POLITECNICO DI TORINO
Repository ISTITUZIONALE

NanoMagnet Logic: an Architectural Level Overview

Original
NanoMagnet Logic: an Architectural Level Overview / Vacca, Marco; Graziano, Mariagrazia; Wang, JUAN CHI; Cairo, Fabrizio; Causapruno, Giovanni; Urgese, Gianvito; Biroli, ANDREA DARIO GIANCARLO; Zamboni, Maurizio (LECTURE NOTES IN COMPUTER SCIENCE). - In: Field Coupled NanocomputingSTAMPA. - Berlin : Springer, 2014. - ISBN 978-3-662-43722-3. - pp. 223-256 [10.1007/978-3-662-43722-3 10]

Availability:
This version is available at: 11583/2525155 since: 2020-10-24T19:26:26Z

Publisher:
Springer

Published
DOI:10.1007/978-3-662-43722-3 10

Terms of use:

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

Publisher copyright
Springer postprint/Author's Accepted Manuscript (book chapters)

This is a post-peer-review, pre-copyedit version of a book chapter published in Field Coupled Nanocomputing. The final authenticated version is available online at: http://dx.doi.org/10.1007/978-3-662-43722-3 10

(Article begins on next page)

01 September 2025

NanoMagnet Logic: an Architectural Level Overview

Marco Vacca[1], Mariagrazia Graziano[1], Juanchi Wang[1], Fabrizio Cairo[1], Giovanni Causapruno[1], Gianvito Urgese[2], Andrea Biroli[1], and Maurizio Zamboni[1]

[1] Dipartimento di elettronica e telecomunicazioni, Politecnico di Torino, Italy,
[2] Dipartimento di Ingegneria informatica e dei sistemi, Politecnico di Torino, Italy

Abstract. In recent years Field-Coupled devices, like Quantum dot Cellular Automata, are gaining an ever increasing attention from the scientific community. The computational paradigm beyond this device topology is based on the interaction among neighbor cells to propagate information through circuits. Among the various implementations of this theoretical principle, NanoMagnet Logic (NML) is one the most studied, due to some interesting features, like the possibility to combine memory and logic in the same device and the possible low power consumption. Since the working principle of Field-Coupled devices is completely different from CMOS technology, it is important to understand all the implications that this new computational paradigm has on complex circuit architectures.

In this chapter we deeply analyze the major issues encountered in the design of complex circuits using Field-Coupled devices. Problems are analyzed and techniques to solve them and to improve performance are presented. Finally, a realistic analysis of the applications best suited for this technology is presented. While the analysis is performed using NanoMagnet Logic as target, the results can be applied to all Field-Coupled devices. This chapter therefore supplies researchers and designers with the essential guidelines necessary to design complex circuits using NanoMagnet Logic and, more in general, Field-Coupled devices.

1 Introduction

With the conception of Quantum dot Cellular Automata (QCA) [1] technology, a completely new computational principle to process and propagate information was presented to the scientific community. Signals were no more represented by voltage or current levels, but by charge configurations. Circuits are made by many identical cells and information processing is driven by electrostatic interaction among neighbor cells [2]. Different practical implementations of this principle were proposed, the two most promising are Molecular QCA [3] and Magnetic QCA also called NanoMagnet Logic [4]. Molecular QCA [5][6][7][8] use complex molecules as basic cells. The main interest for this QCA implementation resides in the amazing clock speed (1THz) that can be theoretically reached

[9], however, the experimental fabrication is extremely challenging with current technological processes. NanoMagnet Logic uses instead single domain nanomagnets (Figure 1.A) as basic cells [10]. Due to magnetic properties, like shape or magnetocrystalline anisotropy, magnets can have only two stable states that are used to represent logic values '0' and '1'. The main interest around this technology lies in its magnetic nature, that allows to potentially mix logic and memory in the same device. For the same reason circuits built with this technology have an high resistance to radiations and heat and, potentially a power consumption lower than that of state-of-the-art CMOS circuits [11].

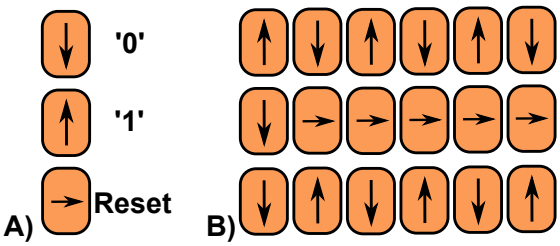

Fig. 1. A) Single domain nanomagnets are used as basic cell in NML technology. Only two stable states are possible and are therefore used to represent digital values '0' and '1'. A third intermediate state is possible but it is unstable and can therefore be forced only by external means. B) To switch magnets from one state to the other a RESET mechanism is required. Magnets are driven in the RESET state with an external mean, a magnetic field in the most classical implementation. When the magnetic field is removed, magnets realign following the input element.

Information propagation depends on the magnetostatic interaction among neighbor magnets. Nonetheless, the magnetic field generated by one magnet is not strong enough to switch a neighbor element from one stable state to the other. As a consequence a mechanism called clock must be used to help magnets switching, forcing them in an intermediate unstable state (RESET in Figure 1.A). An external mean (a magnetic field in the most classical implementation [12][13]) must be used to force magnets in the RESET state (Figure 1.B). When the magnetic field is removed magnets realign following the input element, propagating therefore the information through the circuit.

Signals propagate therefore with a Domino-like effect, where magnets switch one by one in a sort of chain reaction. Unfortunately the number of magnets that switch correctly in a chain is limited by the intrinsic instability of the RESET state. The RESET state is unstable, so external stimuli (like thermal noise [14]) can generate unwanted switches, leading therefore to errors during signals propagation. To solve this problem a multiphase clock system must be used [15][16]. Circuits are divided in small areas called clock zones. Every clock zone is made by a limited number (less than 5 in the ideal case [14]) of cascaded elements. A different clock signal is applied to every clock zone. Different clock

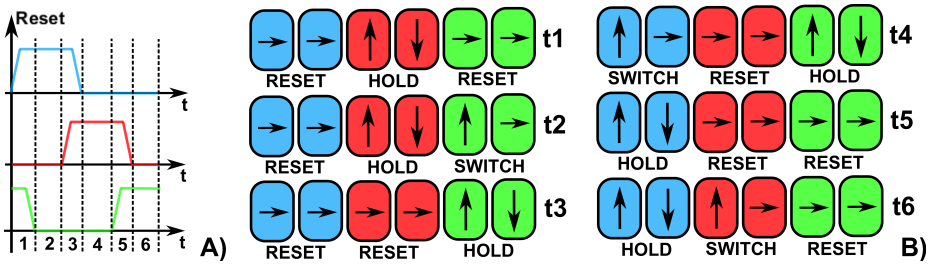

Fig. 2. 3-phase overlapped clock system. A) Clock signal waveforms. 3 clock signals with a phase difference of 120°, partially overlapped, are used to assure a correct signals propagation. B) Detailed signal propagation through a simple NML wire. 6 different time step can be identified thanks to this clock system. When magnets of a clock zone are switching (SWITCH) magnets on their left are in the HOLD state and act as an input, while magnets on their right are in the RESET state so they have no influence.

schemes can be used. One of the most simple uses three overlapped clock signals [17], with a phase difference of 120° (Figure 2.A). Figure 2.B shows an example of NML wire and how its state changes according to the clock signals applied. Three different states can be identified: HOLD when no clock signal is applied, RESET when the clock signal is applied and SWITCH, when the clock signal is slowly removed. When magnets of a clock zone are switching (SWITCH state), they see on their left magnets that are in the HOLD state and therefore act as an input, while magnets on the right are in the RESET state and they have no influence on the signals propagation. To obtain an errorless signal propagation magnets of a clock zone must be forced in the RESET state before neighbor magnets start to switch (see Figure 2.B). This is obtained overlapping the clock waveforms as described in detail in [17].

1.1 Designing Circuits With Clock Zones Constraints

The clock mechanism is central to the entire NML (and QCA) technology and it is the source of the main problems encountered at architectural level. Particularly in case of NML implementation four possible clock mechanism can be identified, as shown in Figure 3. The first mechanism uses a magnetic field generated by a current flowing through a wire placed under magnets plane [13], as shown in Figure 3.A. This clock system was experimentally demonstrated in [18]. The second mechanism uses Spin-Torque coupling with a current flowing through the magnets themselves [19][20][21], as can be seen from Figure 3.B. In this case the basic element is no more a simple magnet but it is a Magneto-Tunnel Junction (MTJ). The third clock mechanism, shown in Figure 3.C, uses the mechanical stress induced by the strain of a piezoelectric layer to force magnets in the RESET state [22][23]. In this case the clock mechanism is no more based on a current but on a voltage. Finally a 4th clock topology is available as shown in Figure 3.D. In this case magnets are made by Cobalt/Platinum multilayered

structures where the magnetization lies out of plane [24][25][26]. In this case the clock is an oscillating magnetic field applied perpendicular to the plane. No clock zones are used because the magnetic field is applied globally to the circuit.

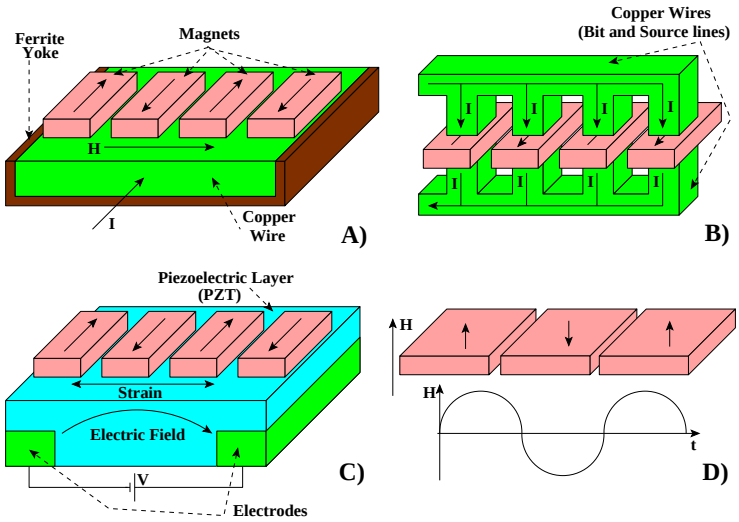

Fig. 3. NML main clock systems. A) Magnetic field clock. A magnetic field is generated by a current flowing through a wire placed under the magnets plane. B) STT clock. Spin-Torque coupling due to a current flowing through the magnets, which in this case are MTJ junctions. C) Magnetoelastic clock. The mechanical stress induced by the strain of a piezoelectric layer at which a voltage is applied forces magnets in the RESET state. D) Out-of-plane NML clock. In this particular implementation of NML, the magnetization of each dots points out-of-plane. In this case the clock is an oscillating magnetic field applied perpendicularly to the plane. No clock zones are present because clock field is applied uniformly to the entire circuit.

The clock generation system is important from the circuits architecture point of view because it has a strong impact on the layout of the clock zones. Particularly, to drive the currents and the voltages required by different clock systems, wires must be properly routed through the circuit. Thanks to the size of these wires and the limitations of the technological processes available, not every clock zones layout is possible. Different clock systems have different constraints. For example, relatively to the magnetic field clock we have developed the "snake-clock" system [15][17]. The clock zones layout is outlined in Figure 4.A. Clock zones are made by parallel strips that represent the physical wires used to generate the magnetic field. To propagate correctly, signals must be routed through a precise sequence of clock zones, from zone 1 to zone 2 and finally to zone 3. As can be seen from Figure 4.A, with this clock zone layout magnets can propagate in only one direction, from left to right. Moreover, signals cannot propagate vertically in the same clock zone because the number of magnets that

can be cascaded is quite limited [14][27]. As a consequence vertical signals follow a stair-like propagation (see Figure 4.A). This clock system is called "snake clock" because the vertical signals propagation recalls the movement of a snake. To propagate signals in the opposite direction, from right to left it is instead necessary to switch the order of phase 2 and 3. This is possible locally twisting the correspondent wires as shown in Figure 4.B. Wires are physically located on different planes [13] so they can be twisted freely without interferences. Magnets cannot be placed in the area correspondent to the clock wires crossing. We have demonstrated in [28][29] that this clock system is immune to possible crosstalk between neighbor clock wires.

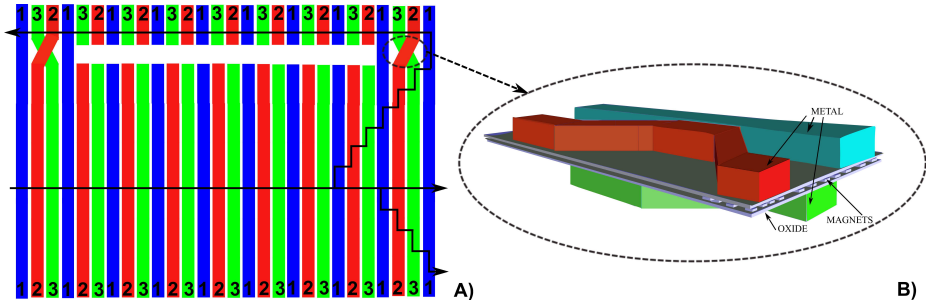

Fig. 4. Snake clock system. A) Clock zones are made by parallel strips that correspond to the clock wires used to generate the magnetic field. B) To allows signals propagation wires are twisted when signals need to propagate in different directions. The wire twisting is possible because wires are alternatively placed over and under magnets plane.

Other clock solutions are possible, for example in [30] a more simple 2-phases clock was presented. Moreover other clock systems will have different clock constraints. However two important things must be understood: First, the clock zones layout must be chosen carefully, according to the technological processes used to fabricate the clock generation network and to the requirements of the NML signals propagation. The second important fact to be considered are the constraints and limitations generated by the chosen clock system, that must be carefully taken into account in the circuits design. More in general this is true also for other QCA implementations.

1.2 Problems and Solutions

While the clock mechanism influences the circuits layout differently, depending on the particular system chosen, it has a second more important consequence that is shared by all kind of NML and QCA implementations. Considering a N-phase clock system, every group of N consecutive clock zones has a delay of exactly one clock cycles and it is therefore equivalent to a CMOS register. A

QCA wire can therefore be seen as a shift register [17][31], leading therefore to an intrinsic pipelined behavior. Pipelining is a common feature also in CMOS technology but in this case there are two important differences: in CMOS the level of pipelining is generally small and moreover it is a parameter chosen by the designer, while in this case the level of pipelining is extremely high and it is not a parameter chosen by the designer but it depends on the circuit layout [32]. This is valid for all the clock implementations, also in case of the Out-of-plane NML clock (Figure 3.D). While with this clock system no clock zones exists, in a wire signals needs exactly one clock cycle to propagate through two magnets [26]. As a consequence a block of two magnets in a wire has a delay of one clock cycles.

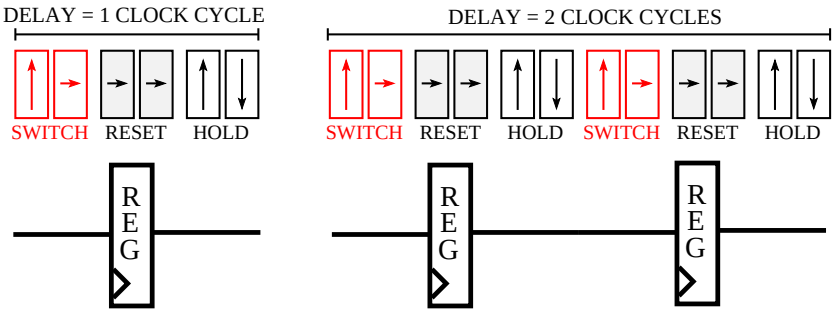

Fig. 5. NML intrinsic pipelining. Every group of 3 consecutive clock zones has a delay of 1 clock cycle and is therefore equivalent to a CMOS register.

In Field-Coupled technologies, like QCA and NML, three important problems can be identified at the architectural level. Some of then are already known. Here we assess them and we discuss thoroughly several alternative solutions, some of them previously mentioned and some newly proposed here.

- *LAYOUT=TIMING*. This is the first problem that arises thanks to the intrinsic circuits pipelining. The propagation delay of a signal in terms of clock cycles depends on the length of its correspondent wire in terms of clock zones. As a consequence mismatches in the wires length generates errors in the logic operations performed by the circuit. This problem is detailed analyzed in Section 2.
- *LOOPS*. This is the second problem due to the intrinsic pipelining. If a loop with a length of N clock cycles is present inside the circuit, the throughput is reduced of N times. Moreover serious synchronization problems arise. This problem is thoroughly studied in Section 3.
- *INTERCONNECTIONS*. This further issue is connected to both the Field-Coupling principle that represents the base of this technology, and to its intrinsic pipelining. In complex circuits a lot of area is wasted for interconnections, moreover long interconnections generates a long delay in signals propagation. This problem is described in Section 4.

It is important to further underline that while the results here presented are obtained using as a target technology NanoMagnet Logic, they apply with different extent to all Field-Coupled devices. This chapter shows therefore the guidelines that a designer should follow to effectively develop circuits with this technology.

2 Layout=Timing

The first important problem that a designer must face during the development of any NML circuit is called "layout=timing" [32]. Thanks to the multiphase clock system, every group of N consecutive clock zones has a delay of exactly one clock cycle. The main consequence is that the propagation delay is no more a designer choice but it depends on the circuit layout. However, synchronization issues can arise due to mismatch in the wires length and therefore in the propagation delay of signals. The situation is explained in Figure 6. Two input wires are connected to a NML AND gate. AND/OR gates can be obtained changing the shape of one magnets as shown in [30], and altogether with the majority voters [33][29] and inverter they represents the logic gates set available with this technology. In Figure 6.A input wires have a different length in terms of clock zones, therefore the two input signals arrive at the gate inputs with the difference of one clock cycle and the output of the gate is consequently wrong. To solve this problem the wires length must equalized, as shown in Figure 6.B. Signals are therefore perfectly synchronized and the gate output is correct.

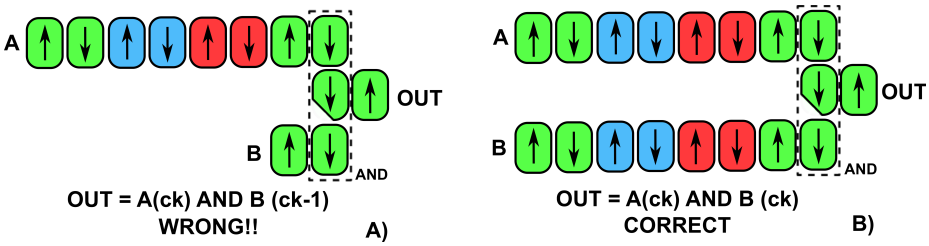

Fig. 6. Layout=Timing problem. Since the propagation delay of a signals depends on the wires length, if inputs wires of one gate have different lengths, their signals will arrive with different delays and the results will be wrong. Wires length must be equalized to synchronize signals and obtain the correct result.

Nonetheless, considering a complex circuit based on hundreds of thousands or millions of gates, the equalization of the length of every signal in it can be a nearly impossible task. In the following we discuss two different solutions, one at logic level and one at layout level. The first solution, described in Section 2.1, is based on the use of delay insensitive asynchronous logic to obtain automatic signals synchronization. The second solution, described in Section 2.2, is based on the introduction of constraints on the clock zones layout making it very regular, automatically equalizing the length of every wire in the circuit.

2.1 Asynchronous Logic

The first solution that can be adopted to solve the "layout=timing" problem is to use a different kind of logic, particularly a specific type of asynchronous logic called Null Convention Logic (NCL). In this logic [34][35] signals are coded using two bits (Figure 7.A). The logic '0' correspond to the coding '01' while logic '1' correspond to the coding '10'. '00' correspond to a particular state called NULL, while '11' is a not allowed state. Figure 7.B shows an example of NCL gate, called TH22, and its logic equation. The particularity of this kind of logic is that it is completely delay insensitive. The circuit behavior is depicted in Figure 7.C. Circuits switch periodically from DATA to NULL and from NULL to DATA, however the transition happens only when all inputs switch. That means that circuit will switch from NULL to DATA only when all inputs switch from NULL to DATA. Circuits will remain in the DATA state until at least one of the inputs is in the DATA state. The transition from DATA to NULL will happen only if and when all inputs will switch from DATA to NULL. Then the cycle will restart. As a consequence a complete delay insensitivity is reached, because it does not matter if there is a difference in the propagation delay of signals, the transition from one state to the other will occur only when the last signal arrives at the circuit inputs.

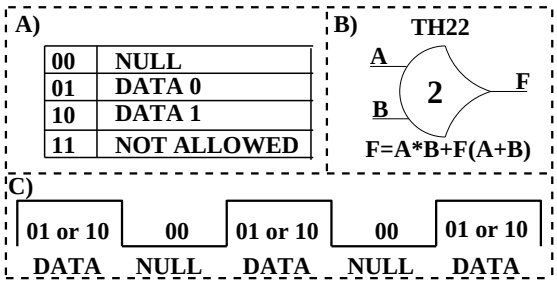

Fig. 7. Null Convention Logic (NCL). A) Signals are coded using two bits. B) Example of NCL gate and its logic equation. C) NCL circuits switch periodically from DATA to NULL state and then from NULL to DATA.

The delay insensitivity of NCL logic apparently make it perfect for NML circuits, and indeed this solution works correctly as demonstrated in [31][36]. However, it greatly reduces circuits performance. This happens also if CMOS technology is used as a target, because the high robustness of this logic comes at the price of a big area overhead. In particular, in case of NML circuits the price that must be paid to achieve complete delay independence is much higher. There are two reasons behind this fact: The signal coding and the communication protocol. With this logic signals are coded using two bits, that means that circuit area is at least two times higher with respect to a not-coded circuit, but in NML more area means more delay. NCL logic is also an asynchronous logic and as a

consequence it uses a communication protocol, which plays an important role in the overall performance losses. Figure 8 shows the communication protocol used. Circuits can be generically represented by a block of combinational logic embraced by two asynchronous registers which implement the communication protocol. The communication protocol works following the step explained in Figure 8. First a new DATA is sent to the circuit (Figure 8.A). When the DATA reaches the second register, an acknowledgment signal is sent back to the first register (Figure 8.B) which then sends a NULL to the circuit (Figure 8.C). When the NULL is received by the second register, an acknowledgment is sent back to the first register (Figure 8.D) and a new cycle can start. While this communication protocol works very well, it requires a signal to traverse the circuit four times. As a consequence every DATA cycle requires a time equal to four times the signal propagation delay of the circuit, and this explains the high losses of performance due to the use of NCL logic.

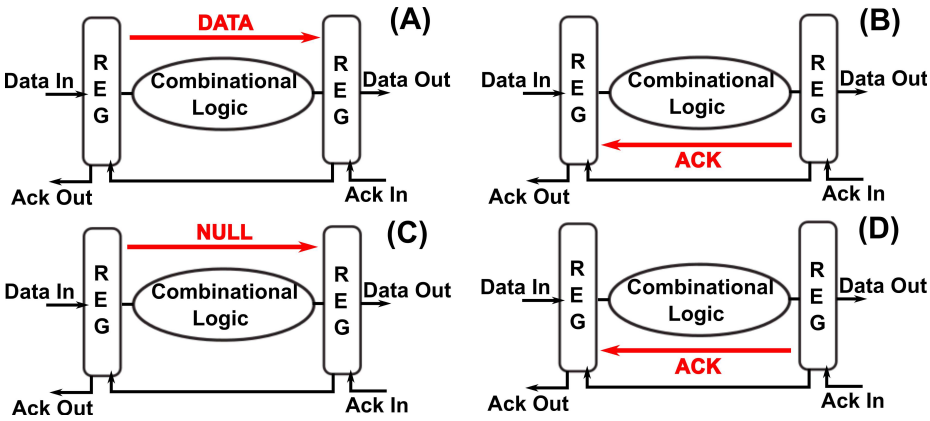

Fig. 8. NCL asynchronous protocol. A) A DATA is sent to the circuit. B) An acknowledgment that states the reception of the DATA is sent back. C) A NULL is sent to the circuit. D) An acknowledgment that states the reception of the NULL is sent back.

Other types of asynchronous logic with a simplified communication protocol can be studied and possibly adopted to improve performance. However, asynchronous logic is based by definition on a communication protocol which uses messages between units to assure signals synchronization. The problem is that in this kind of technology the delay penalty along interconnections is very high. As a consequence in [36] we have developed a new mixed synchronous-asynchronous protocol to exploit the maximum potential from this technology. The protocol is described in Figure 9. It is based on simple boolean logic gates with no signals encoding, thus greatly reducing the area overhead. Asynchronous registers are substituted by a synchronization block shown in Figure 9, which is simply a multiplexer with the output connected to one of its inputs. Normally the Selection

bit (*Sel* in Figure 9) is at logic '0', that means that the multiplexer is in memory state and its output is constant. Every N clock cycles a new data is sent to the multiplexer input followed by an enable signal which is connected to the *Sel* pin of the multiplexer. As a consequence the multiplexer samples the new data and after it returns in the memory state. The consequence is that a new data is sent to the circuit every N clock cycles and in the middle the output value of the multiplexer is kept constant. The value of N is chosen according to the longest propagation delay in the combinational path (in case of Figure 9 it is equal to 2 clock cycles) in this way all signals have time to propagate through the circuit, granting a correct operation of the circuit.

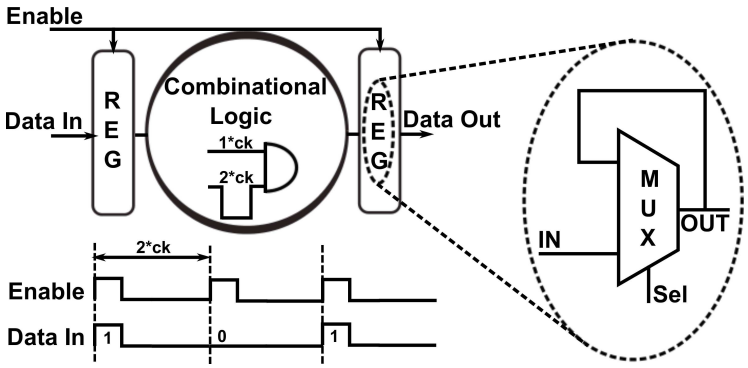

Fig. 9. New asynchronous protocol. No signals coding is used and asynchronous registers are substituted by a multiplexer with the output connected to one of its inputs. The output of the multiplexer is kept constant and only every N clock cycles a new data is sampled. The value of N is chosen according to the maximum propagation delay in the combinational path, in order to grant that all signals had time to propagate correctly through the circuit.

To evaluate the performance of different logic solutions a simple but complete microprocessor was developed in [31] and [36]. The microprocessor is schematically reported in Figure 10.A. The architecture is based on four main components, a program counter to run programs, two memories, one for instructions and one for data, and an arithmetic-logic unit to execute the selected instruction. Asynchronous registers are exploited to implement the communication protocol. The microprocessor architecture is simple but it allows to execute all kind of instructions commonly found in the instruction set of commercially available microprocessors, thus it represents a good test to evaluate the performance of different logic solutions applied to NML logic. The microprocessor is described using a RTL model of NML technology written using VHDL language. The model is reported in Figure 10.B. The basic idea behind the high level modeling of NML technology is to exploit its intrinsic pipelining. Figure 10.B shows an example of NML circuit and its equivalent RTL model. Registers are used to

emulate the signals propagation delay while ideal logic gates without delay are used to model the circuit logic behavior. The result is a circuit that behaves exactly like its NML counterpart, but the advantage is that this circuit can be described and simulated using powerful design tool already available for CMOS technology, like Modelsim [37]. In this way complex NML circuits can be easily described and simulated. More information on the model itself can be found in [28].

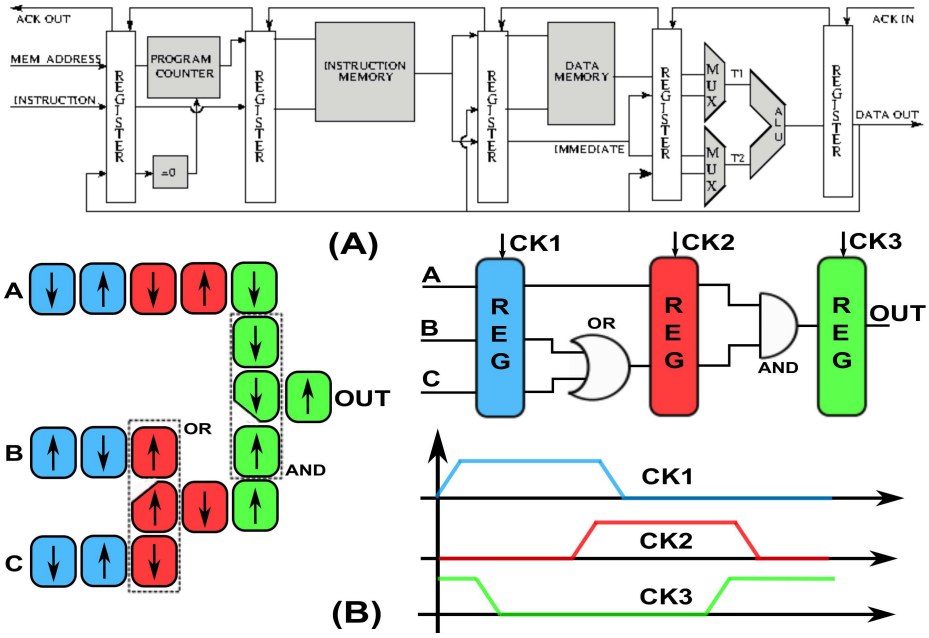

Fig. 10. NML microprocessor. A) The microprocessor architecture is made by a program counter, to execute programs, two memories, one for data and one for instructions and an ALU to execute the instructions. The architecture is simple but it allows the execution of all common microprocessors instructions. B) NML equivalent RTL model used to describe and simulate the microprocessor.

Detailed simulations of the microprocessor are not reported here, and can be found together with a thorough description in [31] and [36]. The most important result obtained from simulations is indeed the time needed to execute one instruction. Using NCL logic one instruction takes 5.35 μs to be executed, while with the improved asynchronous protocol only 0.546 μs are required. As a consequence a speed up of 10 times can be observed. This result demonstrates the validity of the logic solution here proposed, which allows to solve the layout=timing problem without speed penalties. The clock frequency used in the simulation is about 100 MHz, that means a clock period of about 10 ns. The execution of one instruction with the improved logic requires therefore 53 clock

cycles. This slowdown, however, is not due to the use of asynchronous logic but to the presence of a long feedback signals inside the circuit. This aspect is described in detail in Section 3.

2.2 Clock Zones Layout for Automatic Signals Synchronization

In order to synchronize signals in a complex circuit it is however possible to work on the clock zones layout instead that on the logic type. The idea is to exploit the potential of circuit layout shown in Figure 4.A. A circuit detailed example is reported in Figure 11. Clock zones are made by parallel strips and inputs arrive from the left side, all of them starting from the first clock zone. Output signals are generated at the circuit right side. With these constraints perfect signals synchronization is achieved. At any point inside the circuits all signals are perfectly synchronous. This result is obtained without the need of using asynchronous logic keeping the circuit as simple as possible, minimizing the area overhead and maximizing performance. Moreover this solution allows to gain two further advantages: First, signals synchronization is automatically achieved independently of circuit complexity, second, this approach can be successfully applied for hierarchical circuit descriptions.

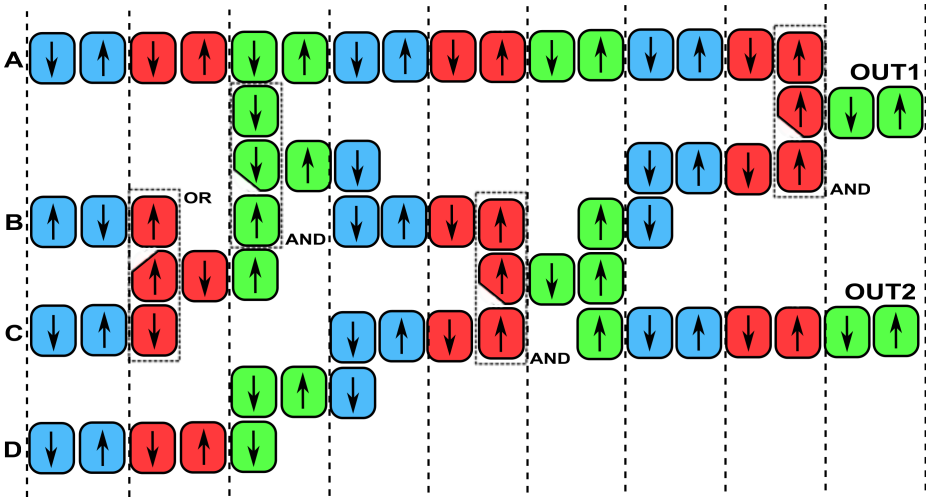

Fig. 11. Regular clock zones layout to obtain automatic signals synchronization. Clock zones are made by parallel strips, input comes from left direction while outputs are generated at the circuit right side.

While the clock zones layout of Figure 11 was built upon the constraints of NML circuits based on magnetic field clock, it allows to achieve additional remarkably results, like automatic signals synchronization and easy circuits de-

scription. As a consequence this same layout can be exploited also for other clock systems. Its regularity also greatly helps the fabrication processes.

3 Loops

If the pipelined nature of these technologies causes troubles to achieve signals synchronization in combinational circuits, the situation worsens when loops are present inside the circuit. Loops are required to build any complex circuit. An example of such a system is reported in Figure 12, where the detailed layout of a 2 bits multiply and accumulate unit (MAC) is shown. In the detail of Figure 12 the simplified circuit schematic is reported. Two incoming signals are multiplied and then the result is added to the result of the previous operation. MAC units are a fundamental block in digital signals processors (DSP).

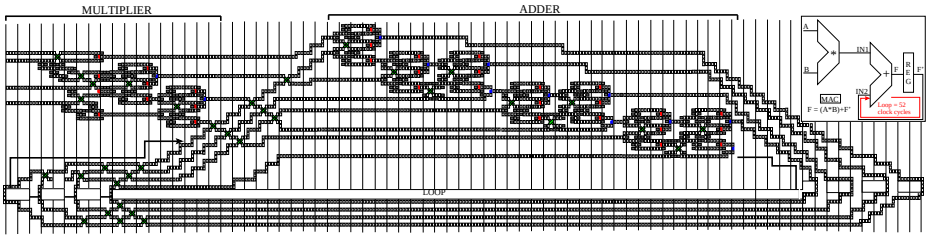

Fig. 12. NML 2 bits Multiply and accumulate unit layout. A) Detailed circuit layout. The loop length is 52 clock cycles. In the upper detail

The first and most important problem due to the presence of loops is the loss of performance. While in pure combinational circuits thanks to the pipelining a new data can be sent to the circuit every clock cycle, if a loop of length N clock cycles is present, a new input can be sent only every N clock cycles, thus reducing the throughput of N times. To demonstrate this fact the MAC unit was described using the VHDL model outlined in Section 2.1. Simulation results are reported in Figure 13. The loop length is 52 clock cycles, so if inputs are fed to the circuit one every 52 clock cycles the MAC output is correct (Figure 13.A). If the delay between subsequent data is lower, for example 50 clock cycles like in Figure 13.B, errors are generated. This behavior can be easily explained by the fact that once the multiplication output is generated it needs 52 clock cycles to travel back and to reach the adder inputs. Therefore to synchronize signals the next output of the multiplier must be generated exactly with a delay of 52 clock cycles, in this way it will reach the adder input together with the previously generated multiplier output. This is a common problem also in CMOS technology, for example in superscalar microprocessors, however in this case the level of pipelining is much deeper, it depends on the circuit layout and it is not a pure designers choice. Solutions to improve performance in circuits with loops are presented in Section 3.1.

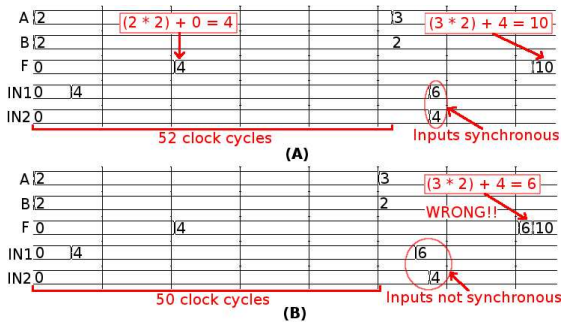

Fig. 13. MAC simulations, with inputs sent one every 52 clock cycles (A) and every 50 clock cycles (B). If the delay between new inputs is lower than the loop length errors are generated.

The second problem encountered with loops in intrinsically pipelined circuits, is related to signals synchronization if more loops are present inside the circuit. Synchronization problems and related solutions are presented in Section 3.2.

3.1 Throughput Maximization

To maximize performance two possible approaches can be used. It is possible to work at algorithmic level, rearranging data avoiding data dependencies, or it is possible to work on the circuit architecture with the aim of reducing the loops length. In the following both solutions are presented.

Interleaving. As seen above, if a loop is present, two consecutive data operands can only be sent to the system after a number of clock cycles equal to the loop length, degrading therefore system performance. The reason of this problem lies in the fact that there is data dependencies between two consecutive data, i.e. one data depends on the previously sent data. It is however possible to work at algorithmic level rearranging data to avoid data dependencies. To do so, some techniques that are commonly adopted in CMOS technology can also be applied here, like dynamic data dependency rearrangements and predictive techniques used in superscalar microprocessors. These solutions may potentially improve performance but they aggravate the system with additional calculations, and require careful effectiveness analysis from application to application. A more simple technique, called "interleaving", can be adopted. It does not require significant modifications to the original algorithm. The basic concept of interleaving is to parallelize relatively independent operations by interleaving data sequence at the inputs. As an example two independent sequences of data A, B and C, D are fed to the circuit. Data from different sequences are completely independent. The aim is to calculate (equations 1 and 2)

$$\sum_{i=0}^{3} A * B \qquad (1) \qquad\qquad \sum_{i=0}^{3} C * D \qquad (2)$$

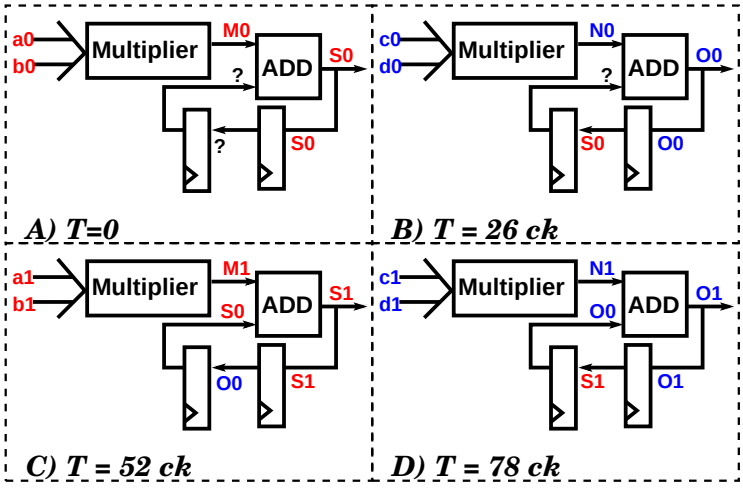

Fig. 14. Example of interleaving applied to a MAC unit. Two input sequences are sent to the circuit, both send one value every 52 clock cycles, with 26 clock cycles between data of different sequences. The left column shows the calculation of sequence A, B, while the right one represents the calculation of sequence C,D. As can be observed, the two sequences are executed in parallel but they do not interfere with each other. A) *a0* and *b0* are sent to the circuit. B) After 26 clock cycles *c0* and *d0* are sent to the circuit. C) At a time correspondent to 52 clock cycles *a1* and *b1* are sent to the circuit and they reach the adder input exactly with *S0*, the result of the previous operation. D) At 78 clock cycles *c1* and *d1* are sent to the circuit.

At the beginning *a0* and *b0*, the first two data of the first sequence are sent to the circuit (Figure 14.A). Just for this example, to better clarify the interleaving principle, the multiplier is considered ideal without delay, so data propagate directly from the general MAC inputs to the adder inputs. After a time equal to half the loop length (26 clock cycles in this case), *c0* and *d0* are sent to the inputs (Figure 14.B). This operation is correct because there is no data dependency between them and *a0*, *b0*. At the 52nd clock cycle, *a1* and *b1* are then sent to the circuit and they arrive at the adder inputs together with *S0* (Figure 14.C), the results of the previous operation. After other 26 clock cycles *c1* and *d1* are sent to the circuit. They arrive at the adder inputs together with *O1* (Figure 14.D). Data from the same sequence are always sent every 52 clock cycles granting perfect signals synchronization, but two sequences are executed in the same time required to execute one sequence alone, effectively doubling the throughput.

Figure 15 shows a MAC simulation with and without interleaving. The sequences, originally executed serially as A, B and then C, D (Figure 15.A), are parallelized and interleaved (Figure 15.B). The original time interval between two consecutive input values (52 clock cycles) is halved. Therefore, the total execution time of these two independent operations is halved, and the through-

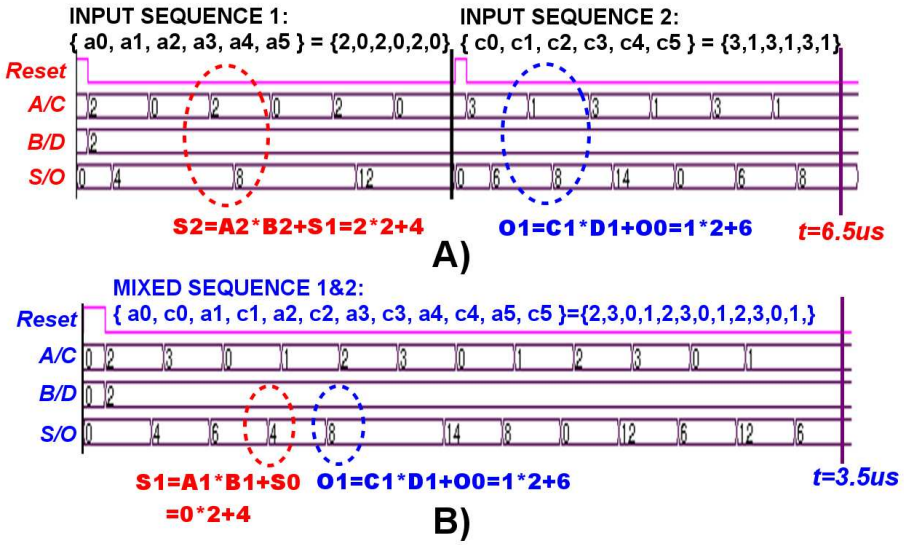

Fig. 15. Simulation comparison between the original circuit behavior (A) and the circuit with data interleaving (B). The only modification done is mixing input sequence 1 and input sequence 2. As can be observed, the calculation results are the same, only are also mixed, but the total simulation time is significantly reduced.

put has been doubled. This principle can be expanded executing 52 operations in parallel, sending therefore effectively one data every clock cycle. The throughput is therefore maximized as in a pure combinational circuit. This clearly demonstrate that interleaving is the perfect technique to be adopted in case of NML (and QCA) circuits. However, in order to fully exploit the potential of this technology, it requires a large number of independent data sequences to process in parallel. Only applications where a large number of data to process is available are therefore best adapted to this technology.

Architecture Redesign for Loops Length Reduction. Besides to algorithm rearrangement techniques like interleaving, it is also possible to modify architectures with the aim of reducing the loops length. Since loops are the major sources of performance degrading in this technology, the shorter the loop is, the better the performance are. With a better analysis of the circuit layout, the MAC circuit can be modified as shown in Figure 16, so the original 52 clock cycle long loop has become only 10 clock cycle long, without changing the system algorithm. The delay is also independent from the MAC bit number, and overall the area is smaller. Without interleaving the delay between one data and the next is therefore of only 10 clock cycles.

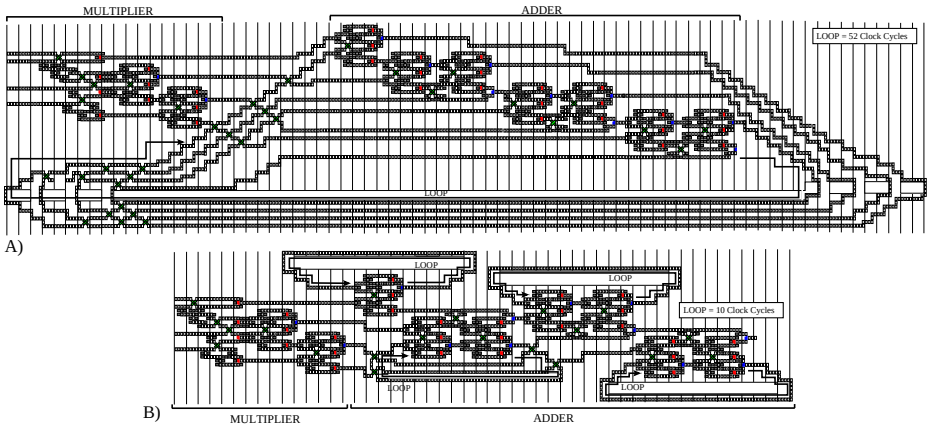

Fig. 16. Redesign of MAC to reduce the loop length. With this new design the loop length is only 10 clock cycles, moreover it is independent from the bit number.

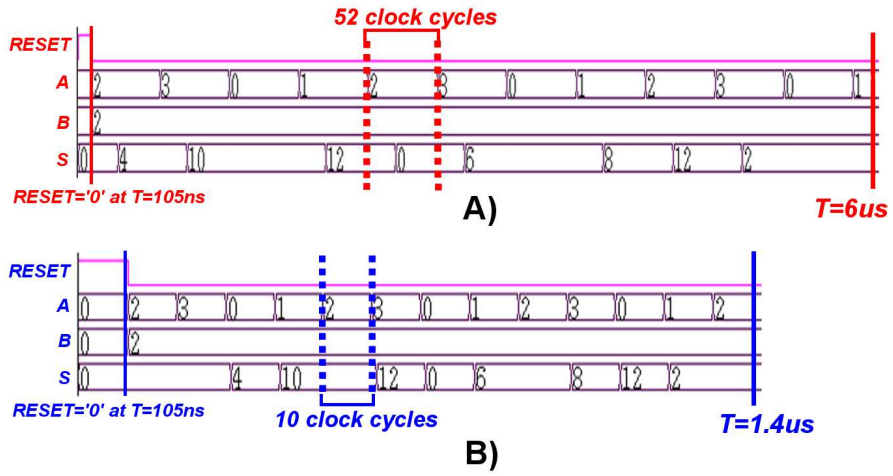

Fig. 17. Simulation comparison between the original architecture (A) and the redesigned architecture (B).

Figure 17 shows a simulation comparison (without interleaving) of the original architecture (Figure 17.A) and the optimized one (Figure 17.B). The execution time, and therefore the throughput, is improved of 5 times. While this technique shows good results, it can be only used as a complementary technique, because it does not allow to completely eliminate the loop. Interleaving is still necessary, however the number of operations required to reach maximum performance is lower.

3.2 Signals Synchronization

While the loss of performance is probably the more relevant problem due to the presence of loops, some important issues arise also for the signals synchronization. Particularly, when more loops are present inside the circuit, they must be carefully designed to achieve perfect signals synchronization. Another issue is instead related to the necessity of adding a specific delay on a particular signal. This is a common requirement in many circuits, where the algorithm mapping implies the necessity of delaying some signals of a specific amount of clock cycles. In case of NML and all the intrinsic pipelined technologies particular rules must be followed to add delays on specific signals. Both problems and the related solutions are described in the following.

Nested Loops. Generally, it is quite common to find multiple loops in a relatively complex system, and they can be independent from each other or they can be nested. In this second case, signal synchronization problems arise. A circuit example with two nested loops is represented in Figure 18. The algorithm is simply $SUB\_OUT = SUM\_OUT - SUB\_IN = A + ADD\_IN - SUB\_IN$. Figure 19 shows instead the circuit simulation. Since $ADD\_IN = SUB\_IN$, the result is simply the value of A.

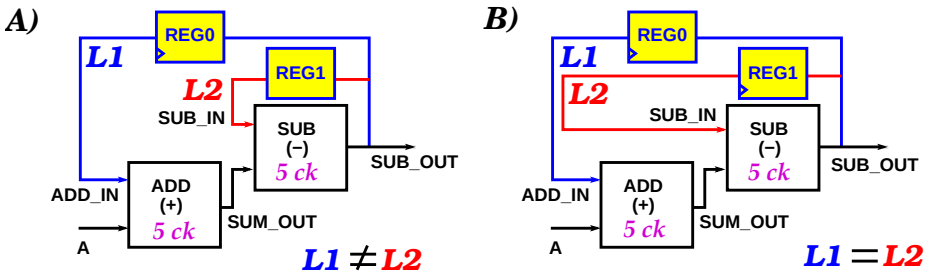

Fig. 18. Circuit representation with two nested loops (A). Originally, the two loops L1 and L2 do not have the same length (B). For signal synconizzation, loop L2 is extended to the same length as loop L1.

If the loops length is different (Figure 18.A) signals are not perfectly synchronized and the output result is wrong as can be observed from Figure 19.A. If instead the loops length is equal (Figure 18.B) the circuit output is correct, as the simulation of Figure 19.B clearly demonstrate.

The consequence is simple: If there are more loops inside the circuit and these loops are nested inside each other, they must have the same length to obtain perfect signals synchronization.

Additional Delay Loops. The presence of loops introduces another problem related to signals synchronization. Sometimes mapping an algorithm to an elec-

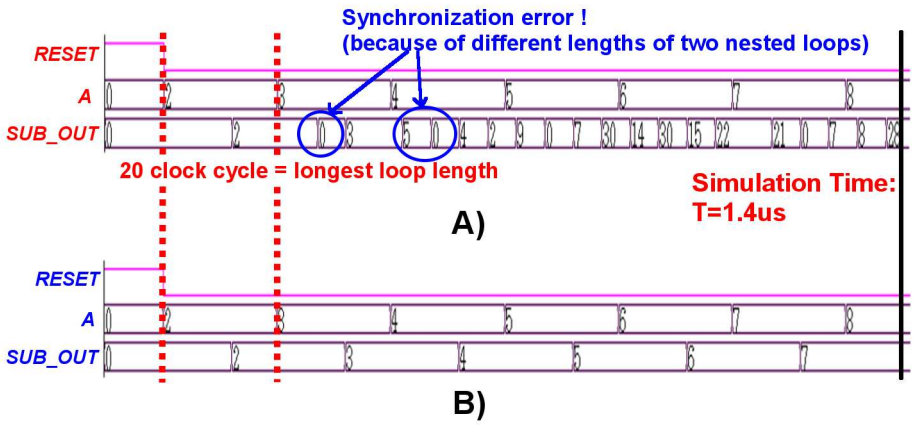

Fig. 19. Simulation comparison with nested loops that have different (A) and equal length (B). As can be observed, synchronization error will appear if the two nested loop lengths are not equal.

tronic circuits requires the insertion of a delay on specific signals. An example is shown in Figure 20.A. An adder calculates the sum between a signal A, the same signal delayed of an additional clock cycle $A(t\text{-}1)$ and the output of the previous operation $SUM(t\text{-}1)$. Trying to map the same circuit on NML (or QCA) technology one important problem arises. In the CMOS implementation the loop has a delay of exactly 1 clock cycle, a new input A is sent every clock cycle, so all signals arrive at the adder input with perfect synchronization, respecting the algorithm.

However in NML (and QCA) loops normally have a delay of N clock cycles, so the result of the previous operation arrives at the adder input after N clock cycles. As explained in Section 3 a new data must be sent every N clock cycles. This is true also in case of interleaving, because N operations are interleaved, but the delay between one data and the subsequent data of the same operation is always N clock cycles. Therefore to map the same algorithm it is not sufficient to delay the input A of 1 clock cycle but it must be delayed of N clock cycles (Figure 20.B). Considering an example, an input $(a0)$ is sent to the first adder input. The output is calculated and after N clock cycles it reaches the adder inputs. After N clock cycles a new data $(a1)$ is sent to first adder input, but at the same time the previous input $(a0)$ reaches the second adder input because it is delayed of exactly N clock cycles. As a general rule, for each additional delay of one clock cycle in the original circuit, an additional delay of N clock cycles must be added to the correspondent wire in the NML implementation.

Figure 21 shows the circuit simulation. In Figure 21.A the CMOS implementation is depicted, while in Figure 21.B the NML simulation, with the use of the additional synchronization delay, is shown. The time scale is different because in the CMOS implementation, one data is sent every clock cycle while in the NML

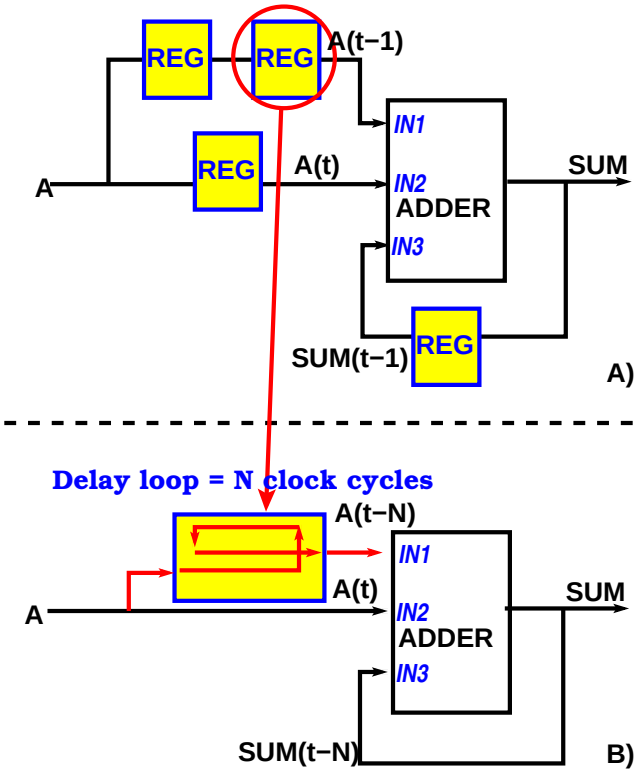

Fig. 20. Circuit example. The algorithm is $SUM(t) = A(t) + A(t-1) + SUM(t-1)$. A) CMOS implementation. B) NML implementation. An additional delay, under the form of a wire loop, is used to map the additional delay on signal A.

implementation one data is sent every N clock cycles, but the behavior is the same. If the signal A is not correctly delayed the result is wrong, as can be seen from Figure 21.C.

4 Interconnections

The final problem encountered in the design of complex circuits with Field-Coupled technologies, is the strong impact of interconnections both on the propagation delay and on the circuit area. Figure 22 shows the detailed layout of a NML 8 bits comparator, that is part of an LDPC decoder for wireless signals [38][39][40]. The circuit schematic is reported in the detail of Figure 22, while the decoder description is not reported here because it is not relevant for this discussion, however details on the architecture itself can be found in [38]. What it is important to understand is that most of the area (up to 99%) is occupied by interconnection area. This layout is based on the technological constraints

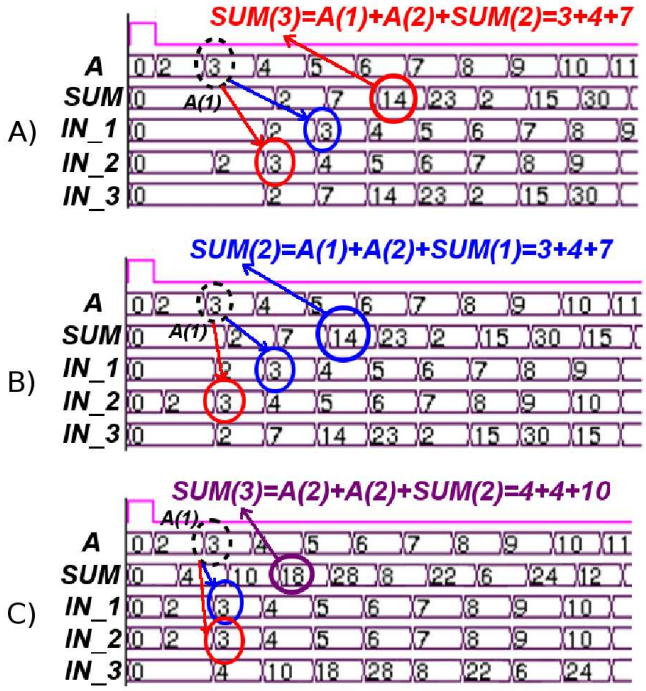

Fig. 21. Simulation comparison between the orignal circuit (A), the correct NML circuit with additional loop (B), and the NML circuit without additional loop (C), which gave the wrong result.

described in Section 1.1, so in other types of NML or QCA the impact of interconnections can be different, but probably interconnections will ever represent an important part of the circuit area. It is worth to underline again that in Field-Coupled devices more area means more delay and more power consumption.

The reason of this problem comes from the nature of the technology itself, which favors local interactions over neighbor elements, penalizing long interconnection wires. A possible solutions is to use particular architectures, called systolic arrays, that can greatly reduce the interconnections overhead. Systolic arrays are briefly described in Section 4.1 alongside with some techniques to optimize the design and to improve performance.

4.1 Systolic Arrays

A Systolic Array (SA) is a network of Processing Elements (PEs), also called "Cells", that are locally interconnected and can work in parallel. SAs were first introduced by Kung and Leiserson in 1978, who stated: "a systolic system is a network of processors which rhythmically compute and pass data through the system" [41]. Each PE receives data from neighboring cells or from outside and

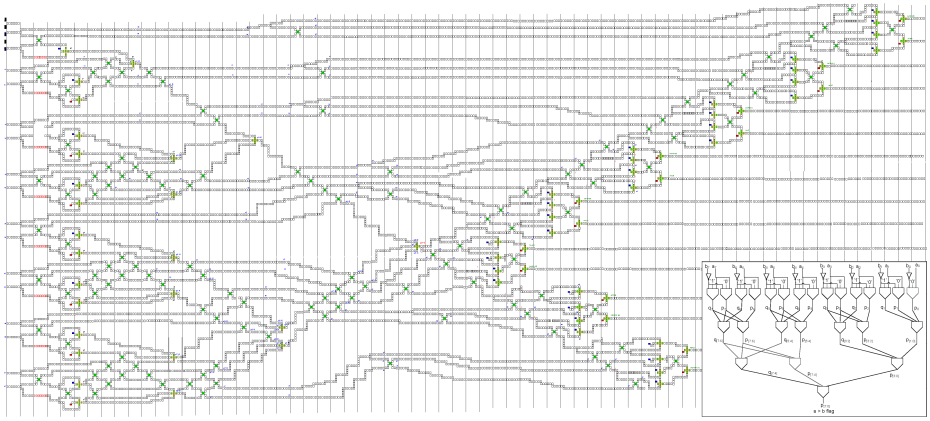

Fig. 22. Example of interconnection impact in complex NML circuits. The circuit, reported schematically in the detail, is a 8 bit comparator used inside a wireless decoder.

outputs result to the outside or to near PEs. Two are the main concepts at the basis of SAs: parallel computation (i.e. all PE work in the same way on different data) and local transmission of data (i.e. there are not global signals). In Figure 23 three examples of SAs arrangements are shown: a bi-dimensional matrix-like shape SA (a), a linear SA (b), and an SA with signals flowing in three different directions (c). The shape is determined by a three step design process: starting from the algorithm description, the Dependence Graph (DG) is derived; this is shrunk along one axis to obtain the Signal Flow Graph (SFG) which represents the real shape of the SA. Finally, PE internal structure is derived.

In last decades, increasing operating frequency has been enough to sustain the request of higher computational efficiency of digital circuits, reducing the need of parallel architectures; for this reason SAs have not represented an attractive solution. However, approaching the boundary limit for CMOS scaling [42], and thus for frequency increase, parallel solutions are being exploited and SAs are back in the limelight. They have been designed for image processing [43] [44] [45], signal processing [46] [47] [48] and video algorithms (such as those for MPEG compression). Recently, automatic tools concerned to translate algorithms to SAs for FPGAs have been explored [49]. Also, reconfigurable arrays, that are not application-specific, have been introduced in [50]. SAs are a good architectural solution for new technologies (beyond-CMOS), where benefits in terms of speed and required area can be achieved only avoiding global interconnections, and SAs verify intrinsically this requirement. SA paradigm allows to design local small circuits (the PEs) and replicate them to organize the whole array structure. In particular, in QCA several circuits have been designed and simulated; as an example, in [51] and [52] SAs for matrix multiplication and Galois field multiplication have been proposed. Also, NML implementations for convolution filters have been proposed in [53].

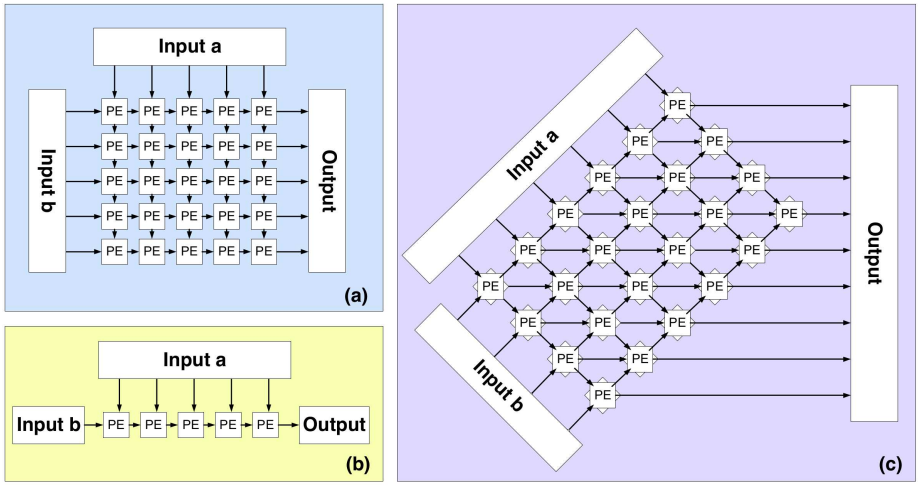

Fig. 23. Three examples of Systolic Arrays. a) Matrix-like SA. b) Linear SA. c) SA with signals flowing in three different directions.

While SAs help to solve the interconnection problem of this technology, they still suffer from a loss of performance in presence of loops. Interleaving must therefore be used in conjunction with SAs to maximize performance [54]. As far as optimization with pipeline interleaving is concerned, SAs can be distinguished between those that have PEs WithOut Internal Loops (WOIL), and those with PEs With Internal Loops. The latter can then be further divided in SAs that Store result in cells (WIL-S) and SAs that evaluate the final results passing partial results through lines (WIL-PT). WOIL SAs are composed of PEs without loops; for this reason, pipelining is enough to achieve the required performance increase. Pipeline interleaving for this reason is applied to WIL SAs only.

The generic PE of a WIL SA is shown in Figure 24. It is composed of 4 parts: an entry section, made of blocks numbered from 1 to i; the forward part of the loop, made of blocks from $i+1$ to j, the feedback part of the loop, made of blocks from $j+1$ to $k-1$; the output block, called k. Each of these blocks has a delay d_n, $n = 1, 2, \ldots, k$ and cannot be internally pipelined. Call Z_e the total delay of the entry block, Z_{fo} the delay of the forward side of the loop, Z_{fb} the delay of the feedback in the loop and Z_o the output delay:

$$Z_e = \sum_{n=1}^{i} d_n \qquad (3) \qquad\qquad Z_{fo} = \sum_{n=i+1}^{j} d_n \qquad (4)$$

$$Z_{fb} = \sum_{n=j+1}^{k-1} d_n \qquad (5) \qquad\qquad Z_o = d_k \qquad (6)$$

Input data coming from outside enter in the first block, while data coming from the neighbor processing element can enter at any stage of the cell. Inputs must be provided every $Z_{loop} = Z_{fo} + Z_{fb}$ cycles, that is the total time of the

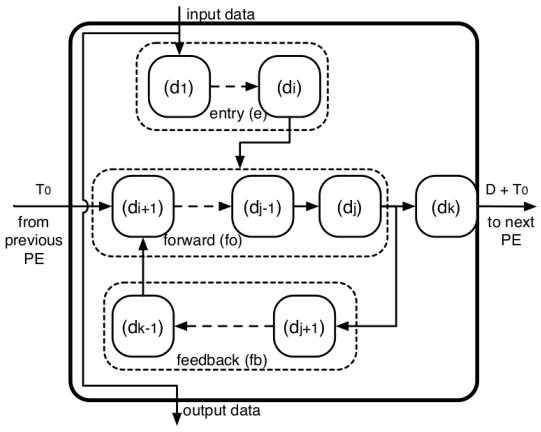

Fig. 24. WIL Systolic Array generic processing element structure.

feedback loop, in order to match them with data coming from the feedback. To apply interleaving the intrinsic pipelined nature of the structure can be exploited. Thus we can improve performance and usage of the cell giving inputs every $J = max\{d_n\}$. Every J cycles a new operation can be started, and in this way M different operations can be interleaved, $M = Z_{loop}/J$ (integer division). After Z_{loop} cycles, the second set of inputs is fed. When Z_{loop} is not a perfect multiple of J, the remainder of the division, called R, must be taken into account: after M successive inputs have been fed, the following one must provided with a delay of $J + R$, so to have synchronization with the value coming from the loop. R represents a number of "stalls" that must be inserted between the one set of M inputs and the following set.

Applying pipeline interleaving it is possible to evaluate M different operations in parallel, having an increase in performance of M. If the number of stalls is high, it would be favorable to increase the delay of the feedback loop in order to achieve a deeper level of interleave. For example, consider $Z_{fo} = 20, Z_{fb} = 6, J = 9$; in this case $M = 2$ and $R = 8$. If it is possible to increase of 1 cycle the delay of the feedback, then 3 operations can be interleaved and no stalls will be present. As far as the global array is concerned, in WIL-S SAs processing elements work independently each from the others, and the only connections are the wires to pass inputs from one cell to another. Being S the delay in clock cycles of connection wires, inputs must be given with the same rule for all cells (number of interleaved operations and stall cycles), but starting at cycle $S_n = m_n \times S$; m_n is the Manhattan distance between cell n and the top-left one (if we consider data moving from top to bottom and left to right); this delay must then be reported to the actual inputs of the array at boundary cells. In WIL-PT SAs instead, results of a cell are re-used in the same cell but also passed to the neighboring cell; in this case the order of inputs is given by the mismatch between the feedback loop delay and the one to transmit data to next cell.

In order to evaluate the benefits of pipeline interleaving the Cell Updates Per Second (CUPS) parameter can be computed. Consider a square SA as the one in Figure 23.(a): the top-left PE is the one that starts operating first, while the bottom-right one is the last. Given a finite number of inputs, the bottom-right PE is also the one that finishes later computation; hence, total time will be given by the time at which this PE will finish to compute the last result. Total time, Z_{end}, is given by the time for last inputs to reach last PE, plus the time to execute the operation inside the PE itself, called Z_{cell}. S was previously defined as the number of cycles needed for an input to pass through a cell, that is the number of registers of the shift chain. It is possible to assume that this value is equal to transmit one value from top input to bottom output of the PE, or from left input to right output. First inputs will be available at last cell after $2(M-1)S$, where $2(M-1)$ is the Manhattan distance between first cell and last one in an array of $M \times M$ PEs. Call $Z_{end}^{(m)}$ the time at which computation of m-th result is available. Then: $Z_{end}^{(1)} = 2(M-1)S + Z_{cell}$. J is the delay between one input and the following one; if l successive inputs are fed into the array, then we obtain equation 7:

$$Z_{end} = Z_{end}^{(l)} = 2(M-1)S + (l-1)J + Z_{cell} \tag{7}$$

Formula (7) expresses the total time needed for executing operations on a $M \times M$ SA that receives l successive data from each input path. During this period of time each cell will execute l operation (one every time a new input is received). The total cell updates are lM^2, and CUPS can be evaluated as:

$$CUPS = f_{clk} \, \frac{lM^2}{2(M-1)S + (l-1)J + Z_{cell}} \tag{8}$$

This formula must be adapted in two cases: when the array is used without interleaving operations, and when interleaving is exploited to achieve an improvement in performance. In case of no-interleaving formula (8) can be adapted considering $l = M$. In case of n-interleaving instead, each cell will update n times more than the previous case, hence $l = nM$; this increase reflects also at the denominator of formula (8) as an increase in total time. Moreover, Z_{cell} and J must be re-evaluated considering interleaving. SAs performance are often evaluated computing peak CUPS $\max(CUPS) = f_{clk} \times \#PEs$ where $\#PEs$ is the number of PEs in the array. Equation 8 is more precise and takes into account the effects of interconnections and delays inside PEs. This equation is upper bounded by $\max(CUPS)$.

In order to demonstrate the effectiveness of the proposed mechanism of pipeline interleaving to increase performance, we report here the evaluation of CUPS according to equation 8 for different levels of interleave in a practical case. Hereinafter values are given in number of clock cycles. Imagine to have a WIL-S SA with $Z_e = 8$, $Z_{fo} = 7$, $Z_{fb} = 5$, with Z_e and Z_{fo} that cannot be further pipelined, while the feedback can be represented as a shift register. Without

interleaving $Z_{cell} = Z_e + Z_{fo} = 15$ and $J = Z_{loop} = 12$. Moreover, let consider $S = 10$ whatever is the level of interleave, since this value is not influenced by interleaving. In this case equation 8 reduces to: $CUPS = M^3/(32M - 17)$. It is possible to achieve interleave 2 considering that $J = \max\{Z_e, Z_{fo}\} = 8$ so it is required to design the feedback loop to have a delay of 12 clock cycles. In this case equation 8 must be adapted considering $J = 8$ and $P = 2M$. Figure 25 shows the trend of CUPS in function of M for different levels of interleave. It can be immediately noticed that applying interleaving results in significant benefits in terms of CUPS. Moreover, it is evident that benefits of applying interleaving saturate with deeper levels of interleave.

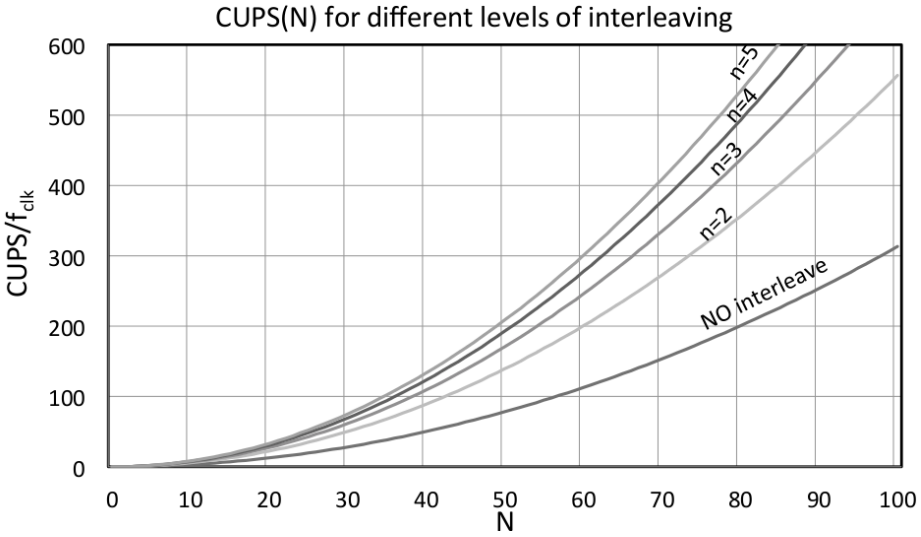

Fig. 25. The effect of *pipeline interleaving* on CUPS: given an array of $M \times M$ cells, increasing the level of interleaving CUPS increase as well.

5 Analysis of Applications Suited for NML Technology

Since the nature of Field-Coupled devices is quite different from traditional CMOS circuits, it is important to identify in which kind of applications the true potential of this technology can be fully exploited. The architectural overview presented in this chapter represents therefore the ideal starting point for such analysis. In the following the most important features of Field-Coupled devices are summarized.

1. **Parallelism**. To maximize performance parallel computation is required. As seen in Section 3 the interleaving of a big number of operations is required to

maximize throughput. As a consequence all applications that are intrinsically parallel, or where it is necessary to process a huge amount of data are the ideal target of these devices.
2. **Locality**. A huge delay penalty is acquired by signals traveling over long interconnection wires. The consequence is simple: Only local interconnections should be used (see Section 4). Therefore all algorithms that require only a local exchange of data are favored in this technology.

Along with these two features that are common to all Field-Coupled devices, NML circuits, due to their magnetic nature, have some more specific advantage.

3. **Logic-in-memory**. Magnets are intrinsically memory devices. They are used here for logic computation but they maintain their status also without external power supply, providing therefore a natural memory ability. This possibility can be exploited to build logic-in-memory devices, where memory and logic functions are combined together.
4. **No stand-by power consumption**. Since magnets maintain their state also without power external supply, when circuits are in stand-by mode there is no power consumption at all. This fact can be exploited to reduce power consumption, activating the circuit only when it is necessary and shutting it down in the meantime.
5. **Radiation Hard**. NML circuits are extremely resistant to radiations. Circuits still work correctly in after taken a considerable amount of high energy radiations.
6. **Low dynamic power consumption**. The dynamic power consumption can be very low, depending on the clock system used.

Every application that can benefit from one of more of these characteristics can therefore be an ideal target for this technology. Clearly the more features are exploited, the more benefit it is possible to gain from this technology. In the following a short (but necessary not complete list) of possible field of applications for NML technology is presented.

Aerospace and Military. The near-immunity to radiations is a property that gives to NML technology and undeniable advantage over CMOS circuits. Aerospace and military applications are therefore the ideal target for NML technology. In these fields there is an always increasing demand for radiation hard, heat resistant processing units that must work in hostile environments. With the CMOS technology is very difficult to match all aforementioned characteristic, because circuits can be damaged by radiations. NML devices are the perfect solution to match these fields necessities. Moreover, the NML logic-in-memory ability paves the way for "on the flight" circuits reconfigurability that could enhances space and military applications with re-programmable hardware and at run-time adaptive functionality. This feature enables the possibility to change the function or to repair part of the system if something went wrong.

Embedded Sensors in Remote Zones. One of the main characteristics offered by NML technology, the zero stand-by power consumption, can be exploited to built intelligent sensor stations in remote zones. In this kind of systems it is quite common to have circuits that need to operate for a limited period of time, then they have to spent the rest of the time in stand-by mode. These kind of systems are actually used in many applications; one of these is the environmental monitoring in remote zones such mountains peaks or buoys placed in the ocean, which detect the rate of seas pollution. The use of NML technology can give a great advantage compared to the devices actually used, mainly thanks to the zero stand-by power consumption, but also thanks to its resistance to extreme environmental conditions.

Automotive. The resistance to extreme environment can be useful also in automotive applications. It is well know that electronic components that must be placed near the mechanical parts of a car, like the engine, are exposed to a great deal of stress. This stress comes under many forms, heat, vibrations, voltage spikes, straining the resistance of CMOS circuits to their limits. Clearly the use of NML logic can represent a substantial improvement to the reliability of electronic circuits applied to the automotive field.

Integration with other magnetic systems. While here magnets are used to build logic circuits, magnetic devices are commonly employed in many field as sensors and actuators, not to mention as memory devices. Hard-disks are the leading devices for mass data storage and in recent years Magnetic RAM were also developed and are gaining an increasing interest for their prominent features [55][56], like radiations resistance, high speed, non-volatile nature, heat resistance and nearly-unlimited endurance. Since all these devices share the same magnetic nature it is possible to think to merge logic devices and MRAM together. This is particularly interesting because memory performance is fast becoming the key bottleneck that limits system performance, and critical applications are becoming more data-dependent, and less compute-dependent. Instant-on will be a requirement for a lot of applications. Merging NML circuits and MRAM can be a solution to this problem, a further way to exploit the logic-in-memory principle.

Nonetheless it is also good to consider that many embedded applications implicate the acquisition of signals from sensors, the processing of acquired signal, the storage of the elaborated data, the command of actuators and the communication with other system. A lot of sensors, actuators and memory devices are based on magnetism. This consideration can be exploited to obtain a deeper level of embedding. If for example magnetic sensors are constructed on the same chip as NML devices using the same fabrication processes a new scenario will be opened. Indeed the result of these devices combination originates a system on a chip with potential advantages in therms of both performance and power.

Dynamic circuit reconfigurability. The ability to mix logic and memory in the same device can be used to enhance the concept of reconfigurable logic

circuit. For example it is possible to think to computational algorithms that dynamically manage the logic and the memory according to the necessaries resources. As an example it is possible to consider an algorithm that works in two phases: the first one needs a big amount of computational power and a little memory; instead the second phase requires less computations but an increased quantity of memory. A dynamic reconfigurability coupled with the fact that both logic and memory are available in the same device can revolutionize the way to think an design the algorithms and their implementations on the hardware architectures. Another possibility is to build logic devices that can be reconfigured run-time, leading to a totally new class of "smart" circuit, obtaining considerable advantages in terms of circuit area, power consumption and speed.

Image processing. The possible applications presented since now are particularly suited for NML logic. However every application wherever a huge amount of data to process is available is the ideal target for all QCA technologies, not only for the magnetic version. An example is the image processing field where digital images, made by a big matrix of pixel, must be elaborated. The amount of data to process is very high, secondly the algorithms involved in the elaboration of an image are often applied to a pixel and its neighbors. Image processing is therefore an application particularly suited for QCA technology because it can exploit both the principle of *Parallelism* and the principle of *Locality*. Among all QCA implementations, NML logic is favored due to its low power consumption. It is possible to think, for example, to a dedicated co-processor coupled with the sensor of a mobile camera. In this case NML logic presents therefore a huge advantage over CMOS technology.

Informatics for biotechnology. Similarly to image processing also the biotechnology field can gain a huge advantage from the use of QCA technology. The massive parallelism offered by QCA could be successfully exploited to develop hardware accelerators or specialized co-processors to execute specific heavy-computation tasks. One of these task that recently has obtained great interest is the modeling of molecular dynamics in the cellular membrane. This kind of simulation is useful for the investigation of cellular effects of molecular alterations due to pathological conditions or to extend the characterization of other sub-cellular structures [57]. Recently a highly parallel implementation of molecular dynamics simulation was developed for general purpose GPUs in order to exploit the massive parallelism offered by such devices [58]. As shown in this GPU implementation, the massive parallelism is one of the major feature that can improve the computation in the biotechnology field.

Another biotechnology task that recently has obtained an increasing interest from many interdisciplinary research groups is the study of biological sequences and the relative development of tools. Biologists study the similarities between proteins to reconstruct phylogenetic trees and to assess the presence of mutations that lead to genetic diseases or tumors. Since proteins consist of long sequences of amino acids, the fastest way to performs a first analysis is to align the stud-

ied protein with the protein coming from huge databases searching for regions of similarity. After that, the short list of proteins that share a sufficient level of similarity are deep investigated. There are many alignment algorithms that are chosen according to the type of analysis that the biologist wants to perform. One of the most used is the method developed by Smith and Waterman that performs, in dynamic programming, the exaustive local alignment between two sequences [59]. We have designed a systolic architecture that accelerate the Smith-Waterman algorithm execution [60][61] mapping it to NML technology [62][63] demonstrating how much gain can be obtained from the use of NML technology.

6 Conclusions

In this chapter we have presented a thorough analysis of the main problems that arise at architectural level in Field-Coupled devices. The analysis is mainly based on NML logic but most of the results here presented are valid for all QCA implementations. For every problem presented a solution is proposed. The solutions described in this chapter are designed around both logic and technological constraints. Finally, a brief overview of the field of applications that allows to fully exploit the true potential of this technology is presented. Overall this chapter gives to researchers and designers most of the guidelines necessary to design complex circuits with this technology and the directions that they should follow in the future development of this technology.

Now it is important to continue the architectural analysis focusing on the system level integration and all related problems. Important problems must be investigated, like the interfaces with the outside world and the generation of clock signals. Particularly, in case of NML logic it is important to investigate how to reduce the necessity of CMOS transistors in the support circuits, like the input/output interfaces and clock waveforms generators. The necessity to use CMOS transistors in the external circuits represents a weakness because it reduces one of the most important advantages of this technology, the immunity to radiations.

We will therefore continue to investigate NML (and QCA) architectures following these guidelines, always keeping an eye at the technological constraints in order to get the most realistic results.

References

1. Lent, C., Tougaw, P., Porod, W., Bernstein, G.: Quantum cellular automata. Nanotechnology **4** (1993) 49–57
2. Csurgay, A., Porod, W., Lent, C.: Signal processing with near-neighborcoupled time-varying quantum-dot arrays. IEEE Transaction On Circuits and Systems **47**(8) (2000) 1212–1223
3. Lu, U., Lent, C.: Theoretical Study of Molecular Quantum-Dot Cellular Automata. Journal of Computational Electronics - Springer **4** (2005) 115–118

4. Imre, A., Ji, L., Csaba, G., A.O. Orlov, Bernstein, G., Porod, W.: Magnetic Logic Devices Based on Field-Coupled Nanomagnets. 2005 International Semiconductor Device Research Symposium (December 2005) 25
5. Pulimeno, A., Graziano, M., Abrardi, C., Demarchi, D., Piccinini, G.: A write-in system based on electric fields for Molecular QCA. In: 2011 IEEE International NanoElectronics Conference (INEC), Tao-Yuan, Taiwan, IEEE (2011) 1–2
6. Pulimeno, A., Graziano, M., Piccinini, G.: Molecule Interaction for QCA Computation. IEEE International Conference on Nanotechnology (2012)
7. Pulimeno, A., Graziano, M., Demarchi, D., Piccinini, G.: Towards a molecular QCA wire: simulation of write-in and read-out systems. Solid State Electronics **77** (2012) 101–107
8. Pulimeno, A., Graziano, M., Saginario, A., Cauda, V., Demarchi, D., Piccinini, G.: Bis-ferrocene molecular QCA wire: ab-initio simulations of fabrication driven fault tolerance. IEEE Transaction on Nanotechnology (2013)
9. Lent, C., Isaksen, B.: Clocked Molecular Quantum-Dot Cellular Automata. IEEE Transactions on Electron Devices **50**(9) (September 2003) 1890–1896
10. Porod, W.: Magnetic Logic Devices Based on Field-Coupled Nanomagnets. Nano & Giga (2007)
11. Pulimeno, A., Graziano, M., Piccinini, G.: UDSM Trends Comparison: From Technology Roadmap to UltraSparc Niagara2. IEEE Transactions on VLSI systems **20**(7) (July 2012)
12. Niemier, M., Hu, X., Alam, M., Bernstein, G., W. Porod, M.P., DeAngelis, J.: Clocking Structures and Power Analysis for nanomagnet-Based Logic Devices. In: International Symposium on Low Power Electronics and Design, Portland-Oregon, USA, IEEE (2007) 26–31
13. Niemier, M., al.: Nanomagnet logic: progress toward system-level integration. J. Phys.: Condens. Matter **23** (November 2011) 34
14. Csaba, G., Porod, W.: Behavior of Nanomagnet Logic in the Presence of Thermal Noise. In: International Workshop on Computational Electronics, Pisa, Italy, IEEE (2010) 1–4
15. Graziano, M., Chiolerio, A., Zamboni, M.: A Technology Aware Magnetic QCA NCL-HDL Architecture, Genova, Italy, IEEE (2009) 763–766
16. Graziano, M., Vacca, M., Zamboni, M.: Magnetic QCA Design: Modeling, Simulation and Circuits. Cellular Automata - Innovative Modelling for Science and Engineering, InTech, http://www.intechopen.com/articles/show/title/magnetic-qca-design-modeling-simulation-and-circuits (2011)
17. Graziano, M., Vacca, M., Chiolerio, A., Zamboni, M.: A NCL-HDL Snake-Clock Based Magnetic QCA Architecture. IEEE Transaction on Nanotechnology **10**(5) (September 2011) 1141–1149
18. Alam, M., Siddiq, M., Bernstein, G., Niemier, M., Porod, W., Hu, X.: On-chip Clocking for Nanomagnet Logic Devices. IEEE Transaction on Nanotechnology (2009)
19. Das, J., Alam, S., Bhanja, S.: Low Power Magnetic Quantum Cellular Automata Realization Using Magnetic Multi-Layer Structures. J. on Emerging and Selected Topics in Circuits and Systems **1**(3) (September 267-276)
20. Das, J., Alam, S., Bhanja, S.: Ultra-Low Power Hybrid CMOS-Magnetic Logic Architecture . Trans. on Computer And Systems (2011)
21. Karunaratne, D., Bhanja, S.: Study of single layer and multilayer nano-magnetic logic architectures . Journal Of Applied Physics (111) (2012)

22. Vacca, M., Crescenzo, L., Graziano, M., Zamboni, M., Chiolerio, A., Lamberti, A., Enrico, E., Celegato, F., Tiberto, P., Boarino, L.: Electric clock for NanoMagnet Logic Circuits . Field Couple Computing Workshop (FCN) (2013)
23. Fashami, M.S., Atulasimha, J., Bandyopadhyay, S.: Magnetization Dynamics, Throughput and Energy Dissipation in a Universal Multiferroic Nanomagnetic Logic Gate with Fan-in and Fan-out. Nanotechnology **23**(10) (February 2012)
24. Rizos, N., Omar, M., Lugli, P., Csaba, G., Becherer, M., Schmitt-Landsiedel, D.: Clocking Schemes for Field Coupled Devices from Magnetic Multilayers. In: International Workshop on Computational Electronics, Beijin, China, IEEE (2009) 1–4
25. Becherer, M., Kiermaier, J., Csaba, G., Rezgani, J., Yilmaz, C., Osswald, P., Lugli, P., Schmitt-Landsiedel, D.: Characterizing magnetic field-coupled computing devices by the Extraordinary Hall-effect. In: Proceedings European Solid State Device Research Conference, Athens, Greece, IEEE (2009) 105–108
26. Ju, X., Niemier, M., Becherer, M., W. Porod, M.P., Lugli, P., Csaba, G.: Systolic Pattern Matching Hardware With Out-of-Plane Nanomagnet Logic Devices. IEEE T. on Nanotechnology **12**(3) (May 2013)
27. Ercan, I., Anderson, N.: Heat Dissipation Bounds for Nanocomputing: Theory and Application to QCA. (2011)
28. Vacca, M., Graziano, M., Zamboni, M.: Nanomagnetic Logic Microprocessor: Hierarchical Power Model. IEEE Transactions on VLSI Systems (August 2012)
29. Vacca, M., Graziano, M., Zamboni, M.: Majority Voter Full Characterization for Nanomagnet Logic Circuits. IEEE T. on Nanotechnology **11**(5) (September 2012) 940–947
30. Niemier, M., Varga, E., Bernstein, G., Porod, W., Alam, M., Dingler, A., Orlov, A., Hu, X.: Shape Engineering for Controlled Switching With Nanomagnet Logic. IEEE Transactions on Nanotechnology **11**(2) (March 2012) 220–230
31. Graziano, M., Vacca, M., Blua, D., Zamboni, M.: Asynchrony in Quantum-Dot Cellular Automata Nanocomputation: Elixir or Poison? IEEE Design & Test of Computers (September 2011) 72–83
32. Niemier, M., Kogge, P.: Problems in designing with QCAs: Layout = Timing. Int. J. Circ. Theor. Appl (2001)
33. Vacca, M., Vighetti, D., Mascarino, M., Amaru, L., Graziano, M., Zamboni, M.: Magnetic QCA Majority Voter Feasibility Analysis . 2011 7th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME) (2011) 229–232
34. Fant, K., Brandt., S.: NULL Convention Logic^{TM}, A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis. In: International Conference on Application Specific Systems, Chicago-Illinois, USA, IEEE (1996) 261–273
35. Choi, M., Patitz, Z., Jin, B., Tao, F., Park, N.: Designing layout-timing independent quantum-dot cellular automata (QCA) circuits by global asynchrony. Journal of System Architecture, Elsevier **53** (2007) 551–567
36. Vacca, M., Graziano, M., Zamboni, M.: Asynchronous Solutions for Nano-Magnetic Logic Circuits. ACM J. on Emerging Tech. in Comp. Systems **7**(4) (December 2011)
37. : Mentor Graphics http://www.modelsim.com.
38. Awais, M., Vacca, M., Graziano, M., Masera, G.: Quantum dot Cellular Automata Check Node Implementation for LDPC Decoders . IEEE Transaction on Nanotechnology (2013) 368–377
39. Awais, M., Vacca, M., Graziano, M., Masera, G.: FFT Implementation using QCA . 2012 19th IEEE International Conference on Electronics, Circuits and Systems (ICECS) (2012) 741–744

40. Martina, M., Masera, G.: Turbo NOC: A framework for the design of network-on-chip-based turbo decoder architectures. IEEE Trans. on Circuits and Systems I **57**(10) (2010) 2776–2789
41. Kung, H., Leiserson, C., Science, C.M.U.D.o.C.: Systolic Arrays for (VLSI). CMU-CS. Carnegie-Mellon University, Department of Computer Science (1978)
42. Haron, N., Hamdioui, S.: Why is CMOS scaling coming to an END? In: Design and Test Workshop, 2008. IDT 2008. 3rd International. (dec. 2008) 98–103
43. Pan, S.B., Park, R.H.: Unified systolic arrays for computation of the DCT/DST/DHT. Circuits and Systems for Video Technology, IEEE Transactions on **7**(2) (apr 1997) 413–419
44. Panchanathan, S., Goldberg, M.: A systolic array architecture for image coding using adaptive vector quantization. Circuits and Systems for Video Technology, IEEE Transactions on **1**(2) (jun 1991) 222–229
45. Iyengar, G., Panchanathan, S.: Systolic array architecture for Gabor decomposition. Circuits and Systems for Video Technology, IEEE Transactions on **5**(4) (aug 1995) 355–359
46. Lim, H., Swartzlander, E.J.: Multidimensional systolic arrays for the implementation of discrete Fourier transforms. Signal Processing, IEEE Transactions on **47**(5) (may 1999) 1359–1370
47. Herzberg, H., Haimi-Cohen, R.: A systolic array realization of an LMS adaptive filter and the effects of delayed adaptation. Signal Processing, IEEE Transactions on **40**(11) (nov 1992) 2799–2803
48. Chang, L.W., Wu, M.C.: A unified systolic array for discrete cosine and sine transforms. Signal Processing, IEEE Transactions on **39**(1) (jan 1991) 192–194
49. Buyukkurt, B., Najj, W.: Compiler generated systolic arrays for wavefront algorithm acceleration on FPGAs. In: Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on. (sept. 2008) 655–658
50. Jin, W., Zhang, C., Li, H.: Mapping multiple algorithms into a reconfigurable systolic array. In: Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on. (may 2008) 001187–001192
51. Lu, L., Liu, W., O'Neill, M., Swartzlander, E.: QCA Systolic Matrix Multiplier. In: VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on. (july 2010) 149–154
52. Lu, L., Liu, W., O'Neill, M., Swartzlander, J.E.: QCA Systolic Array Design. Computers, IEEE Transactions on **62**(3) (2013) 548–560
53. Crocker, M., Hu, X., Niemier, M.: Design and Comparison of NML Systolic Architectures . Nanoarch (2010)
54. Causapruno, G.: Analysis and Optimization of Parallel Processing Architectures for Nanotechnologies . Master's thesis, Politecnico di Torino (November 2012)
55. Rajaram, S., Karunaratne, D., Sarkar, S., Bhanja, S.: Study of Dipolar Neighbor Interaction on Magnetization States of Nano-Magnetic Disks. IEEE Trans. on Magnetics (2013)
56. Panchumarthy, R., Karunaratne, D., Sarkar, S., Bhanja, S.: Magnetic State Estimator to Characterize the Magnetic States of Nano-Magnetic Disks. IEEE Trans. on Magnetics (2013)
57. Deriu, M.A., Shkurti, A., Paciello, G., Bidone, T.C., Morbiducci, U., Ficarra, E., Audenino, A., Acquaviva, A.: Multiscale modeling of cellular actin filaments: From atomistic molecular to coarse-grained dynamics. Proteins: Structure, Function, and Bioinformatics **80**(6) (2012) 1598–1609

58. Shkurti, A., Orsi, M., Macii, E., Ficarra, E., Acquaviva, A.: Acceleration of coarse grain molecular dynamics on GPU architectures. Journal of computational chemistry **34**(10) (2013) 803–818
59. Smith, T., Waterman, M.: Identification of Common Molecular Subsequences. Molecular Biology **147** (1981) 195–197
60. Urgese, G.: Analysis and Design of an Optimized HW Accelerator for Protein Alignment . Master's thesis, Politecnico di Torino (September 2012)
61. Urgese, G., Graziano, M., Vacca, M., Awais, M., Frache, S., Zamboni, M.: Protein Alignment HW/SW Optimizations . The IEEE International Conference on Electronics, Circuits, and Systems (ICECS) (2012)
62. Wang, J.: Emerging Technologies For Biosequence Analysis . Master's thesis, Politecnico di Torino (November 2012)
63. Wang, J., Vacca, M., Graziano, M., Zamboni, M.: Biosequences analysis on Nano-Magnet Logic . International Conference on IC Design and Technology (May 2013)