An Efficient Quantum-Dot Cellular Automata Adder

Francesco Bruschi Francesco Perini Vincenzo Rana Donatella Sciuto
Dipartimento di Elettronica e Informazione
Politecnico di Milano

***Abstract*—This paper presents a ripple-carry adder module that can serve as a basic component for Quantum Dot Automata arithmetic circuits. The main methodological design innovation over existing state of the art solutions was the adoption of so called *minority* gates in addition to the more traditional *majority* voters. Exploiting this widened basic block set, we obtained a more compact, and thus less expensive circuit. Moreover, the layout was designed in order to comply with the rules for robustness again noise paths [6].**

I. Introduction

As the electronic CMOS transistor technology for information processing approaches its limits, new possibilities for the implementation of digital information processing are being explored. Among those raising greatest interest are Quantum Cellular Automaton (QCA). QCA are matrixes of cells, in which information is stored as the position of couples of electrons bound within cell borders. Neighbor cells interact by means of electric Coulombian repulsion, and cell state can be frozen at will by controlling a potential barrier rising signal (clock). Disposing such cells in a two dimensional matrix, driving the state of some cells (inputs) to a desired value, triggering the freezing of the cells state with a multiphase signal, and then reading out the value of some defined cells (outputs), it is possibile to implement digital information processing functionalities. Even though there is agreement on the mathematical model that describes certain fundamental functional features of QCA cells, there are different technologies still competing for their actual implementation. All the technologies offer the perspective of highly scalable nanoscale devices with power consumptions far lower than nowadays microelectronics, even though some seem more likely to be exploitable on an industrial scale. Even though the technological roadmap to industrial production of such devices is still under definition, the problem of circuit design can be (and indeed already is) addressed, relying on the simulation of cell models for verification. For what concerns design techniques, the situation is up to now similar to that of CMOS gates based circuit: a set of basic components is defined to be combined for implementing the desired functionalities. Since it is possible, with QCA cells, to implement universal sets of classical logic gates (NAND, NOR), all the synthesis methods based on such elementary cells apply. Nevertheless, to ground QCA design on such "compatibility" is largely inefficient since, differently from what happens with CMOS transistor, they are not the simplest and most economic components possible. Rather, the role of *basic blocks*, that is of the most efficient basic functionalities implementable, is played in QCA circuits by the *majority* and the *not* gates (majority gates have three inputs, and produce as output the boolean value that is most present at the input ports). One of the possibilities is to then try to define and implement synthesis algorithms that generate functions circuital implementations combining instances of these two basic gates. At the same time, analogously to what happens for CMOS design, it is worth investigating the design of components that implement specific, highly reusable functions, such as modular components for arithmetic circuits. This paper contributes to the latter effort, presenting QCA designs of half and full adders with significantly lower cost than the state of the art implementations. To reach this goal we widen the set of basic gate exploit the *minority* gate for the first time in an arithmetic circuit design. To validate the proposed components, we build adders of different orders and simulate them with state of the art simulation engines and models. The paper is structured as follows: in Section II we summarize the fundamental facts regarding QCA cells. In Section III we review the state of the art of arithmetic design with QCAs. In section IV we introduce the basic adder module and the challenges arisen in the design. In Section V the issues arising from the combination of the basic modules into higher order adders are described, and the solutions discussed. Functional and timing correctness are then verified by simulation. In Section VI we compare timing and cost figures of our solution with the state of the art. Conclusions and perspectives for further works are discussed in Section VII.

978-3-9810801-7-9/DATE11/©2011 EDAA

II. QCA Basics

Quantum Cellular Automaton can be seen as matrixes of cells. Each cell can be empty or it can contain four *quantum dots*. A quantum dot can be described as a well in which an electron can be trapped or, equivalently, stay at rest. The dots can be positioned one near each vertex of the cell, or one close to the medium point of each side, to form, respectively, *cross cells* and *plus cells*. Within each cell, two excess electrons are bound. Each electron can be in one of the four dots, but due to electric repulsion no two electrons can occupy the same dot. Moreover, electrons will tend to occupy dots that are maximally distant, that is *opposed* dots.

In all the QCA technologies, electrons can move from one dot to another by means of quantum tunneling. The cells can then be considered bistables that, at equilibrium, are in one of two different states. The electrons in one cell interact with the electrons of the neighbor cells by means of Coulombian repulsion. Each cell is controlled by a signal, that can either

freeze the electrons in the current position by rising a potential barrier, or, by lowering it, it can allow the electrons to tunnel between dots and arrange according to the interaction with the other cells. By appropriately driving the potential barrier of the cells, with a multiphase periodic signal, it is possible to define a *flow* in the cells sensibility to neighbor cells. This is necessary in order to make the circuit predictably responsive to inputs, as will be shown later on in this section.

A. Digital design with QCA

There are different approaches to digital design with QCAs. Some are based on standard synthesis procedures, that take as input the truth table of the functionality to implement and produce as output a disposition of cells in a matrix that implement it [8]. Such procedures are based, as for the CMOS technologies, on the definition of a universal set of "*gates*". The simplest logical operator that can be realised with QCAs is the so called *majority gate*. The majority gate has three boolean inputs, and produces as output the value that is most present as input. A boolean expression representing it is: $M(x, y, z) = xy + xz + yz$. The majority gate, in conjunction with the negation function is universal. Another elementary gate with QCA is the minority gate, which is the negation of the majority: $m(x, y, z) = \neg M(x, y, z)$. It can be shown that m is universal ([9]). In Figure 1 (A) the implementation of a majority gate is shown. When polarization of cells A, B and C are fixed, their state will propagate to the nearest cells. Then, the cell at the center of the cross will be subject to the Coloumbian interaction with the three polarized neighbours, and its electrons will settle according to the most present polarization value. The state will then be propagated to the output cell. On the other hand, the implementation of the

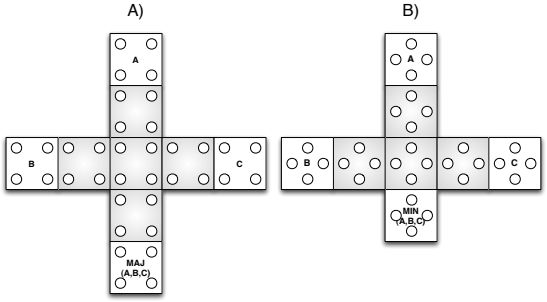

Fig. 1. Structure of a Majority gate (A) and a Minority gate (B)

minority gate, which behaviour is similar to the one of the majority gate, is shown in Figure 1 (B).

B. Wires and clocks

In QCA circuits the state of a cell can be transmitted, (that is, propagated spatially) by means of *wires*.

A wire simply is an array of QCA cells in which the polarization is propagated from one "input" cell to the others. To achieve directional control on the information flow, QCA circuits rely on a clocking mechanism that consists on four periodic signals with equal frequencies, each shifted in phase by 90 degrees. One of the clock signals can be considered the reference ($phase = 0$) and the others are delayed one, two and three quarters of a period. The polarization of a cell which is assigned to clock n doesn't affect the polarization of its neighbours assigned to clock $n - 1$.

It is important to point out that cross cells propagate the same polarization along the wire while, in contrast, plus cells intrinsically generate an inverted polarization in neighboring cells of the same shape, thus achieving a negation logic function.

III. State of the Art in QCA Arithmetic Design

Even though actual implementation of practical QCA devices is still a perspective goal, the behavioral features of the cells are for the most part precisely defined. This allows to build mathematical models of the devices and to simulate them, making it possible to address a wide number of design challenges and issues, ahead of industrial implementations. The de facto standard for the simulation of QCA devices is, to date, QCADesigner [12]. In [6] it is shown how, with QCA, large designs generated by the interconnection of simpler, functionally verified blocks may behave incorrectly due to very peculiar physical effects arising from Coloumbian interaction of neighbor cells. From their analysis, authors drew a set of design best practices aimed at circuit robustness with respect to such effects. Such practices have the form of constraints on the circuit geometric features such as the minimum wire length of a phase block, the minimum space between wires and the maximum length of wires. In particular, the inherent pipelined structure of QCA circuits is taken into account. In [8] an algorithm for the automatic synthesis of booleans functions, using majority and minority gates, is presented. Unfortunately, among the benchmark used to test it, there aren't arithmetics circuits. In [11] a modular ripple-carry structure for adders is presented. The design is not optimized nor robust. Other adder structures were presented in [2][3][4], and a comparison of the different designs can be found in [7]. In [5], with which our work directly compares, the most efficient modular adder design to date is presented. Authors start from a conventional ripple carry design, and optimize the layout for QCA implementation. The paths for the carry propagations are minimized, so that minimum possible delay (in clock cycles) is achieved, and the area is significantly reduced. With respect to [5], we exploited minority gates in addition to majority ones. This, together with careful optimization, allowed us to obtain an even more compact adder, maintaining the same temporal performance. Moreover, with respect to [5], the adder here presented complies with the constraints drawn in [6] for robust design. In [9], a minority based full-adder is presented, but it is not modular and is only suitable as a stand-alone component, since its inputs and outputs are enclosed by logic wires, making it impossible to interconnect it to other modules without long multilayer crossovers. What is more, the performance optimization issue is not considered at all: the circuit spawns across nine clock zones for both Sum and

Carry_out results. In contrast, our design fits into just 1 clock zone for the Carry_out plus two more for the Sum.

IV. A COMPACT ADDER MODULE

In this Section we propose the design of a compact full-adder module to be implemented as a QCA. We followed a logic simplification process similar to the one presented in [5], in which the most efficient adder implementation to date it is described. We started adding the minority gate to the toolset of basic blocks, and sought for the simplest implementation of full adder outputs.

The expressions obtained are:

$$CarryOut = \neg m(A, B, CarryIn) \tag{1}$$

$$Sum = \neg m(m(\neg A, \neg B, CarryIn), m(A, B, CarryIn), CarryIn) \tag{2}$$

The proposed adder module, implemented with minority gates only, is shown in Figure 2.

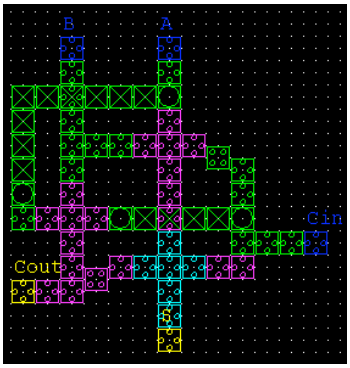

Fig. 2. Schema of the ripple-carry adder used as basic block

The advantage of using minority rather than majority gates resides in the opportunity of saving the area and the cells devoted to *not* gates, at the cost of a slightly increased design complexity. The saving comes from the fact that minority gates are natively implemented by plus shaped cells that, along a wire, propagate a complemented signal to their neighbours. This allows to save area and cells although maintaining the same end to end delay, with respect to [5], for both Sum and CarryOut signals. In addition, the proposed full-adder complies with the best practices for robust design derived in [6] in terms of both maximum wire length and minimum adjacent cells in a clock region, while the adder proposed in [5] does not, since it has a single-cell clock zone. This aspect will turn out to be of particular significance in the implementation of 2 and 4 bit adders.

The increased design difficulty with plus shaped cells comes from the need to obtain a correct number of value inversions along the wires and, at the same time, a reduced maximum wire length allowed. The design proposed is extremely compact and, notwithstanding the aforementioned difficulties, has been proved stable and effective through thorough simulation.

There are two main simulation techniques to test a design [12]: Coherence Vector and Bistable. The Coherence Vector simulation engine computes evolution of a cell state relying on the kink energy between the considered cell and all the other cells in the circuit. The accuracy of Coherence Vector model depends on the granularity of the time step and can be used to evaluate the dynamic behavior of cells polarization switching [6]. On the other hand, Bistable Approximation models are still based on kink-energy but don't analyse the transient between two equilibrium states, thus reducing the total time of simulation but resulting less accurate.

We tested our designs with both simulation engines: in Bistable Approximation the number of samples was set to $2000 * 2^n$ where n is the number of inputs, as recommended by QCADesigner guidelines. As shown in Fig. 3, we obtained a correct and neat output waveform, not affected by any of the distortions that in many QCA logic components emerge, as described in [6]. This makes it possible to use the proposed adder module as a stable basic block for the design of 2 and 4 bits adders, as is shown in the following sections. Unfortunately, the work proposed in [5] does not provide any waveform, making it impossible to perform a robustness comparison between the two designs.

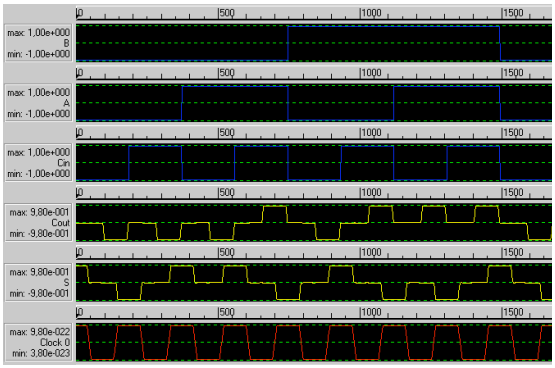

Fig. 3. Simulation results of the ripple-carry

V. HIGHER ORDER ADDING CIRCUITS

As previously hinted, the full adder design described in the previous section can be used as a basic building block for the design of higher order adding circuits. It is worth noting that, differently from what happens in CMOS design, it is not possible to simply connect two single adders in order to obtain a 2-bit adder. In fact, in QCA designs, it is necessary to both extend and add clock regions on the input and output wires in order to synchronize them, thus increasing the total amount of cells and the total area usage. As a side effect, these added cells may create interferences and potentially lead to inconsistencies in the output signals, if not correctly handled. It is then necessary to shift the clock regions to save the consistency of the results. To overcome the impact of this adjustments on the circuit cost, we introduced specific optimisations that achieved a neat output waveform though introducing only one clock zone more than the single full adder. In other words, the most significant carry will be ready after 2/4 of a clock period after the beginning of the computation, while both the sum signals are ready in a single clock cycle. In fact, increasing of

one bit the capacity of the adder leads to an increment of its latency of only 1/4 of clock cycle, both for the *CarryOut* and the for the *Sum* signals.

We now focus on the 4 bit adder, which, for its design structure and its latencies, can be connected modularly without requiring any clock regions re-phase, and it is then very suitable for the implementation of higher order adders such as 8, 16, 32 and 64 bit adders. As shown in Figure 4, the initial *CarryIn* and the *CarryOut* are in the same clock zone. Hence, this module can be just replicated and connected to a module of the same kind *as is*, with no need for internal clock re-assignmenet or re-synchronization. An issue would arise in adders large enough to require wires for input skewing and output deskewing so long that would exceed the maximum wire lenght allowed. But for adders of such high order, a carry-look-ahead approach would be more convenient, as shown in Section VI.

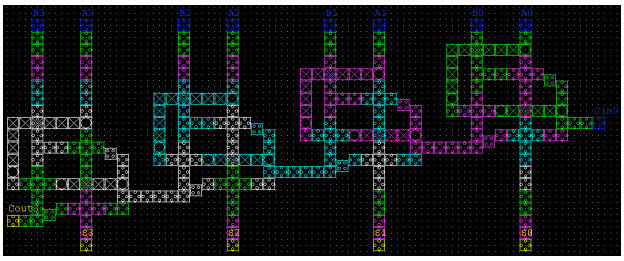

Fig. 4. Schema of the 4-bit ripple-carry adder

VI. Performance figures

Aim of the work presented in this paper is the optimization of the basic ripple-carry adder module by minimizing the number of utilized cells. For what concerns the timing performance, our objective has been to preserve the same latency of the best adder module that can be found in literature, both for the *CarryOut* and the *Sum* signals. In particular, the proposed approach makes it possible, considering the simple full-adder, to save around the 17% of the cells (70 cells against 84 cells) with respect to the best approach that can be found in literature. Considering the timing performance, since every Majority gate introduces the same delays of a Minority gate (a 1/4 clock cycle delay), the *CarryOut* signal is ready with only 1/4 clock delay, while the *Sum* signal is valid after 3/4 of a clock cycle, that are the same timing performance of the best approach of the state of the art. Table I presents a complete comparison among the proposed adder module and the best carry-look-ahead and ripple-carry adder modules that can be found in literature [5] with respect to several different higher order adders (the reduction of the number of cells of the proposed 4 bit ripple-carry adder, RCA4, is around 16% with respect to CFA4, while the reduction of the number of cells of the proposed 8 bit ripple-carry adder, RCA8, is around 18% with respect to CFA8).

TABLE I
Comparison among different implementations of adder circuits

Name	Complexity (cells)	Area (μm)	Delay (clocks)
CLA4	1575	0.74x1.09	$3\frac{2}{4}$
CLA8	3988	3.50x1.58	$6\frac{2}{4}$
CFA4	371	0.90x0.45	$1\frac{2}{4}$
CFA8	789	1.79x0.53	$2\frac{2}{4}$
proposed RCA4	313	1.00x0.40	$1\frac{2}{4}$
proposed RCA8	721	1.97x0.56	$2\frac{2}{4}$

VII. Conclusions and further works

The work proposed in this paper has shown that it is possible to significantly reduce the number of cells required to design basic components, such as adder circuits, by utilizing minority gates instead of majority gates. In addition to this, the proposed design presents an increased robustness (also shown by the results of the different simulations) with respect to previous approaches that can be found in literature, since almost all the best practices have been followed. The main drawback of the proposed approach could be found in the increased complexity of the design, with respect to both the length of the wires and the setup of the clock regions. These issues has been successfully faced by means of ad hoc optimizations performed by hands, and the latencies of the resulting adder components have been shown to be equal to the best adder components that can be found in literature, while the reduction of the number of cells ranges from 16% to 18%.

An interesting extension of the proposed work could be the automation of the optimizations proposed in this paper, in order to make it possible to synthesize also more complex circuits. This would require the automatic synthesis of a logic function with only minority and not gates, in addition to the correct handling of wires length and clock regions.

References

[1] Antonelli Dominic A. et al. Quantum-dot cellular automata (qca) circuit partitioning: problem modeling and solutions. In *DAC '04*, pages 363–368, New York, NY, USA, 2004. ACM.

[2] H. Cho et al. Pipelined carry lookahead adder design in quantum-dot cellular automata. In *Proc. Conf. Record of the 39th Asilomar Conf. Signals, Systems, and Computers*, pages 1191–1195, 2005.

[3] H. Cho et al. Modular design of conditional sum adders using quantum-dot cellular automata. In *6th IEEE Conf. Nanotechnology*, July 2006.

[4] H. Cho et al. Adder designs and analyses for quantum-dot cellular automata. In *IEEE Trans. Nanotechnology*, volume 6, pages 374–383, May 2007.

[5] Heumpil Cho et al. Adder and multiplier design in quantum-dot cellular automata. *IEEE Transactions on Computers*, 58(6):721–727, 2009.

[6] Kim Kyosun et al. Towards designing robust qca architectures in the presence of sneak noise paths. In *DATE '05*, pages 1214–1219, Washington, DC, USA, 2005. IEEE Computer Society.

[7] R. Zhang et al. Performance comparison of quantum-dot cellular automata adders. In *Proc. IEEE International Symposium on Circuits and Systems*, volume 3, pages 2522–2526, 2005.

[8] Rui Zhang et al. Majority and minority network synthesis with application to qca-, set-, and tpl-based nanotechnologies. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 26(7):1233–1245, 2007.

[9] Samir Roy et al. Minority gate oriented logic design with quantum-dot cellular automata. In *ACRI*, pages 646–656, 2006.

[10] Vankamamidi V. et al. Tile-based design of a serial memory in qca. In *GLSVSLI '05*, pages 201–206, New York, NY, USA, 2005. ACM.

[11] Walus K. et al. Computer arithmetic structures for quantum cellular automata. In *Signals, Systems and Computers, 2003. 37th Asilomar Conference on*, volume 2, pages 1435–1439 Vol.2, Nov. 2003.

[12] QCADesigner Website: http://www.mina.ubc.ca/qcadesigner.