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

Mariagrazia Graziano, Marco Vacca, Davide Blua, and Maurizio Zamboni
Politecnico di Torino

Editor's note:
Emerging computing technologies inherently exhibit high process and timing variation. Many researchers believe that an asynchronous approach is likely to play an enabling role in making these technologies feasible. This article compares the cost and performance of fully synchronous and mixed synchronous-asynchronous implementations of quantum cellular automata, and makes the case that asynchrony is inevitable at the top levels of QCA designs.
—Montek Singh, UNC Chapel Hill

THE CELLULAR AUTOMATA principle has been successfully applied to electronic digital circuits, leading to the quantum-dot cellular automata (QCA) concept.[1] The general principle of this concept is that a QCA cell has two different charge configurations, representing the two logic values 0 and 1 (see Figure 1a). These building blocks are placed a short distance from each other on the same plane. The electrostatic interaction between neighbor cells drives the information through the circuit. Two main implementations of this theoretical principle exist: molecular QCA, in which the cell is a complex molecule, and magnetic QCA (MQCA),[3] in which the cell is a single-domain nanomagnet (see Figure 1b). Although MQCA allow speeds (hundreds of MHz) lower than the molecular ones (a few THz), they offer several specific advantages. These include small area, low power consumption, and the possibility to combine computation and storage on the same circuit.[5] Most importantly, though, they offer experimental feasibility with technology that's currently available.[3] The International Technology Roadmap of Semiconductors thus mentions that MQCA are worth studying to prove whether cellular automata can generally replace CMOS (see http://public.itrs.net/Links/2010ITRS/Home2010.htm).

QCA might offer many advantages in replacing CMOS, but many issues and constraints come with this technology (as we detail in the "Considerations and Constraints" sidebar). Asynchrony has been introduced when designing QCA circuits at the architectural level, and it's tempting to think of this approach as a magical elixir. Is it, though? Or, because of the overhead it introduces, is this approach more of a poison? In this article, that's exactly what we intend to discern.

Specifically, we're interested in applying and testing Wave Semiconductor's asynchronous Null Convention Logic (NCL) to QCA technology. Tabrizizadeh et al. proposed a general NCL implementation applied to QCA circuits,[6] whereas we presented a specific solution for magnetic circuits in previous work.[2,7] That research informed the system that we currently propose, which is globally asynchronous (the handshake protocol to allow information propagation) and locally synchronous (the clock system to enable information propagation). Some traditional designs have used globally asynchronous, locally synchronous (GALS) circuits recently to successfully leverage the burdens caused by interconnect delays[8] or by different synchronization subsystems.[9] They can thus be effectively adapted to nanotechnology designs to enlighten their real potential.

0740-7475/11/$26.00 © 2011 IEEE Copublished by the IEEE CS and the IEEE CASS

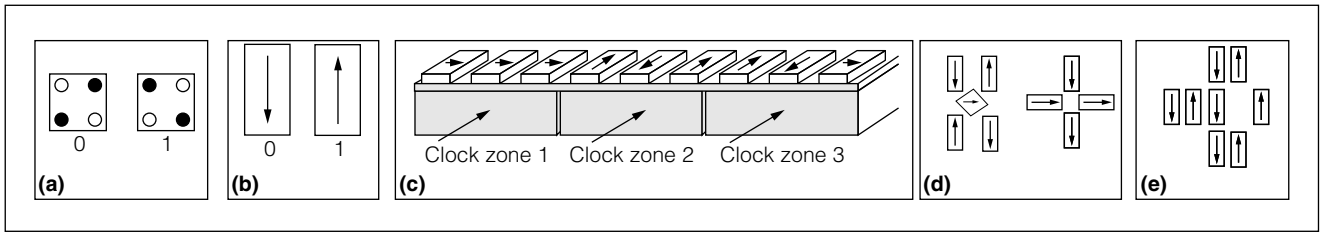

Figure 1. Elementary quantum-dot cellular automata (QCA) structures (a). Magnetic QCA (MQCA) cells (b). Possible clock signal distribution below a wire of magnets (c).[2] Two examples of magnetic wires crossing (d).[3,4] MQCA inverter (e).

Despite the potentially beneficial effects of NCL for QCA circuits' functionality, our preliminary investigations based on simulations show that NCL applied to QCA technology has several consequences, as demonstrated in previous work.[10] The plain application of this logic is not necessarily a panacea. With this in mind, we present a complete comparison (which, as far as we know, has never been attempted before) between a fully synchronous QCA implementation based on standard Boolean logic and a GALS QCA solution based on NCL. The comparison includes speed, latency levels, power dissipation, and area. It's based on a VHDL behavioral model of QCA circuits that we developed[2] and applied to two specific circuits: a 32-bit ALU and a parallel memory with four address bits and 14 data bits. We also consider a problem related to feedback signals that can occur in layout- and technology-aware designs of complex circuits such as microprocessors.

MQCA clock system and NCL

MQCA circuits are built using single-domain nanomagnets with only two stable magnetizations (see Figure 1b). This is favored by their rectangular structure, which implies an evident shape anisotropy. The magnetization vector is parallel to the nanomagnets' long side (it is called an *easy axis* because the magnetic energy assumes a minimum in the longer of two sides), and this state is difficult to change. A strong magnetic field (called a clock) is required to switch a nanomagnet from one state to another. The field is directed along the nanomagnets' short side (hard axis). The field, when applied, forces the nanomagnets into 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 realignment, only a small number of nanomagnets (between 10 and 20)[3] can be placed together. Therefore, the circuit plane is divided into small areas, each influencing a limited number of nanomagnets. The nanomagnets can be organized in one of several ways: in a simple sequence within each area (see Figure 1c, which represents a wire); in more complex structures (see Figure 1d, which shows two examples of crossing between two wires);[3,4] or in blocks able to execute a logic function, such as the inverter in Figure 1e or the majority voter (or MV), which we detail later.

A multiphase clock system is necessary to drive information through the circuit independently on the logic function. We have proposed a three-phased snake clock for this purpose.[2,7] The whole circuit area is split in groups of subareas, each organized in three clock zones (a simplified example is in Figure 1c), driven by a different signal. Figure 2a shows the signal waveforms, such as pulses with a phase shift of 120 degrees. The sequence of the three clock phases associated to as many clock zones is repeated along the entire circuit in an order of 1-2-3-1-2-3.

Figure 2b shows the nanomagnets' operations according to clock phases. In the first time step, the second clock zone's cells are in a hold state: no external field is applied and they're therefore in a stable state. This significantly influences the following (third) clock zone's nanomagnets, which are in a switch state. These magnets reorder themselves following the second clock zone's nanomagnets, 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 small influence on clock zone 2. In the next 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,

Considerations and Constraints

Researchers have demonstrated that switching from one quantum-dot cellular automata (QCA) logic state to another should be adiabatic.[1] By means of an external field, cells are thus temporarily driven to an intermediate unstable state,[1] a magnetic one in the magnetic QCA (MQCA) case. The external field reduces the potential barrier between the two stable states and erases the previous value stored in a cell. When a signal releases the external field, this drives the cells toward a stable hold state. This transition toward a new hold state occurs through transient switching, which is favored by the neighbor cell's influence.[1] This on-and-off switching likens this field to a clock signal. A special wire provides the switching. The wire is external to the nanomagnets' circuit, in which a current flows with proper timing generating the magnetic field. Note that this signal is far different from the clock normally used in CMOS digital structures. In fact, the signal's only job is to enable information propagation for every cell throughout the whole circuit—it's not a signal that delivers synchronization to special gates like registers. To propagate information in every direction, this wire would have to be routed using a complex layout. At the same time, the physical feasibility of the structure that generates it can't be ignored: in previous work, we investigated this problem and discussed a ''snake clock'' solution, which we briefly discuss in the main text of this article.[2-4]

The unavoidable use of complex clock organization leads to two main issues. First, the clock signal gives QCA circuits a wavefront pipelined behavior and leads to the ''layout = timing'' problem:[5] the propagation delay of a QCA wire depends on its layout. We can better understand the crucial impact of a QCA wire's layout by comparing it with a situation in CMOS circuits in which every wire has a pipelined structure. More specifically, we mean a wire pipelined with a depth that depends on routing (i.e., the longer the routing, the more stages) and not on the circuit-logic function. In this situation, the standard skewing and deskewing stages aren't just a design choice, they're a constraint complicated by cell placement and routing. Those constraints could be unsatisfiable in complex circuits, and automatic CAD tools wouldn't help leverage this problem in realistic designs.

A second issue also arises. We could argue that the necessity of this snake clock used to move information through the external field doesn't prevent the circuit from having a real clock signal, as is the case in standard CMOS circuits. However, although it's not impossible to have a real clock signal, it would be almost unfeasible in practice for circuits of realistic complexity.

Currently, no effective direct solutions have been proposed to provide a clock meant as a synchronization

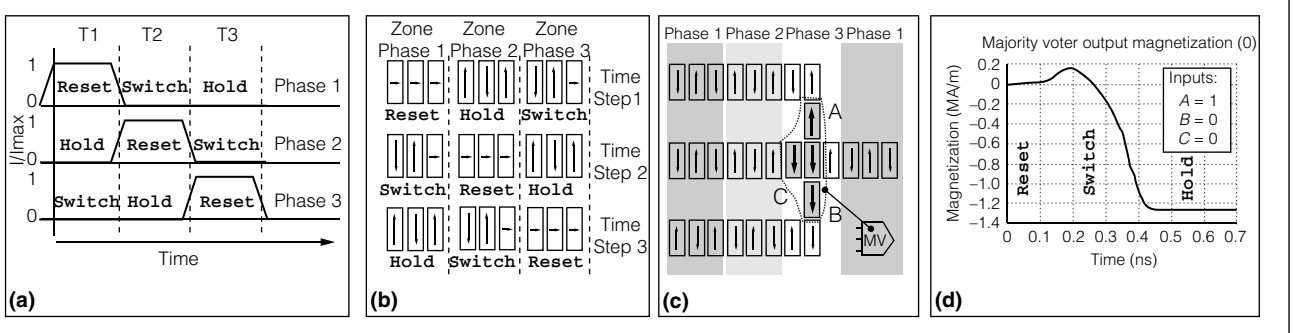

Figure 2. MQCA clock system. Clock signals' timing behavior (a), where the time slots are listed as T1, T2, and T3. Clock phase sequence and magnets' behavior (b). Top view of a layout example (c) spanning four clock zones for three wires and a basic logic QCA cell, the majority voter: MV = AB + AC + BC (symbol in the bottom right detail). Magnetization behavior (d) in time of the output MV magnet obtained using a finite-difference nanomagnetic simulator:[11] magnetization starts from 0, because of an applied reset field, and then changes (switch) to a negative stable value (hold). Values are shown as mega ampere per meter (MA/m).

signal (rather than having a real, traditional clock signal). Instead, there are two other possibilities: using another external field, or delivering a traditional synchronization signal using the magnetic cells to route the traditional clock signal through the magnetic cells coping with the layout = timing constraint. Both, at the moment, aren't practicable solutions.

The indirect approach consists in confronting the fact that this technology doesn't let us use the well-known primitives of the synchronous world, and exploiting the potential of an asynchronous design style. If, in the CMOS digital world, asynchronous systems are often seen as a niche for specific applications (which are now rapidly multiplying),[6] in the QCA case they're an enabling solution. In asynchronous circuits, a block is not only associated with the logic function that it executes in an information flow, but also with the time at which it's going to execute it.

Embedding the execution's timing in asynchronous circuits' information is potentially a perfect scenario for QCA and is the reason why one of the proposed solutions consists of adopting Wave Semiconductor's asynchronous Null Convention Logic (NCL).[7] It doesn't need a clock and manages the timing of logic propagation using encoded acknowledge signals. It's totally delay insensitive and thus helps solve the layout = timing issue. The consequence of NCA is then to reduce synthesis and physical design constraints and the possibility of avoiding the burden of using a real clock. The number and position of the logic gates are constrained much as they would be in standard digital circuits, so the same principles and automatic algorithms can be adopted.

References

1. M.T. Niemier et al., "Clocking Structures and Power Analysis for Nanomagnet-Based Logic Devices," *Proc. Int'l Symp. Low Power Electronics and Design,* ACM Press, 2007, pp. 26-31.
2. A. Imre et al., "Investigation of Shape-Dependent Switching of Coupled Nanomagnets," *Superlattices and Microstructures,* vol. 34, nos. 3-6, 2003, pp. 513-518.
3. M. Vacca, "Nanoarchitectures Based on Magnetic QCA," master's thesis, Electronics Department, Politecnico di Torino, 2008.
4. M. Graziano et al., "A NCL-HDL Snake-Clock Based Magnetic QCA Architecture," *IEEE Trans. Nanotechnology,* 2011, doi:10.1109/TNANO.2011.2118229.
5. C.S. Lent et al., "Quantum Cellular Automata," *Nanotechnology,* vol. 4, no. 1, 1993, pp. 49-57, doi:10.1088/0957-4484/4/1/004.
6. J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design—A Systems Perspective,* Kluwer Academic Publishers, 2001.
7. K.M. Fant and S.A. Brandt, "NULL Convention Logic™: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *Proc. Int'l Conf. Application Specific Systems, Architectures, and Processors,* IEEE CS Press, 1996, pp. 261-273.

the second in the reset, and the third in the hold state. The information moves along the circuit following the clock-zone sequence.

Figure 2c shows an example of a top view of a circuit wherein three signals are routed through three zones, and carry information to the MV as a basic QCA logic gate: ($MV = AB + AC + BC$). The information propagates in this example from left to right horizontally. As a vertical propagation should also be assured, the clock zones' organization is slightly more complex.[2,7] The snake clock zones allow information propagation in all directions, but they follow a snake-like sequence along the circuit plane.

Figure 2d depicts the magnetization transient of the MV output (the central element in the evidenced MV subblock). This magnetization transient is obtained by an accurate finite-difference nanomagnetic simulator.[11] The case reported in the simulation in Figure 2d corresponds to the input values shown by arrows in Figure 2c ($A = 1$, $B = 0$, 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; afterward it switches to a negative magnetization value (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 wires 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 one-third 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 the size of the wire delivering the clock. Typical sizes

of magnets are 50 to 100 nm with a space of 20 nm between two of them. It's easy, then, to see an opposite constraint: the more magnets in a zone, the easier the clock fabrication, at least as long as nanowires really can't be used for this purpose. The estimated realistic frequency reported is around 100 MHz.[5]

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

One possible solution in this situation is to use a real clock signal—that is, a further signal that delivers synchronization to blocks similar to D-flip-flops in CMOS circuits. However, the D-FF would delay the progress of data until a synchronization signal is sampled. Although it's natural to think of this solution as similar to CMOS structures, it's difficult to implement and unfeasible with current technology.

A better remedy for this situation is to design a circuit with information propagation split conceptually into two levels. The first level in such a design is strictly connected to the physical signal propagation, which must be synchronous. The other level is for logic signal propagation, which can rely on an asynchronous delay-insensitive logic. One of the possible asynchronous implementation choices, as we mentioned earlier, is NCL.[12]

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 vice versa. The advantage is that the switch of a gate in either direction 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 increased complexity. The consequences of this choice (of using NCL) are twofold. First, it's unnecessary 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. We've seen a few attempts at basic design flows using QCA, both for synthesis[13] and for placement.[14] Although a lot of work is yet to be done, especially with respect to any MQCA, results show that it's possible to rely on these design-flow algorithms.[13,14]

NCL is based on several basic cells: Fant and Brandt have detailed the cells' behavior,[12] whereas Vacca,[7] as well as Graziano et al.,[2] have explained the cells' application to an MQCA circuit. It's noteworthy that the two bits' encoding doubles the wires, and as a consequence, many crossing points could be necessary. Although this is a complication of NCL, it is readily resolved, as coplanar wire crossings have been experimentally demonstrated.[3,4]

At this point in our research, what we're most interested in clarifying is whether the NCL asynchronous circuit organization is an effective solution for QCA. We have previously compared Boolean and NCL using our VHDL behavioral model.[2,7] Logic gates are considered ideal, with no delay, but we simulated the wire propagation delay through the clock zones, using a register for every phase zone, with the corresponding zone's clock signal as a clock (see Figure 2a), thereby reproducing the pipelined circuit behavior. We based our model on the physical structure of every NCL gate, and on the basis of our snake clock.[2] We designed every gate to be feasible, and the VHDL description (proposed elsewhere)[2,7] reflected this design.

Over time we've improved this VHDL model, allowing for a hierarchical estimation of the circuit area and power dissipation. For the sake of simplicity, we won't describe the model's structure here in detail. We based the model on the real number of magnets used by the basic logic gates, and we've hierarchically and parametrically estimated the total number of nanomagnets to obtain a realistic approximation of the circuit area and power dissipation. We then calculated the power dissipated by the nanomagnets by multiplying their total number for the power dissipated (approximately) by each one.[5] Starting from the total number of magnets, we've also evaluated the circuit area, using the magnets' dimensions and some parameters to account for wasted space. Using the reckoned circuit area, and knowing the

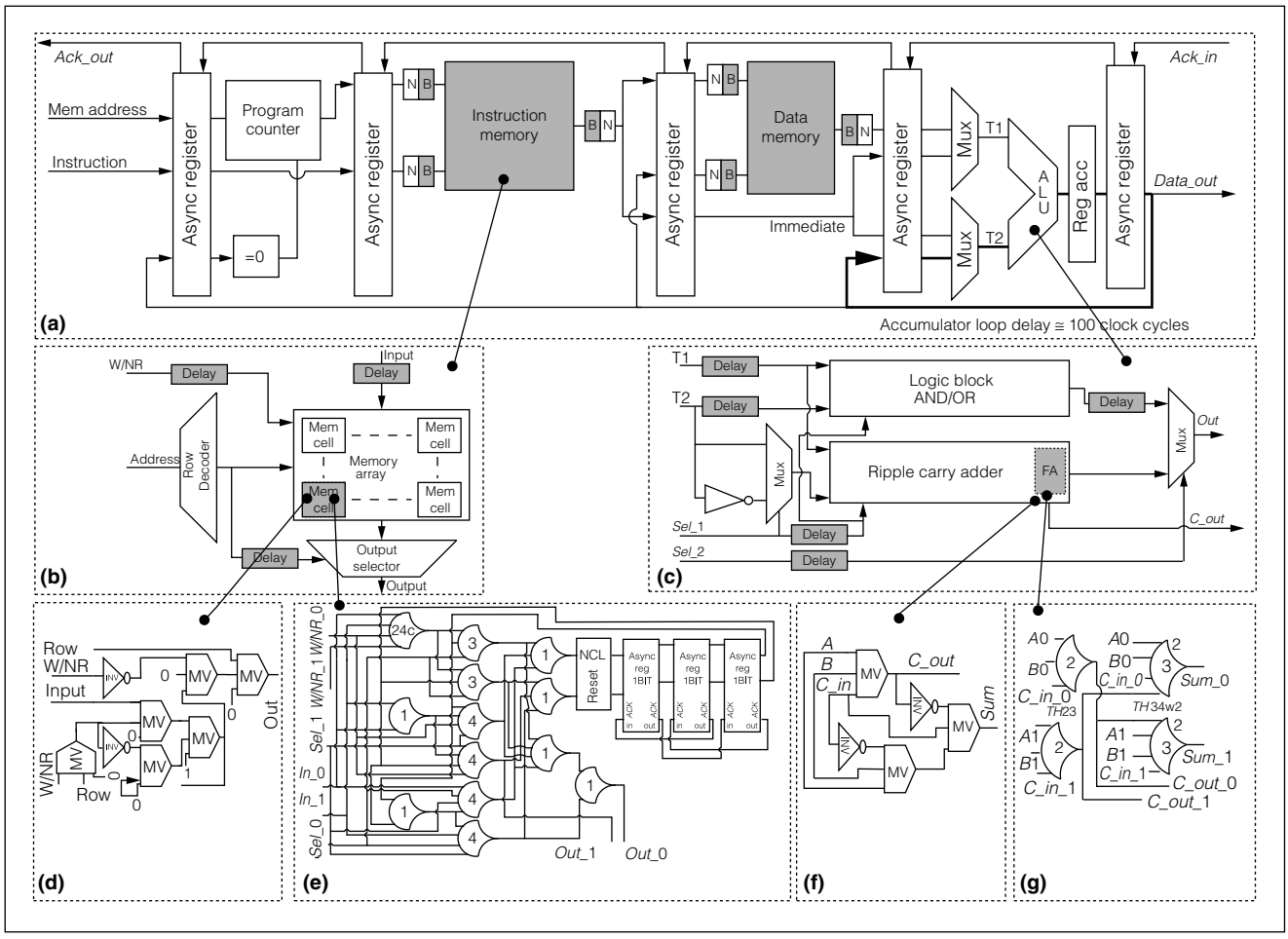

Figure 3. This figure gives a top-down view of the microprocessor structure. Top hierarchy architecture (a). Instruction memory architecture (b). The gray arithmetic logic unit (ALU) architecture delay blocks (c) are only present in the Boolean implementation. Memory cell structure based on majority voters (MVs) in a Boolean implementation (d). Memory cell structure based on Null Convention Logic (NCL) gates, themselves based on MVs (e).[2] Full adder (FA) structure based on the MV in a Boolean implementation; inputs are *A*, *B*, and *C_in* (or "carry in"), while outputs are *Sum* and *C_out* (or "carry out") (f). Full adder structure based on NCL gates, themselves based on the MV (g).[2] (B: Boolean, INV: inverter, N: NCL, W/NR: write/not read.)

clock zone's dimensions, we can estimate the clock wires' length and power dissipation caused by the joule effect. The power is estimated using the most efficient clock-generation system currently available.[15]

Synchronous versus asynchronous solutions

Now we can discuss the processor we designed and simulated using a simulator able to understand the VHDL model. In this work, the referenced architecture is a microprocessor, as Figure 3a shows. For a detailed comparison between fully synchronous and asynchronous solutions, we chose two of the main processor components: the instruction memory (Figure 3b) and the ALU (Figure 3c), both discussed herein.

Arithmetic logic unit

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 3c) 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 logic and arithmetic operations, while the input multiplexer selects, if needed, the negated operand for two complements' subtraction.

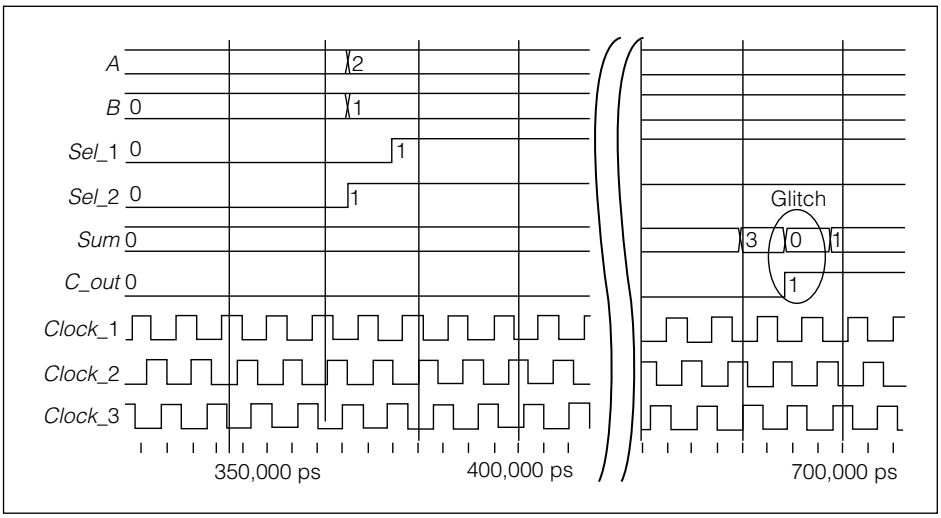

Figure 4. Boolean logic ALU simulation results: 2 + 1 and 2 − 1 operations are shown. Signal *Sel*_1 defines addition or subtraction. The overflow is listed here as *C_out*. Time values are shown in picoseconds (ps).

Data have been represented using 32 bits. Figures 4 and 5 show our simulation results for the Boolean and NCL versions.

As previously mentioned, the behavior is intrinsically pipelined, as Figure 4 shows. At every clock cycle, a data item is accepted at the inputs, and an output is correspondingly generated with high latency. The Boolean implementation of the ALU can be problematical because signals must be precisely synchronized to obtain working circuits, otherwise problems arise. In this simulation, we intentionally altered the delay of the synchronizing blocks to show how critical this synchronization problem is in the Boolean implementation. In the example, two logic operations are performed, first the addition of 2 + 1, and then the subtraction 2 − 1, according to the selection signal *Sel*_1. The circuit has a long latency because of its intrinsic structure, and results are available after a long delay (not shown in Figure 4). The first available output is correct (3, because of the addition). During the next clock cycle, the output correctly changes because of the pipeline, but the result shown, which is 0, is

Each full adder for the two logic cases is based on the structures in Figures 3f and 3g. The Boolean solution is implemented starting from the MV; the NCL circuit relies on NCL gates, internally based on the MV. Details on this structure are provided elsewhere.[2] It's enough to note here that double wires for each logic signal are present, and that, for example, the TH23 gate has the function OUT = $MV(B, C, \text{OUT}) + A$. The internal feedback is necessary to assure the delay insensitivity.

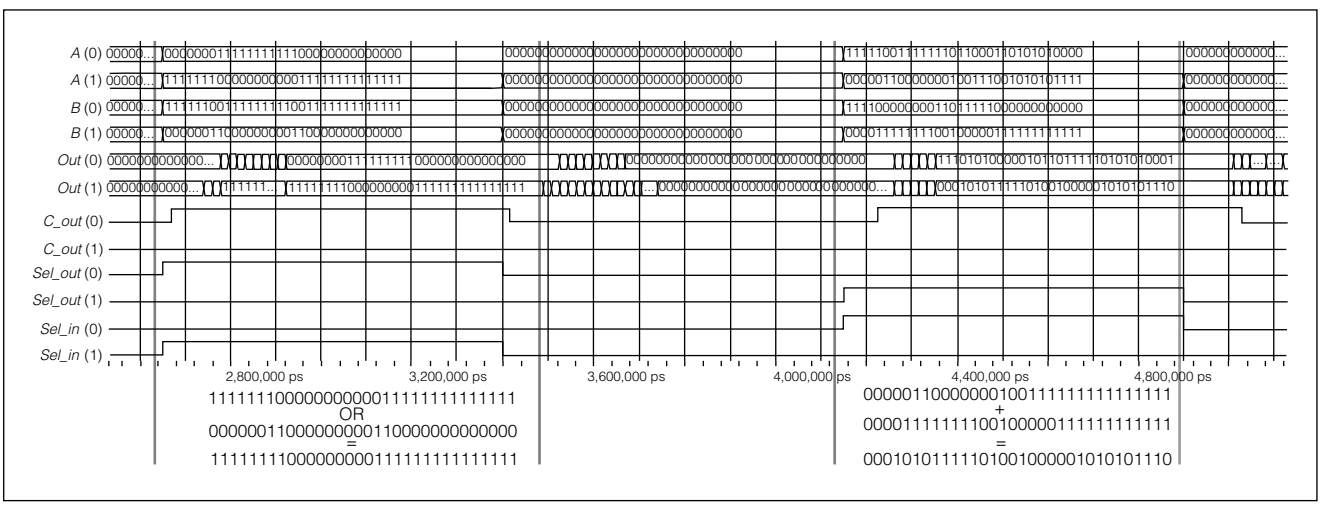

Figure 5. NCL ALU simulation results: an OR and an addition (+) are shown. The top waveforms are NCL encoded, while the bottom details are their Boolean translation. Waveforms *A* and *B* are the ALU inputs, waveforms labeled with *Out* and *C_out* are the ALU outputs "result" and "overflow" ("carry out") respectively.

wrong. At the next clock cycle, the output changes again, and now the results are correct (*Sum* is 1 and *C_out* is 0). 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, meaning a temporary variation to a wrong value (this is a demonstration of the layout = timing issue). It's noteworthy that this glitch has the same importance in this QCA circuit as it has in CMOS circuits when it's sampled by a flip-flop; it's an error that propagates throughout the circuit.

The use of NCL rather than Boolean totally eliminates this problem. Figure 5 shows an example where two operations use NCL encoding: a logic OR and an addition (+). Note that every signal is encoded using two bits and that signals periodically switch from Data to Null: a new data item is accepted only when all the inputs switch to the Null state independently from their delay. In this way, circuits work normally also in the presence of different propagation delays.

NCL, like every asynchronous logic, requires a communication protocol to operate. Figure 3a shows this, wherein every block of combinational logic is embraced by two asynchronous registers (a transmitter TX on the left and a receiver RX on the right of each block) that generate and exchange this handshake protocol:

- A data item 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 acknowledgment (*ack*) to the previous register (TX).
- When *Ack* is received at TX, a null (all outputs set to 0) is sent through the combinational circuit.
- RX receives the null and sends back another *Ack*.
- Once this second *Ack* is received, the TX register is ready to accept 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's point of view and accepts new data only after the completion of the Data-Null cycle. Note that the propagation time of the signals through the circuit and the propagation time of the *Ack* signal are equal to the combinational circuit's latency. This means that an asynchronous register accepts new data only after a time equal to four 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 1. Asynchronous versus synchronous design performance comparison.

Implementation	Area (μm^2)	Latency (no. of clock cycles)	Power (μW)	Power clock (μW)
ALU (Boolean)	1.33	34	0.52	0.86
ALU (NCL)	2.64	72	1.04	1.65
Memory (Boolean)	1.04	6	0.39	0.65
Memory (NCL)	44.38	26	36.60	27.66
Microprocessor	10.60	426	3.88	6.62

ALU: arithmetic logic unit; NCL: Null Convention Logic

Table 1 shows the comparison between the two ALUs in terms of area, latency, and power dissipation caused by the nanomagnets' switching. The power dissipation increases, 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, and the relative additional interconnections' overhead. The circuits' latency is also twice that of the Boolean circuit. Therefore, NCL solves the layout = timing issue, allowing a less-constrained design flow through standard design automation algorithms. This comes at the price of increasing the circuit's area and slowing down operations. We previously remarked that a Boolean QCA circuit accepts new data after each clock cycle. On the contrary, however, a QCA implemented with NCL accepts new data after a time that's equal to a multiple of the latency. The time itself is bigger than the latency of the Boolean version.

Parallel memory

We ran the same type of comparison—NCL versus Boolean—for a parallel memory (Figure 3b) organized as a 16 × 14 matrix (the microprocessor instruction memory). The structure was quite simple. A decoder selected the desired matrix row and the corresponding memory output. As with the ALU, we used delay blocks (colored in gray) only in the Boolean version to synchronize signals. Figures 3d and 3f show the details of a memory cell for the two logic types.

We omitted showing the waveforms for the sake of brevity, but Table 1 shows a comparison between the two implementations. Here the difference between the two logic choices is significant because of the complexity of the memory cell implemented in NCL, as Figure 3f shows. The memory's power consumption is roughly 100 times bigger than in the Boolean implementation, and the area is 44 times bigger. The power dissipation caused by the clock wires—estimated using the area, the zones' width, and the technological choices that Augustine et al. described for what concerns the magnetic field application[15]—are reported in Table 1's last column. The data we obtained are on the same order of the magnets' power consumption, and maintain in every case a trend similar to the one we obtained for the magnets.

The difference in terms of latency between Boolean logic and NCL isn't so big but still high (about six 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 eventually. However, these results show that NCL isn't suited for memory structures (at least when applied to QCA technology), as it could severely worsen results, even defeating the advantages of adopting this technology. In this situation, it's better to use a Boolean memory, provided that a good input signal synchronization is assured. In the case of a memory, it's easier to assure the absence of glitches in the Boolean version. This is because of the memory's (Boolean's or NCL's) 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 it's physically realized, this assumption cannot be proven.

Feedback in QCA circuits

Working with a complex circuit such as a microprocessor lets us 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-hand section of Figure 3a. 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. Because the circuit is intrinsically pipelined, it accepts new data at every clock cycle, but as Figure 3a shows, the feedback loop has a propagation delay of ≈ 100 clock cycles because of 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 that occurred 99 clock cycles before.

This propagation-delay problem is well-known, as it's typical of conditional jumps in reduced-instruction-set computer (RISC) microprocessors. However, in the case of QCA circuits, the feedback-delay is heavily amplified by the high pipeline stages and by every loop in the circuit. Only the adoption of an asynchronous logic such as NCL can solve it, because the computation is performed only when all the signals arrive at the circuit inputs. This means that a new ALU operation is executed only when the result of the previous operation has passed through the feedback loop and arrived at the ALU's inputs.

Microprocessor: Mixed-logic solution

On the basis of what we've discussed thus far, we 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—such as delay synchronization and feedback in sequential circuits—prevent its actual use. If we could overcome the delay synchronization in some cases with the development of an ad hoc algorithm to automatically synchronize signals (when possible), then we could solve the feedback issue using NCL, and accept a resultant loss in performance.

We thus propose a solution to achieve the best compromise between circuits' feasibility and performance optimization: a mixed Boolean-NCL logic design. We use the NCL asynchronous structure for the global architecture, leaving the Boolean synchronous solution for subblocks (for example, the memory), in which NCL would critically compromise results. Such an 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 3a) with four main components: a program counter, a parallel memory able to store 16 instructions of 14 bits, a data memory with four memory cells of 4 bits each, and the ALU. We organized the microprocessor in

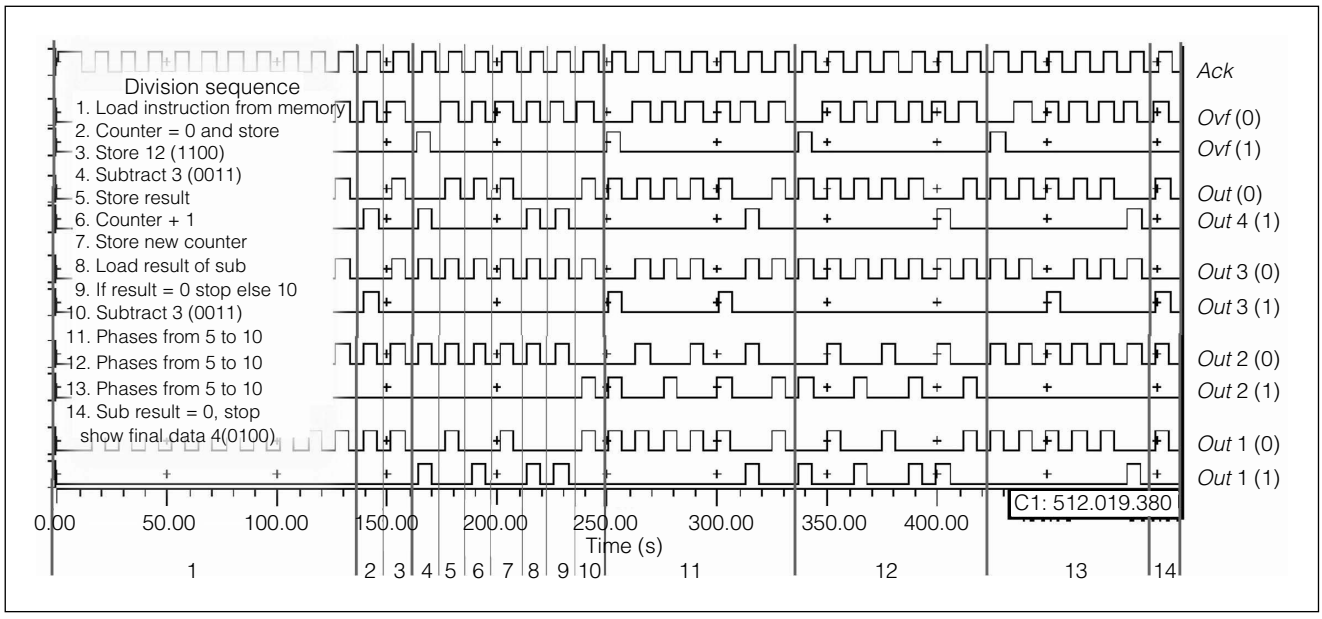

Figure 6. Division algorithm execution: 12/3 = 4. In the leftmost inset is the "Division sequence," which briefly explains the phases numbered at the bottom of the figure.

four 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, the two memories aren't NCL; rather, they're Boolean based. We substituted the memories because from our previous analysis they're the most critical and because in their Boolean implementation they're the simplest to synchronize.

A division algorithm helps us test the architecture: in this example, input $A = 12$ is divided by $B = 3$ (12/3). The result is reported in the example in Figure 6. The waveforms are organized in phases from 1 to 14 for the sake of clarity.

Table 1 shows the microprocessor's performance. The latency is high because of the design's complexity and lack of optimization, but it's interesting to compare its performance to the parallel NCL memory alone. The whole mixed-logic microprocessor is four times smaller and has 10 times less power dissipation than a memory NCL design alone. This shows that our approach works well because it lets us build every kind of QCA circuit without losing too much performance. Clearly, the memory must be carefully designed to synchronize delays. However, this isn't as complex for other blocks because of intrinsic regular organization in arrays of identical cells.

Finally, regarding power dissipation, we note that we evaluated and compared this QCA mixed solution to an equivalent CMOS solution. We implemented (using CMOS technology) a CMOS-based microprocessor with the same structure as our QCA processor and which operated at the same frequency of 100 MHz. We synthesized it on 45-nm standard cell technology and calculated its total power dissipation, which resulted in 536 μW. This proved that the QCA solution is advantageous: as Table 1 shows, it dissipates just $3.88 + 6.62 = 10.5$ μW in the mixed case. We estimated the power based on several constraints and parameters chosen to obtain a realistic evaluation.

THIS STUDY CLEARLY shows that our proposed mixed solution is viable. A totally synchronous architecture in QCA technology, although granting high data throughput, would be unfeasible, because it would require complex synchronization procedures to solve the layout = timing constraint and could be applied only to combinational circuits. An important improvement can be achieved if we were to adopt a global asynchronous circuit organization (based, for example, on NCL). In this way, no synchronization of signals is needed and both combinational and sequential circuits with any order of feedback can be implemented without the need of particular care while routing signals.

As with all "medications," collateral effects might arise, especially if the "dosage" isn't respected: circuits

are bigger, slower, and need more power (because of their increased complexity). If a poisonous effect is to be avoided, trade-offs must be carefully evaluated, and when block regularity reduces the burden of signal synchronization, synchronous blocks can coexist with asynchronous blocks.

This joint synchronous-asynchronous solution—Boolean logic and NCL—is a compromise between performance and circuit feasibility. It's also clear that QCA technology is best suited for pure combinational circuits, wherein 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 only at the cost of lowering the overall performance. However, our results show that even in this case it's advantageous.

Our future efforts will be directed toward finding a different solution—still asynchronous but not based on NCL—to explore whether it's possible to exploit the advantages of asynchrony without suffering the burdens that NCL implies in terms of area and latency. At the same time, our efforts will be directed toward an automatic circuit synthesizer, placer, and router for Boolean QCA logic circuits. ■

References

1. C.S. Lent et al., "Quantum Cellular Automata," *Nanotechnology,* vol. 4, no. 1, 1993, pp. 49-57, doi:10.1088/0957-4484/4/1/004.
2. M. Graziano et al., "A NCL-HDL Snake-Clock Based Magnetic QCA Architecture," *IEEE Trans. Nanotechnology,* 2011, doi:10.1109/TNANO.2011.2118229.
3. A. Imre et al., "Investigation of Shape-Dependent Switching of Coupled Nanomagnets," *Superlattices and Microstructures,* vol. 34, nos. 3-6, 2003, pp. 513-518.
4. J. Pulecio and S. Bhanja, "Magnetic Cellular Automata Coplanar Cross Wire Systems," *J. Applied Physics,* vol. 107, no. 3, 2010, pp. 034308–034308-5.
5. M.T. Niemier et al., "Clocking Structures and Power Analysis for Nanomagnet-Based Logic Devices," *Proc. Int'l Symp. Low Power Electronics and Design,* ACM Press, 2007, pp. 26-31.
6. E. Tabrizizadeh, H.R. Mohaqeq, and A. Vafaei, "Designing QCA Delay-Insensitive Serial Adder," *Proc. IEEE Int'l Conf. Emerging Trends in Engineering and Technology,* IEEE CS Press, 2008, pp. 447-452.
7. M. Vacca, "Nanoarchitectures Based on Magnetic QCA," master's thesis, Electronics Department, Politecnico di Torino, 2008.
8. M.R. Casu and L. Macchiarulo, "Adaptive Latency-Insensitive Protocols," *IEEE Design & Test of Computers,* vol. 24, no. 5, 2007, pp. 442-452.
9. M. Martina and G. Masera, "Turbo NOC: A Framework for the Design of Network-on-Chip-Based Turbo Decoder Architectures," *IEEE Trans. Circuits and Systems I,* vol. 57, no. 10, 2010, pp. 2776-2789.
10. J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design—A Systems Perspective,* Kluwer Academic Publishers, 2001.
11. T. Fischbacher et al., "A Systematic Approach to Multiphysics Extensions of Finite-Element-Based Micromagnetic Simulations: Nmag," *IEEE Trans. Magnetics,* vol. 43, no. 6, 2007, pp. 2896-2898.
12. K.M. Fant and S.A. Brandt, "NULL Convention Logic™: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *Proc. Int'l Conf. Application Specific Systems, Architectures, and Processors,* IEEE CS Press, 1996, pp. 261-273.
13. R. Zhang, P. Gupta, P. and N.K. Jha, "Synthesis of Majority and Minority Networks and Its Applications to QCA, TPL and SET Based Nanotechnologies," *Proc. IEEE Int'l Conf. VLSI Design,* IEEE CS Press, 2005, pp. 229-234.
14. J.C. Wook, B. Smith, and S.K. Lim, "QCA Physical Design with Crossing Minimization," *Proc. 5th IEEE Conf. Nanotechnology,* IEEE Press, 2005, pp. 108-111.
15. C. Augustine et al., "Ultra-Low Power Nano-Magnet Based Computing: A System-Level Perspective," *IEEE Trans. Nanotechnology,* 2010, doi:10.1109/TNANO.2010.2079941.

Mariagrazia Graziano is a researcher and assistant professor at the Politecnico di Torino and an adjunct faculty member at the University of Illinois, Chicago. Her research interests include models and algorithms for the design of CMOS high-speed and low-noise digital circuits, as well as "beyond CMOS" devices, circuits, and architectures. Graziano has a PhD in electronics engineering from the Politecnico di Torino. She's a member of IEEE.

Marco Vacca is a doctoral student in electronic and communications engineering at the Politecnico

di Torino, where he also teaches courses on power electronics and the design of digital circuits. His research interests include quantum-dot cellular automata and other ''beyond CMOS'' technologies. Vacca has a DrEng in electronics engineering from the Politecnico di Torino.

Davide Blua is a master's student in electronic and computer science engineering at the Politecnico di Torino. His interests are in quantum-dot cellular automata, CAD tools for nanostructures, and embedded systems. Blua has an MS in electrical engineering from the University of Illinois, Chicago.

Maurizio Zamboni is a professor in the Electronics Department at the Politecnico di Torino. His research interests include new low-power circuits and innovative technologies beyond the CMOS world. Zamboni has a PhD in electronics engineering from the Politecnico di Torino.

■ Direct questions and comments about this article to Mariagrazia Graziano, corso Duca degli Abruzzi 24, I10129, Torino, IT; mariagrazia.graziano@polito.it.