Asynchrony in Quantum-Dot Cellular Automata Nanocomputation: Elixir or Poison?

Asynchrony in QCA nanocomputation: elixir or poison?

Mariagrazia Graziano *Member IEEE*, Marco Vacca, Davide Blua, Maurizio Zamboni
Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi, 24 Torino, Italy

***Abstract*—Quantum dot Cellular Automata (QCA) technology is a possible CMOS substitute. It requires a clock-like signal to assure information propagation. This likens a QCA circuit to a fully pipelined architecture, where pipeline depth is layout dependent. Only asynchronous organization of architectural blocks enables the realization of complex circuits. The use of Null Convention Logic (NCL) is one among the proposed solutions. However there is no certainty that the balance between positive and negative aspects of this logic will be favorable for QCA. We analyze pros and cons of NCL and Boolean logic circuits applied to QCA, using a VHDL behavioral model of QCA gates, in terms of speed, latency, power and area. We point out a critical problem related to feedback signals: Clearing it is mandatory in order to implement complex QCA circuits. Finally we propose a hybrid logic solution which is the best compromise between performance and problem solutions, and we give a sound answer to this paper title question.**

***Index Terms*—Magnetic Quantum Doc Cellular Automata; magnetic memory; VHDL modelling; Null Convention Logic; Asynchronous architecture; latency insensitive design.**

I. Introduction

Cellular automata principle was recently and successfully applied to electronic digital circuits, leading to the Quantum dot Cellular Automata (QCA) idea [1]. In the general principle a QCA cell has two different charge configurations, representing the two logic values ’0’ and ’1’ (Figure 1.A). These building blocks are placed at small distance on the same plane. The electrostatic interaction between neighbor cells drives the information through the circuit. There are two main implementations of this theoretical principle: molecular QCA, where the cell is a complex molecule, and magnetic QCA [2], where the cell is a single domain nanomagnet (Figure 1.B). Though magnetic QCA (MQCA) allow speeds (hundred of MHz) lower than the perspective molecular ones (few THz), they offer several specific advantages. These include small area, a very low power consumption and the possibility to combine computation and storage [3], but, especially, the experimental feasibility with technology currently available [2]. The International Technology Roadmap of Semiconductor mentions thus MQCA as worth studying in order to proof whether they can be a replacement for CMOS.

It has been demonstrated that the switching from one to the other QCA logic state should be adiabatic [3]. Cells are thus temporarily driven to an intermediate unstable state using an external field [3], a magnetic one in the MQCA case. It reduces the potential barrier between the two stable states and erases the previous value stored in a cell. Releasing the external field means to drive the cells toward a stable hold state. This occurs through a transient switching favored by the influence of the neighbor cell [3] (see section II for details). This *on* and *off* switching likens this field to a *clock* signal. It is provided by a special wire, external to the nanomagnets circuit, in which a current flows with a proper timing generating the magnetic field. It is worth remarking that this signal is far different from the clock normally used in CMOS digital structures. In fact, it just enables the information propagation for every cell throughout the whole circuit. It is not a signal which delivers a synchronization to special gates like registers. To grant the information propagation in every direction, this wire must be routed using a complex layout. At the same time, the physical feasibility of the structure which generates it should not be neglected: Starting from [2], we investigated this problem and discussed a “snake-clock” solution in [4][5].

The unavoidable use of this type of organization leads to two main issues. First, the clock signal gives to QCA circuits a wavefront pipelined behavior and leads to the “layout=timing” problem [1]: The propagation delay of a QCA wire depends on its layout (see section II for a more detailed explanation). To really understand the burden of this critical aspect a comparison could be done with a particular situation in CMOS circuits in which every wire has a pipelined structure. More specifically, a wire pipelined with a depth which depends on routing (i.e. the longer the routing the bigger the number of stages) and not on the circuit logic function. In this situation the standard skewing and deskewing stages are not just a designer choice, they are a constraint complicated by placement and routing of cells. Those constraints could be unsatisfiable in complex circuits, and automatic CAD tools could not be of help to leverage this problem in realistic designs.

A second issue also arises. One could argue that the necessity of this *clock* used to move information does not prevent from having a real clock signal, as in the standard CMOS circuits and completely unrelated to the former one. But this is the point: though not impossible in principle, it would be almost unfeasible in practice for circuits of realistic complexity. Up to now, no effective *direct* solutions to this problem have been proposed. There are two possibilities: Using another external field, or delivering it through the magnetic cells coping with the “layout=timing” constraint. Both, at the moment, are not practicable solutions. The *indirect* approach consists in fronting the fact that this technology does not allow to use the well known primitives of the synchronous world, and to exploit the potentialities of an asynchronous design style. If, in the CMOS digital world, asynchronous

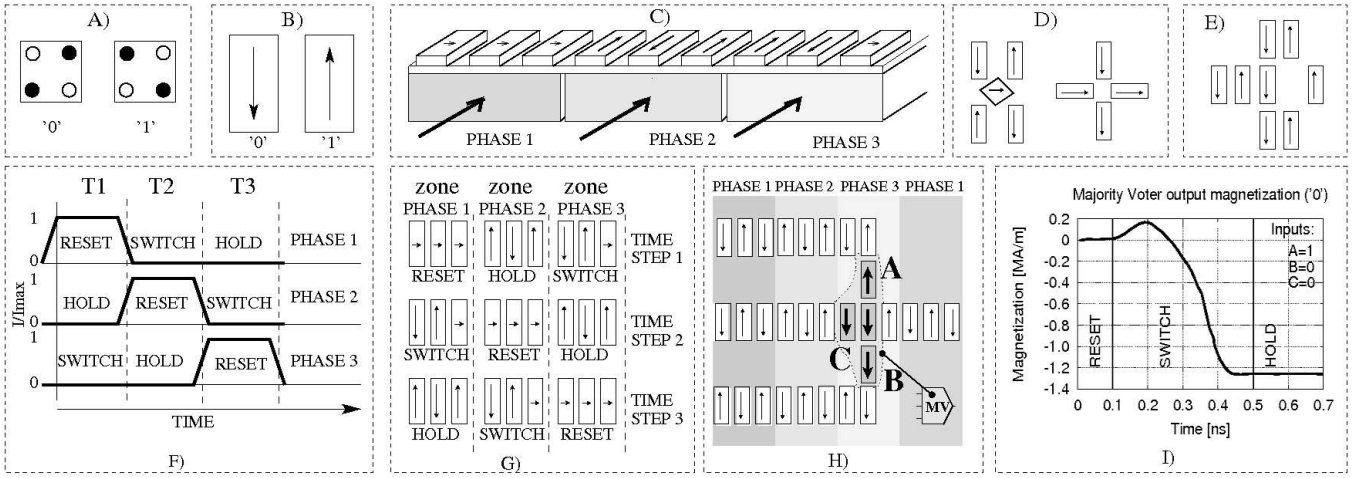

Fig. 1. A) QCA cells. B) Magnetic QCA cells. C) Possible Clock signal distribution below a wire of magnets (see [5] for details). D) Two examples of magnetic wire crossing (see [2],[11] for details). E) A Magnetic QCA inverter. F) Clock signals timing behavior. G) Clock phases sequence and magnets behavior. H) Top view; layout example spanning four clock zones for three wires and a basic logic QCA cell, the Majority Voter: MV = AB + AC + BC (symbol in the bottom right detail). I) Magnetization behavior in time of the output MV magnet obtained using a finite-difference nanomagnetic simulator (NMAG [12]): Magnetization starts form 0, due to an applied RESET field, and then changes (SWITCH) to a negative stable value (HOLD).

systems are often seen as a niche for specific applications, which are now rapidly expanding [8], in the QCA case they are an enabling solution. In asynchronous circuits a block is not only associated to *which* logic function it is executing, but also to *when* it is going to execute it, with respect to the information flow. This is a potentially perfect scenario for QCA. And this is the reason why one of the proposed solutions consists in adopting the asynchronous Null Convention Logic^{TM} (NCL) [6]. It does not need a clock and manages the timing of logic propagation using encoding and exploiting acknowledge signals. It is totally delay insensitive and thus helps solving the "layout=timing" issue. The consequence is then the reduction of synthesis and physical design constraints and the possibility to avoid a real clock. The number and the position of logic gates is constrained like in the standard digital circuits, and the same principles and automatic algorithms can then be adopted. A general NCL implementation applied to QCA circuits is proposed in [7], while a specific solution for magnetic circuits is presented in our work in [4][5] and is the starting point for this work. The proposed system is thus a GALS one: Globally Asynchronous (the handshake protocol to *allow* information propagation) and Locally Synchronous (the clock system to *enable* information propagation). GALS circuits are recently used in traditional designs to successfully leverage the burdens due to interconnect delays [9] or to different synchronization sub-systems [10]. They can thus be effectively adapted to nanotechnology designs to enlighten their real potentialities.

Despite the beneficial effects of NCL to the functionality, our preliminary investigations show that NCL logic applied to QCA technology have several consequences, as already demonstrated in previous works [8]. The plain application of this logic is then not necessarily a panacea. Thus, after a brief background on QCA clock system in section II, we present (section III) the complete comparison, never attempted before, between a fully synchronous QCA implementation based on standard Boolean logic and a GALS QCA solution based on NCL logic. The comparison includes speed, latency levels, power dissipation and area. It is based on a VHDL behavioral model of QCA circuits we developed [5], and applied to two specific circuits: a 32-bit ALU and a parallel memory with 4 address bits and 14 data bits. A discussion follows on a problem (section IV) related to feedback signals. This arises from a layout and technology aware design of a complex circuit like a microprocessor. We thus propose a hybrid solution (section V), between NCL and Boolean logic, that solves the feedback problem and which, though not the only possible solution, is demonstrated being a good compromise between performance and feasibility.

II. Magnetic QCA clock system

Magnetic QCA circuits are built using single domain nanomagnets with only two stable magnetizations (Figure 1.B). This is favored by their rectangular structure which implies an evident shape anisotropy. The magnetization vector is parallel to the long side (easy axis) of the nanomagnets, and this state is difficult to change. To switch a nanomagnet from a state to another a strong magnetic field is required, called *clock*. It is directed along the short side (hard axis) of the nanomagnet. When applied, it forces the nanomagnets in an unstable state: Their magnetization is redirected along the hard axis. When the field is removed, nanomagnets realign themselves in an antiferromagnetic order along the easy axis.

To avoid information propagation errors during the reordering, only a small number of nanomagnets (between 10 and 20 [2]) can be placed together. Therefore the circuit plane is divided into small areas, each influencing a limited number of nanomagnets. In each area they can be organized in a simple sequence, like in Figure 1.C, which represents a wire; or in more complex structures, like in Figure 1.D, where two examples of crossing between two wires proposed in literature are given [2][11]; or even in blocks able to execute a logic function, like the inverter in Figure 1.E, or the Majority Voter (see later on).

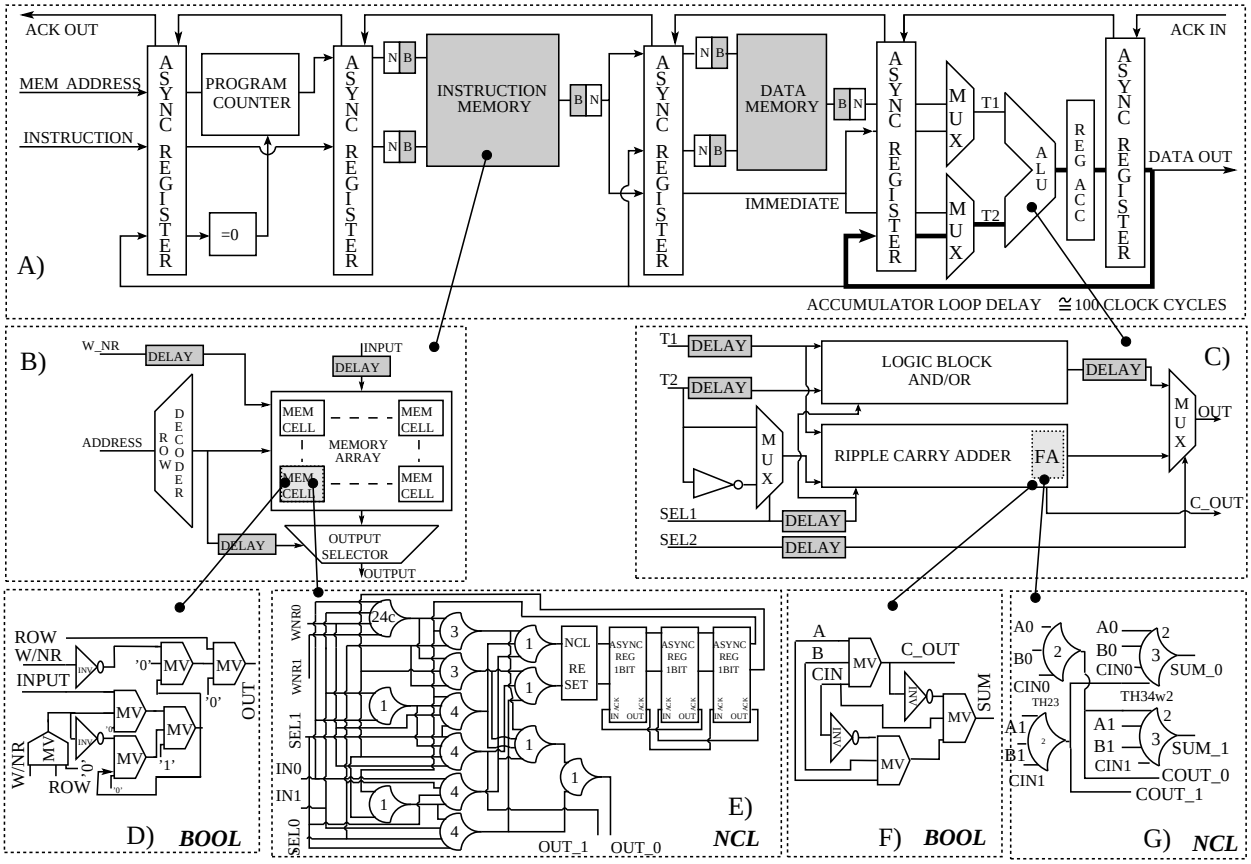

Fig. 2. A) Microprocessor structure. B) Instruction memory architecture. Delay blocks (gray colored) are only present in the Boolean implementation. C) ALU architecture Delay blocks (gray colored) are only present in the Boolean implementation. D) Memory cell structure based on Majority Voters (MV) in a Boolean implementation. E) Memory cell structure based on NCL logic gates, themselves based on MV [5]. F) Full Adder structure based on Majority Voters (MV) in a Boolean implementation. G) Full Adder structure based on NCL logic gates, themselves based on MV [5].

To drive the information through the circuit, independently on the logic function, a multi-phase clock system is necessary. We have proposed a 3 phases snake-clock [4],[5]. The area is split in groups of sub-areas, each organized in three clock zones (a simplified example is in Figure 1.C), driven by a different signal. The signal waveforms, i.e. pulses with a phase shift of 120 degrees, are shown in Figure 1.F. The sequence of three clock phases is repeated along the entire circuit, with order 1-2-3-1-2-3.

Figure 1.G shows the magnets operations according to clock sequences. In the first time step the cells of the second phase are in a HOLD state: No external field is applied and they are therefore in a stable state. This has a great influence on the nanomagnets of the following (the third) clock zone, that are in a SWITCH state. These magnets reorder themselves following the nanomagnets of the second clock zone, which act like an input signal. At the same time, magnets in the previous zone, the first, are in a RESET state: This means that the external field is applied, their magnetization is directed along the short axis, and they have a very small influence on the clock zone 2. In the further time step, this situation is repeated, but with the first clock zone (which is the next in the clock zone sequence 1-2-3-1-2-3) in the SWITCH phase, the second in the RESET and the third in the HOLD state. The information moves along the circuit following the clock zone sequence. Figure 1.H shows an example of a top view where three signals are routed through three zones, and carry information to a basic QCA logic gate: the Majority Voter (MV=AB+AC+BC). In Figure 1.I the magnetization transient of the MV output (the central element in the evidenced MV sub-block) is depicted. This is obtained by an accurate finite-difference nanomagnetic simulator [12]. The case reported here corresponds to the input values shown by arrows in Figure 1.H (A=1, B=0 and C=0), so the output is expected to go to '0'. Indeed, the magnetization starts from a zero value, as the magnet was previously in a RESET state; afterwards it switches to a negative magnetization value, the logic 0, which is then maintained in the hold state. The delay found here depends on the magnets aspect ratio, on their horizontal and vertical distances, and on the magnetic material used (typically Permalloy or Cobalt). The maximum clock frequency is bounded not only to this delay, but also to the delay of the nanomagnets representing the wire that carry the information within the same phase. So the sum of all the delays of magnets within a zone during the SWITCH phase roughly defines 1/3 of the clock period. To achieve fast frequencies, then, a limited number of magnets should be placed within a phase zone. Nevertheless, the smaller the phase zone, the smaller is the size of the wire delivering the clock. Typical sizes of magnets are $50nm \times 100nm$ with a space of $20nm$ between two of them. It is then easy to see

an opposite constraint: The higher is the number of magnets in a zone, the easier will be the clock fabrication, at least as long as nanowires can not be really used for this purpose. The estimated realistic frequency reported in literature is around 100MHz [3].

It is thus clear that in a QCA circuit, even in the case of a simple wire, which crosses many clock zones, the information propagates with a delay of 1 clock cycle for every group of three zones. This is an intrinsic pipelined behavior and makes clear a fundamental QCA property: The length of a wire depends on the number of clock zones it crosses, and the propagation delay of a wire depends on its length. The consequence is the complexity in designing QCA circuits based on synchronous Boolean logic. The length of the wire at the inputs of every logic gates must be equalized to synchronize signals ("layout=timing"). This is in theory feasible only using specific algorithms, but for complex circuits the constraints could be unbearable.

As mentioned in the introduction, a possible solution could be using a real clock signal, i.e. a further signal which delivers synchronization to blocks similar to D-Flip-Flops in CMOS circuits. The D-FF would delay the progress of data until a synchronization signal would be sampled. Though it is natural to think at it as similar to CMOS structures, it is difficult to be implemented. As long as we want to stick to a solution which is really feasible with current technology, this solution has to be set aside at present time.

A balm to such a situation is to conceptually split information propagation into two levels. The first is strictly connected to the physical signal propagation, which must be synchronous. The other is the logic signal propagation. The latter can rely on an asynchronous delay insensitive logic. One of the possible choices is the NCL^{TM} [6]. Other asynchronous implementations could fit this problem. We started this analysis from NCL considering that it was proposed in [7] as a promising solution, but that its benefits and aftereffects have not been established jet in terms of realistic performance.
In NCL every signal is coded using two bits, that can assume two different values: DATA = 01 or 10 (that stand for a Boolean '0' and '1') and NULL = 00, while 11 is forbidden. Circuits switch periodically from NULL to DATA, and viceversa. The advantage is that the switch of a gate in either directions occurs only when all the inputs assume the same coherent value (all NULL to DATA or, in the opposite case, all DATA to NULL). The delay insensitivity is thus assured, at the cost, of course, of an increased complexity. The consequences of this choice are twofold. First, it is not necessary to deliver a synchronization signal; second, gates can be placed without worrying about delays and synchronization, and thus top-down synthesis and physical design can be inherited from CMOS circuits design flow. Currently a few attempts to this purpose have been done in literature in the general QCA case, both for synthesis and for placement. Although a lot of work has to be done, especially in the magnetic case, results show that it is possible to rely on these algorithms.

NCL logic is based on several basic cells: Their detailed behavior can be found in [6], while their application to a magnetic QCA circuit can be found in [4][5]. It is worth underlying that the two bits encoding doubles the wires, and as a consequence, many crossing points could be necessary. Though this is a complication, it can be faced, as coplanar wire crossings (examples are in Figure 1.D) have been experimentally demonstrated [2][11].

In this paper we aim at clarifying whether this asynchronous circuit organization is an effective solution for QCA. We compare Boolean and NCL logic using our VHDL behavioral model [4][5]. Logic gates are considered ideal, with no delay, but the wire propagation delay through the clock zones are simulated using a register for every phase zone, having as a clock the correspondent clock signal of the zone (as in Figure 1.F), and thus reproducing the pipelined circuit behavior. Our model is based on the realistic physical structure of every NCL gate, basing it on the physically feasible "snake-clock" we proposed in [5]. Every gate is carefully designed in order to be feasible, and the VHDL description reflects this design.

We have improved this model allowing for a hierarchical estimation of the circuit area and power dissipation. The structure of this model is not described here in details for sake of simplicity, as not the aim of the paper. The model is based on the real number of magnets of the basic logic gates, and hierarchically estimates the total number of nanomagnets in a parametric way, in order to obtain a realistic approximation. The power dissipated by the nanomagnets is then calculated multiplying their total number for the power dissipated by each one (approximately $30 - 40K_BT$ [3]). Starting from the total number of magnets we also evaluate the circuit area, using the magnets dimensions and some parameters to take into account wasted space. Using the reckoned circuit area, and knowing the clock zone dimensions, we can estimate the length of the clock wires and their power dissipation due to joule effect. The power is estimated using the most efficient clock generation system currently proposed in literature [13].

III. Synchronous versus asynchronous

In this work, a reference architecture is a microprocessor, described in section V and sketched in Figure 2.A. For a detailed comparison between a fully synchronous and an asynchronous solution we chose two of the main processor components: the Arithmetic Logic Unit (Figure 2.C) and the instruction memory (Figure 2.B), both discussed herein.

A. ALU

The ALU organization is the same in both Boolean and NCL cases at the higher hierarchical level, the only difference being the delay blocks (gray in Figure) added to the Boolean solution to synchronize signals. The architecture is a simple ripple carry adder for addition and subtraction, and a logic block for AND/OR operations. The output multiplexer selects between logical and arithmetical operations, while the input multiplexer selects, if needed, the negated operand for two complement's subtraction.

Each Full Adder is based, for the two logic cases, on the structures in Figure 2.F and 2.G respectively. The Boolean solution is implemented starting from the MV; NCL circuit relies on NCL gates, internally based on MV. Details on this

structure can be found in [5]. It is enough to note here that double wires for each logic signal are present and that, for example, TH23 gate has function OUT=MV(B,C,OUT)+A. The internal feedback is necessary to assure the delay insensitivity.

Data are represented using 32 bits. Simulation results are in Figure 3 and 4 for the Boolean and NCL version respectively. As previously mentioned the behavior is intrinsically pipelined

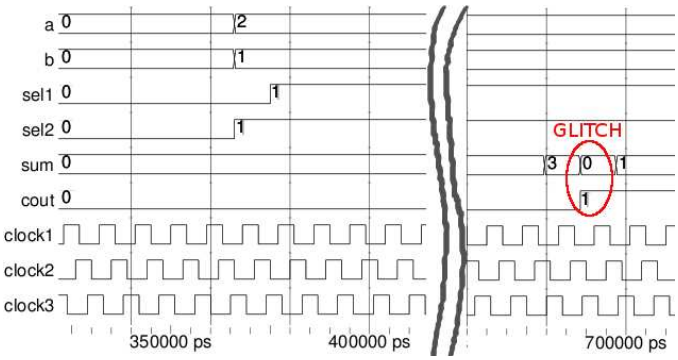

Fig. 3. Boolean logic ALU simulation results: 2+1 and 2-1 operations are shown. Signal SEL1 defines addition or subtraction.

as evident in Figure 3. At every clock cycle a data is accepted at the inputs, and an output is correspondingly generated with a high latency. The Boolean implementation of the alu is troublesome. Signals must be precisely synchronized in order to obtain working circuits, otherwise problems arise as discussed in the following. In this simulation the delay of the synchronizing blocks was intentionally altered to show how critical is this problem. In the example chosen, two logic operations are performed, first the addition 2+1, and then the subtraction 2-1, according to selection signal SEL1. The circuit has a long latency due to its intrinsic structure, and results are available after a long delay (skipped in Figure). The first available output is correct (3 due to the addition). During the next clock cycle, the output correctly changes because of the pipeline, but the result shown is wrong. At the further clock cycle the output changes again and now the results is correct. This behavior is caused by a mismatch in the propagation time of the first selection bit, which internally changes one clock cycle after the input signal, generating the glitch (this is a demonstration of the “layout=timing” issue). It is worth noticing that this glitch has the same importance that it has in CMOS circuits when for sure it is sampled by a flip-flop; It is an error which propagates throughout the circuit.

The use of NCL logic totally eliminates this problem. An example is in Figure 4 where two operations are shown using NCL encoding, a logic OR and an addition (bottom details show the Boolean conversion for clarity). It is worth noticing how every signal is encoded using two bits and how signals periodically switch from DATA to NULL: A new data is accepted only when all the inputs switch to the NULL state independently from their delay. In this way circuits work normally also in presence of different propagation delays.

NCL logic, like every asynchronous logic, requires a communication protocol to operate. This can be gathered by Figure 2.A, where every block of combinational logic is embraced by two asynchronous registers (a transmitter TX and a receiver RX) that generate and exchange this handshake protocol:

- A DATA is propagated from a register output (TX) to the input of the next one (RX) through the combinational circuit.
- At this point register RX receives the DATA and sends back an acknowledgement (ACK) to the previous register (TX).
- When ACK is received at TX, a NULL (all the outputs to ’0’) is sent through the combinational circuit.
- RX receives the NULL and sends back another ACK signal.
- Once this second ACK signal is received TX register is ready to accept a new data from its combinational input.

So, the behavior of QCA circuits is pipelined for what concerns magnetic signal propagation, but the asynchronous protocol freezes the circuit from the logic point of view and accepts a new data only after the completion of the DATA-NULL cycle. An important remark is due at this point. The propagation time of the signals through the circuit and the propagation time of the ACK signal are equal to the latency of the combinational circuit. This means that an asynchronous register accepts a new data only after a time equal to 4 times the circuit latency (one time for the propagation of the DATA, one time for the propagation of the NULL and two times for the propagation of the ACK signals).

Table I shows the comparison between the two ALUs in terms of latency, area and power dissipation due to nano-magnets switching. The power dissipation increased, but the area occupied by the NCL version is more than two times bigger. This is easy to explain with the two bits coding of the NCL logic, and the relative additional interconnections overhead. The latency of the circuits is also double than the Boolean one. Therefore NCL logic solves the “layout=timing” issue allowing a less constrained design flow through standard design automation algorithms. This comes at the price of increasing the area of the circuit and slowing down the operations. As previously remarked, a Boolean QCA circuit accepts a new data after each clock cycle. On the contrary a QCA NCL accepts a new data after a time equal to a multiple of the latency. This time is itself bigger than the latency of the Boolean version.

B. Parallel Memory

We run the same type of comparison for a parallel memory (Figure 2.B) organized as a 16x14 matrix (the microprocessor instruction memory). The structure is quite simple. A decoder is used to select the desired row of the matrix and the correspondent output of the memory. Like in the ALU, delay blocks (colored in gray) are used only in the Boolean version to synchronize signals. A detail for a memory cell for the two logic types are in Figure 2.D and 2.F.

The waveforms are not shown for sake of brevity, but table I shows a comparison between the two implementations. Here the difference between the two logic choices is huge. This is due to the complexity of the memory cell implemented in NCL logic as evident in Figure 2.F. The power consumption

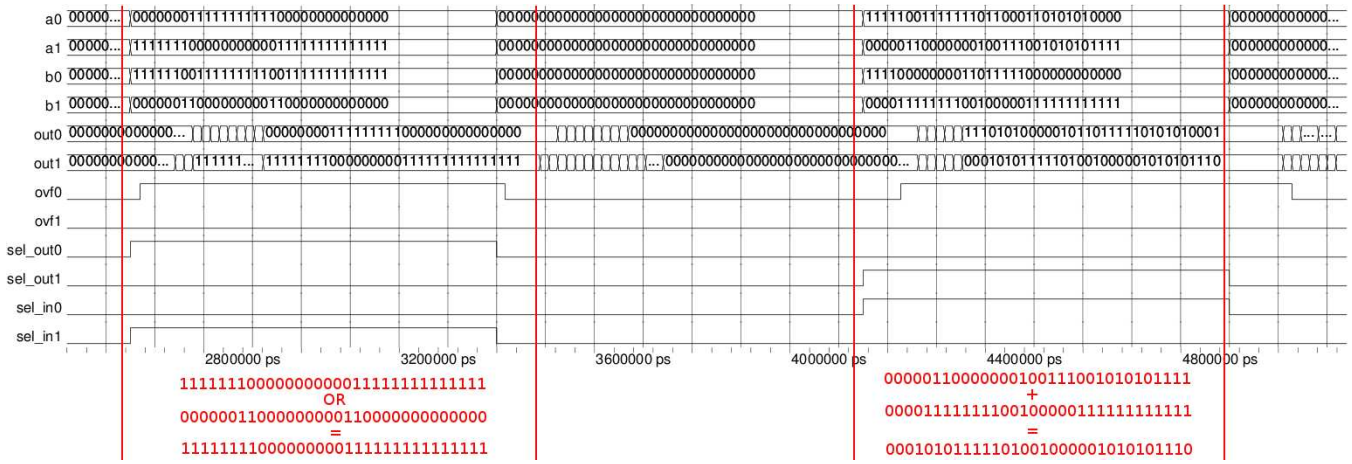

Fig. 4. NCL logic ALU simulation results: an OR and and addition are shown. Top waveforms are NCL encoded, while bottom details are their Boolean translation.

of the memory is around 100 times bigger than the Boolean one, and the area is 44 times bigger. The power dissipation due to the clock wires, estimated using the area, the zones width and the technological choices in [13] for what concerns the magnetic field application, are reported in the last column in table. Data are on the same order of the magnets power consumption, and maintain in all the cases a trend similar to the one obtained for the magnets.
The difference in terms of latency is not so big but still high (about 6 times). Notwithstanding the higher memory complexity, the latency of both memories is lower than the two ALUs. This is caused by the choice of the ripple carry adder, which is a high latency circuit.

Further optimizations on the NCL memory architecture are possible, and therefore we can expect a performance improvement. However this results show that NCL logic is not suited for memory structures (at least applied to QCA technology), as it could severely worsen results even defeating the advantages of adopting this technology. A Boolean memory would be better to use in this case, provided that a good input signal synchronization is assured. In the case of a memory it is easier to assure the absence of glitches. This because of it its higher regularity, because the cells are small, and because given a memory style choice, only the parallelism could change, without impacting the relations or delays among cells. Clearly, until a physical realization is not done, this assumption cannot be really proven.

TABLE I
ASYNCHRONOUS VERSUS SYNCHRONOUS COMPARISON

	Area [um^2]	Latency [n. clock cycles]	Power [uW]	Power Clock [uW]
ALU Bool	1.33	34	0.52	0.86
ALU NCL	2.64	72	1.04	1.65
Mem Bool	1.04	6	0.39	0.65
Mem NCL	44.38	26	36.60	27.66
Microprocessor	10.60	426	3.88	6.62

IV. FEEDBACK IN QCA CIRCUITS

Working with a complex circuit like a microprocessor allows us to pinpoint a negative characteristic which is typical of pipelined circuits, but amplified in QCA technology. To focus on an example we can consider the structure shown in the right section of Figure 2.A. Here one of the ALU inputs is connected, using a feedback signal, to its output. The ALU connected in this way performs the addition between an input and the result of the previous operation. However, since the circuit is intrinsically pipelined, it accepts a new data at every clock cycle, but, as shown in Figure 2.A, the feedback loop has a propagation delay of ≈ 100 clock cycles due to the number of clock zones it crosses in this example. Therefore at the next clock cycle, it performs the addition between an input and the result of the operation occurred 99 clock cycles before.

This problem is well known, as it is typical of conditional jumps in RISC microprocessors. However, in the case of QCA circuits, it is heavily amplified by the high pipeline stages and by every loop in the circuit. Only the adoption of an asynchronous logic like NCL can solve it, as the computation is performed only when all the signals arrive to the inputs of the circuit. This means that a new ALU operation is executed only when the result of the previous one has passed through the feedback loop and arrived at the inputs of the ALU itself.

V. MICROPROCESSOR: MIXED LOGIC SOLUTION

On the basis of previous considerations we can claim that for QCA technology the Boolean logic in a synchronous environment is the best theoretical solution to obtain maximum performance. However, implementation related aspects, like delay synchronization and feedbacks in sequential circuits, prevent its actual applicability. If the first issue could in some cases be overcome with the development of an ad-hoc algorithm to automatically synchronize signals (when possible), the second issue can be solved only using NCL logic, and accepting to lose performance.

We thus propose a solution to achieve the best compromise between circuits feasibility and performance optimization: a

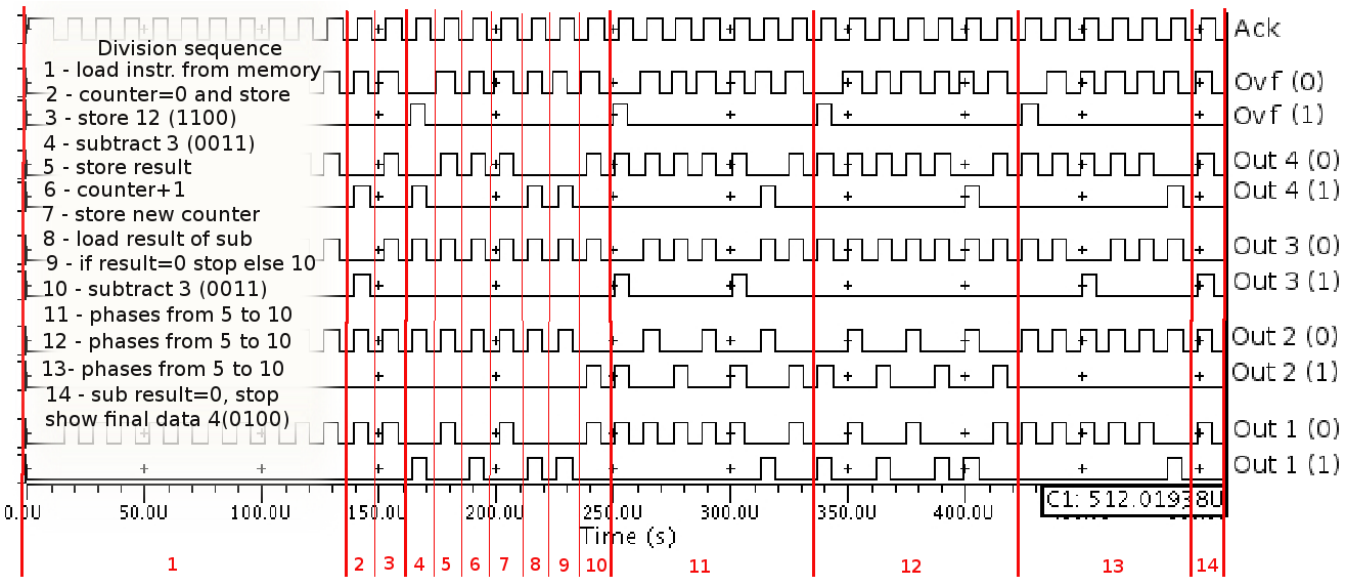

Fig. 5. Division algorithm execution: 12/3=4. In the left inset a description of the phases numbered in the bottom of the Figure.

mixed Boolean-NCL logic. The asynchronous structure is used for the global architecture, leaving the synchronous solution for sub-blocks (for example the memory) where NCL would critically compromise results. Such approach requires the use of appropriate interfaces between the two logic topologies.

To test the proposed solution we implemented a simple four bit microprocessor (Figure 2.A) composed by four main components: a program counter, a parallel memory able to store 16 instructions of 14 bits, a data memory with 4 memory cells of 4 bits each, and the above mentioned ALU. The microprocessor is organized in 4 asynchronous (logic) pipeline stages, where a pipe stage is a combinational circuit enclosed within two asynchronous registers, which solve the feedback problem. In this implementation, only the two memories use Boolean logic (grey in Figure) as the most critical if NCL based.

A division algorithm allows to test the architecture: a 12/3 division is reported in the example in Figure 5. The waveforms are organized in phases from 1 to 14 for sake of clarity. The phases are briefly described in the Figure inset (left).

The microprocessor performance are shown in table I. The latency is high, due to the design complexity and the lack of optimization, anyway an interesting comparison can be made with the parallel NCL memory alone. The whole microprocessor is 4 times smaller and it has 10 times less power dissipation than the memory NCL alone. This clearly shows that our approach is correct, as it allows to build every kind of QCA circuits without losing too much performance. Clearly, the memory must be carefully designed in order to synchronize delays. However this is not complex as for other blocks due to intrinsic regularity.

A final remark can be done on power dissipation. An evaluation has been done to compare this QCA mixed solution to an equivalent CMOS one. We have implemented, using CMOS technology, a microprocessor with the same structure and operating at the same frequency of 100MHz, as our QCA processor. We have synthesized it on a 45nm standard cell technology and we have calculated its total power dissipation, which results in $536\mu W$. The QCA solution is then advantageous: As shown in table I, it dissipates just $3.88+6.62=10.5\mu W$ in the mixed case. The power is estimated as explained in section 2 and it is worth to remind that it is based on several constraints and parameters chosen to obtain a realistic evaluation.

VI. Conclusions

This study soundly allows to answer to this paper title question. A totally synchronous architecture in QCA technology, thought granting high data throughput, would be unfeasible, as it would require really complex synchronization procedures to solve the ”layout=timing” constraint and could be applied to combinational circuits only. Decisive relief is granted if a global asynchronous circuit organization, for example based on Null Convention Logic, is adopted. No synchronization of signals and both combinational and sequential circuits with any order of feedback can be implemented.

As for all medications, collateral effects may arise, especially if dosage is not respected: Circuits are bigger, slower and more power hungry (due to the increased complexity). If a poisonous effect is to be avoided, trade-offs must be carefully evaluated, and, when a block regularity reduces the burden of signal synchronization, synchronous blocks can coexist with asynchronous ones.

This solution is a compromise between performance and circuits feasibility. It is also clear that QCA technology is best suited for pure combinational circuits, were it can grant a consistent advantage over CMOS technology, in terms of speed and especially power dissipation. General purpose circuits are feasible using the asynchronous approach, but at the price of lowering the overall performances. However, results showed, even in this case, advantageous.

Our future efforts will be directed toward finding a different solution, still asynchronous but not based on NCL logic. The aim is clearly to obtain a significant leverage of the NCL burdens. At the same time our efforts will be directed towards an automatic circuit synthesizer, placer and router for boolean QCA logic circuits.

REFERENCES

[1] C. S. Lent, P.D. Tougaw, W. Porod and G.H. Bernstein "Quantum cellular automata", *N*anotechnology, Vol. 4, 49, 1993.

[2] A. Imre, G. Csabaa, G.H. Bernstein, W. Porod and V. Metlushkob, "Investigation of shape-dependent switching of coupled nanomagnets", Superlattices and Microstructures, 34, 513-518, 2003.

[3] M.T. Niemier, X.S. Hu, M. Alam, G. Bernstein, W. Porod, M. Putney nad J. DeAngelis, "Clocking Structures and Power Analysis for nanomagnet-Based Logic Devices", proc. ISLPED, August 2007

[4] M. Vacca "Nanoarchitectures based on magnetic QCA", *M*aster Thesis, Politecnico di Torino, November 2008.

[5] M. Graziano, M. Vacca, A. Chiolerio and M. Zamboni "A NCL-HDL Snake-Clock Based Magnetic QCA Architecture", IEEE Transaction on Nanotechnology, accepted, DOI:10.1109/TNANO.2011.2118229.

[6] K.M. Fant and S.A. Brandt., NULL Convention Logic^{TM}: "A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis", Proc. Int. Conf. on Application Specific Systems, Architectures, and Processors, 1996.

[7] E. Tabrizizadeh, H.R. Mohaqeq and A. Vafaei, "Designing QCA Delay-Insensitive Serial Adder", Proc. IEEE Int. Conf. on emerging trends in Engineering and Technology, 2008.

[8] J. Sparsø and S. Furber, "Principles of asynchronous circuit design - A systems perspective" Kluwer Academic Publishers, 2001.

[9] M. R. Casu and L. Macchiarulo, "Adaptive Latency-Insensitive Protocols", IEEE Des. & Test of Comp., Vol. 24 , I. 5, 2007, pp. 442 - 452.

[10] M. Martina and G. Masera, "Turbo NOC: A Framework for the Design of Network-on-Chip-Based Turbo Decoder Architectures", IEEE Tran. on Circuits and Systems I, Vol. 57, I. 10, 2010, pp. 2776 - 2789.

[11] J. Pulecio and S. Bhanja, "Magnetic cellular automata coplanar cross wire systems", J. Appl. Phys. Vol. 107, Issue 3, 2010.

[12] T. Fischbacher, M. Franchin, G. Bordignon, and H. Fangohr. "A Systematic Approach to Multiphysics Extensions of Finite-Element-Based Micromagnetic Simulations: Nmag", in IEEE Transactions on Magnetics, 43, 6, (2007). (Available online)

[13] Augustine, C. Fong, X. Behin-Aein, B. Roy, K., "Ultra-Low Power Nano-Magnet Based Computing: A System-Level Perspective", IEEE Transaction on nanotechnology, Volume: PP, Issue: 99, 2010.