48	1/6N
		Parallel	1/5	1/5	1/5
Maximum Interleaving ($Noperations$)		Serial			1
		Serial with sharing			1
		Serial-Parallel			3
		Parallel			5

V. RESULTS

This section presents the performance outcomes for the MAC unit implementations proposed. The serial MAC depicted in Fig. 7(b) and the one with shared adder and accumulator of Fig. 7(d) are considered as two distinct circuits. The four architectures are first examined in terms of throughput and latency.

The computation of throughput and latency has been presented for each architecture in Section IV. Results are summarized in Table I. From this comparison, it is clear that the parallel solution has the best results, in particular, if interleaving is applied. The same is true when considering the latency, even though for small number of bits, the serial with sharing implementation has some advantage over the others.

For a complete comparison, it is necessary to understand if the advantages of parallel MAC in throughput and latency are balanced by worse performance in terms of area occupation and power dissipation.

Table II puts side by side the two serial implementations, showing that as expected, the one with sharing improves area occupation and power consumption, becoming the serial implementation of choice. Fig. 10 depicts area and power results by considering isothroughput circuits for the serial MAC with sharing together the serial–parallel and parallel MACs. The comparison is performed both with and without exploiting the interleaving technique. To reach their maximum throughput, both parallel MAC and serial–parallel MAC necessitate the interleaving technique. Moreover, for a fair comparison, in term of area and power, each implementation should have the same throughput, but that is not the case. The output rate of the parallel MAC has been used as reference

TABLE II

RESULTS OF OCCUPIED AREA AND POWER CONSUMPTION FOR THE TWO SERIAL MAC IMPLEMENTATIONS

Area and Power			Number of bits			
			4	8	16	32
With Interleaving	Area (μm^2)	Serial	388	3,528	43,941	639,013
		Serial with sharing	291	1,300	8,030	55,300
	Power (μW)	Serial	35.0	320	4,006	58,323
		Serial with sharing	26.3	117	724	4,990
No Interleaving	Area (μm^2)	Serial	78	706	8,788	127,803
		Serial with sharing	58	260	1,610	11,100
	Power (μW)	Serial	7.0	64.1	801	11,665
		Serial with sharing	5.3	23.5	145	998

for both the cases: with and without interleaving. Therefore, we combined as many MAC modules as needed to reach a throughput rate equal to 1. For example, since the serial MAC has throughput $1/(2N^2)$, the area and power of a single serial MAC have been multiplied by $(2N^2)$ as if $(2N^2)$ MAC units were working together to achieve a $1/1$ throughput. Thus, in Table II, the results of parallel MAC are simply those of a single unit. The results of the other implementations have been multiplied by a coefficient, which is the number of units that should work in parallel to reach the same throughput as the parallel MAC.

In the following, the results of each implementation are analyzed in detail.

A. Serial MAC

Even if only the 4- and 8-bit serial MAC with sharing have been designed and simulated, it was trivial to obtain a projection of the number of cells for the other parallelisms. What varies with the number of bits are all the feedback loops, the accumulator and, only for the version with sharing, the shift register that brings the products to the adder. In each single MAC block, to obtain the $2N$-bit circuit from the N-bit one, the multiplier's loops must get N clock periods longer. The same is true for the segment of the products' shift register and for each of the two segments of the accumulator. From these considerations, it was possible to predict with good approximation the growth of the serial MAC with the number of bits.

In Table II, the *serial with sharing* rows of data refer to the whole circuit with shared adder and accumulator. Since such circuit contains N MAC units, to get an idea of the weight of a single MAC, the total number of cells has been divided by the number of bits, which is also the number of MAC blocks enclosed by the entire circuit. Area and power are the effective values for a single MAC unit, not those for the whole structure containing N of them. The results show that the idea of sharing accumulator and adder has been fruitful, with an area and power losses reduction of more than 11 times for the 32-bit circuit. However, the throughput of the serial MAC decreases quadratically with the number of bits. Therefore, ideally, to keep up with the parallel MAC performance, the increase rate of a single MAC should be equal to 1. Unfortunately, this is clearly not the case.

B. Serial–Parallel MAC

Since the serial–parallel MAC layout is very compact, the area calculated by the VHDL model corresponds to the actual space occupied by the circuit. So increase rates of area and power are pretty much the same as they are both proportional to the number of cells. The body and the interconnection parts grow differently as the number of bits increases. As expected, the body and the input conditioning expand linearly, while the interconnection regions grow quadratically.

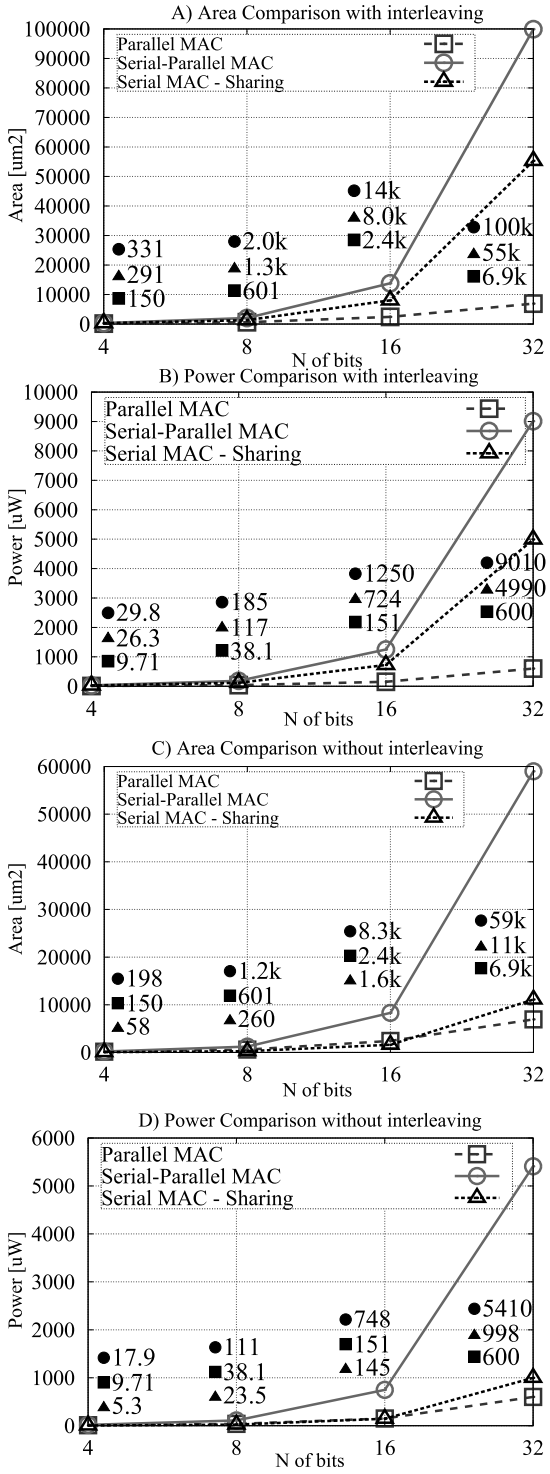

Fig. 10. Comparison among the three MAC implementations. (a) Area for throughput = 1 exploiting interleaving. (b) Power for throughput = 1 exploiting interleaving. (c) Area for throughput = 1/5 without interleaving. (d) Power for throughput = 1/5 without interleaving.

The ratio total area/body area goes from 1.9 for 4 bits to 16.2 for 64 bits. This result gives an idea of how much input–output preskew/deskew networks can affect performance and circuit area.

C. Parallel MAC

The layout of this circuit, as clear from Fig. 9, has many empty internal regions. So the area evaluated by the model is smaller than it should be, because it only considers the space occupied by cells. The value actually assigned to the parallel MAC is rounded up to the parallelogram circumscribed to the circuit. To obtain the parallelogram area, we derived a generic equation to evaluate height and width (in terms of cells) for any number of bits.

When the number of bits doubles, both the area and power grow four times. Hence, as expected the increase rate is quadratic and regular, because is how both the multiplier and the interconnection regions grow. The wasted space due to the empty inner regions has a negative effect on the area occupation, while it does not affect the power consumption.

Referring to Table II, starting from the *with interleaving* part, the parallel implementation is undoubtedly the most efficient, while the serial–parallel has the worst outcomes.

If interleaving cannot be used (e.g., if there are no unrelated operations to be executed), the hierarchies among the four MAC versions undergo slight changes. The performance of the parallel and serial–parallel MAC units worsens, respectively, of five and three times, according to their previous interleaving usage. The serial MAC with sharing gains a lot in this situation, because it cannot exploit interleaving anyway. Therefore, referring to Fig. 10, looking at data for the circuit that cannot exploit interleaving (C and D), we can notice that the serial MAC with sharing is the best architecture up to a 16-bit parallelism (the value for area and power is smaller for serial architecture—labeled with the triangle symbol—with respect to the other two). However, the trend for area and power are worse in serial and serial–parallel with respect to parallel architecture. Thus, none of the implementations can keep up with the parallel one when the number of bits increases. The parallel MAC is again the best solution for 32 or higher number of bits.

VI. Conclusion

We stated in the introduction that we wanted to understand if there was the necessity for a paradigm change from parallel to serial with the change of technology from CMOS to NML. Achieved results undoubtedly mark parallel architecture as the best in all metrics also in NML technology.

Indeed, in particular for high parallelisms, the parallel MAC results are overall better than the other two implementations, while for smaller circuits, the performance is comparable. The serial MAC can be a valid alternative when unable to provide interleaved operations to the circuit. The idea of sharing accumulator and adder boosts the performance, but it can also be a setback, as it requires N multiple MAC modules to be connected together. It is not possible for one of them to function without the presence of the others. In conclusion, the parallel MAC has by far the most promising architecture organization.

In addition, some important advices for NML logic circuits design have emerged from this paper.

1) Synchronization networks like preskew/deskew greatly affect performance. Indeed, the serial–parallel MAC greatly suffers the input–output conditioning networks. The only way for this circuit to be very competitive would be a system where it could interface itself with other modules without the need of its additional preskew/deskew networks.
2) Long interconnections and feedbacks are the main problem with the ME-NML technology. The systolic array organization of the parallel MAC keeps them to the minimum, avoiding for the long feedback required for serial multiplication that in the serial MAC also affect the loops of adder and accumulator.
3) Standard cells design can give an added value to the automatic design of NML circuits. The definition of the standard cell library, together with the high regularity of circuit layout, can be the foundation for the creation of a design tool that could greatly improve the future research in this field. The design and simulation methodology proposed for this paper consists of a hierarchical RTL model, based on the set of standard cells. The model contains all the information concerning the physical placement and orientation of cells. Furthermore, the embedded capability of exact performance evaluation makes the model an advanced stand-alone tool.

References

[1] D. Rairigh, "Limits of CMOS Technology scaling and technologies beyond-CMOS," Michigan State Univ., Lansing, MI, USA, Tech. Rep., 2005.
[2] M. Vacca *et al.*, "Electric clock for NanoMagnet logic circuits," in *Field-Coupled Nanocomputing* (Lecture Notes in Computer Science), N. G. Anderson and S. Bhanja, Eds. Heidelberg, Germany: Springer-Verlag, 2014.
[3] M. Vacca *et al.*, "NanoMagnet logic: An architectural level overview," in *Field-Coupled Nanocomputing*. Berlin, Germany: Springer, 2014, pp. 223–256.
[4] G. Causapruno, G. Urgese, M. Vacca, M. Graziano, and M. Zamboni, "Protein alignment systolic array throughput optimization," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 1, pp. 68–77, Jan. 2015.
[5] 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, Jul. 2015.
[6] 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, pp. 368–377, May 2013.
[7] M. Crocker, X. S. Hu, and M. Niemier, "Design and comparison of NML systolic architectures," in *Proc. NANOARCH*, Jun. 2010, pp. 29–34.
[8] H. Kung, "Systolic algorithms for the CMU warp processor," Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, USA, Tech. Rep., 1978.
[9] 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, Mar. 2013.
[10] L. Lu, W. Liu, M. O'Neill, and E. Swartzlander, "QCA systolic matrix multiplier," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Oct. 2010, pp. 149–154.
[11] D. Giri, M. Vacca, G. Causapruno, M. Zamboni, and M. Graziano, "Modeling, design, and analysis of magnetoelastic nml circuits," *IEEE Trans. Nanotechnol.*, vol. 15, no. 6, pp. 977–985, Nov. 2016.
[12] D. S. Silva, L. H. B. Sardinha, M. A. M. Vieira, L. F. M. Vieira, and O. P. V. Neto, "Robust Serial Nanocommunication With QCA," *IEEE Trans. Nanotechnol.*, vol. 14, no. 3, pp. 464–472, May 2015.

[13] H. Cho and E. E. J. Swartzlander, "Serial parallel multiplier design in quantum-dot cellular automata," in *Proc. 18th IEEE Symp. Comput. Arithmetic (ARITH)*, Jun. 2007, pp. 7–15.
[14] H. Cho and E. E. Swartzlander, "Adder and multiplier design in quantum-dot cellular automata," *IEEE Trans. Comput.*, vol. 58, no. 6, pp. 721–727, Jun. 2009.
[15] M. Gladshtein, "Quantum-dot cellular automata serial decimal adder," *IEEE Trans. Nanotechnol.*, vol. 10, no. 6, pp. 1377–1382, Jun. 2011.
[16] V. Pudi and K. Sridharan, "A bit-serial pipelined architecture for high-performance DHT computation in quantum-dot cellular automata," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 10, pp. 2352–2356, Oct. 2015.
[17] D. Giri, M. Vacca, G. Causapruno, W. Rao, M. Graziano, and M. Zamboni, "A standard cell approach for MagnetoElastic NML circuits," in *Proc. IEEE/ACM Int. Symp. Nanosc. Archit. (NANOARCH)*, Jul. 2014, pp. 65–70.
[18] M. Vacca *et al.*, "ToPoliNano: NanoMagnet logic circuits design and simulation," in *Field-Coupled Nanocomputing* (Lecture Notes in Computer Science), N. G. Anderson and S. Bhanja, Eds. Berlin, Germany: Springer-Verlag, 2014, pp. 274–306.
[19] F. Riente, G. Turvani, M. Vacca, M. R. Roch, M. Zamboni, and M. Graziano, "ToPoliNano: A CAD tool for nano magnetic logic," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 7, pp. 1061–1074, Jul. 2017.
[20] R. P. Cowburn and M. E. Welland, "Room temperature magnetic quantum cellular automata," *Science*, vol. 287, no. 5457, pp. 1466–1468, 2000.
[21] M. T. Niemier *et al.*, "Shape engineering for controlled switching with nanomagnet logic," *IEEE Trans. Nanotechnol.*, vol. 11, no. 2, pp. 220–230, Mar. 2012.
[22] G. Csaba and W. Porod, "Behavior of nanomagnet logic in the presence of thermal noise," in *Proc. IEEE Int. Workshop Comput. Electron.*, Pisa, Italy, Oct. 2010, pp. 1–4.
[23] M. Vacca *et al.*, "Magnetoelastic clock system for nanomagnet logic," *IEEE Trans. Nanotechnol.*, vol. 13, no. 5, pp. 963–973, Sep. 2014.
[24] M. T. Niemier *et al.*, "Nanomagnet logic: Progress toward system-level integration," *J. Phys., Condens. Matter*, vol. 23, no. 49, p. 34, Nov. 2011.

Davide Giri received the M.S. degree in electronic engineering from the Politecnico di Torino, Turin, Italy, in 2014 and the M.S. degree in electrical and computer engineering from the University of Illinois at Chicago, Chicago, IL, USA, in 2015. He is currently working toward the Ph.D. degree at Columbia University, New York, NY, USA, where he is a part of the System-Level Design Group.

His current research interests include range from emerging technologies and circuits architectures to heterogeneous system-on-chip and distributed embedded systems.

Giovanni Causapruno received the D.Eng. and Ph.D. degrees in electronics engineering from the Politecnico di Torino, Turin, Italy, in 2012 and 2016, respectively. He is currently working toward the M.B.A. degree at the SDA Bocconi School of Management, Milan, Italy.

His current research interest includes parallel processing architectures for nanotechnologies.

Fabrizio Riente (M'18) received the M.Sc. degree *(magna cum laude)* in electronic engineering and the Ph.D. degree from the Politecnico di Torino, Turin, Italy, in 2012 and 2016, respectively.

In 2016, he was a Postdoctoral Research Associate at the Technical University of Munich, Munich, Germany. He is currently a Postdoctoral Research Associate at the Politecnico di Torino. His current research interests include device modeling, circuit design for nanocomputing, with particular interest on magnetic quantum-dot cellular automata, and the development of an electronic design automation tool for beyond-CMOS technologies, with a focus on the physical design.