A Quantum-dot Cellular Automata Parallel Prefix Adder

Kim Aragon Escobar[1], Renato Perez Ribas[2]

Institute of Informatics, Federal University of Rio Grande do Sul

Avenue Bento Gonçalves, 9500, Porto Alegre, RS, Brazil

[1]kaescobar@inf.ufrgs.br

[2]rpribas@inf.ufrgs.br

***Abstract* — Quantum-dot cellular automata (QCA) is an emerging technology which has been claimed to be faster and smaller than the most traditional CMOS technology. Since its beginning, many advances have been made, as the introduction of basic devices and the insertion of more complex circuits, like adders. It is not straightforward to implement more complex adders using QCA. This paper proposes the logical blocks to implement a parallel-prefix adder based on QCA technology. Furthermore, some technical issues are discussed through simulation results.**

***Keywords*— Quantum-dot cellular automata, QCA, parallel prefix adder, PPA, digital circuit, logic gates.**

I. INTRODUCTION

In the last decades, the scaling down of the transistors has improved the integration capability of IC. However, such scaling is reaching the physical limitation [1]. There are several good candidates to replace the current standard CMOS technology [2]. Many studies inspect the computation using cellular automata architecture and quantum confinement for that purpose [3].

Quantum-dot cellular automata (QCA) is one of the technologies that have been studied in the last few years to be used in future computation; being among the technologies that is capable to substitute the CMOS. One of the features of QCA is that this technology transmits the information without electrical current transmission, i.e., the information is transmitted based on the Coulomb interaction between QCA cells and the quantum confinement of electrons in each cell. QCA is based on a binary logic because each cell carries one of two logical values. This prominent technology has high device density, extremely low power consumption and very high switching speed. The basic logic devices used in QCA circuits are the wire, the majority voter (also known as majority gate) and the inverter [4]. Using these basic logic gates, it is possible to implement any complex gates like memories [5], flip-flops [6] or adders [7].

Adder circuits are very import in order to implement a microprocessor. There are different architectures of an adder circuit like ripple carry adder (RCA), carry select adder (CSelA), carry skip adder (CSkipA), carry lookahead adder (CLA) and parallel-prefix adder (PPA) [8]. This paper focuses on PPA design. PPA is an algorithm with three steps, based on the principle of propagation and generation signals [8]. The second step is the main differential of PPA because it is possible to model the architecture in order to optimize the speed or the area, being also possible to implement hybrid solution [9]. There are some well-known PPA architectures, like Brent-Kung [10] and Kogge-Stone [11], each one of this focus on one constraint (speed and area), whereas Ladner-Fischer [14] and Hans-Carlson [15] proposed hybrid implementations.

Several adder architectures implemented using the QCA technology have been proposed in the literature, like RCA [4][7], CSelA [7] and CLA [7]. This work presents an implementation of each block used in the PPA circuit, discussing the implementation of each step of the algorithm. Moreover, each logic gate is analyzed separately in order to evaluate the individual timing, which is quite straightforward in the QCA technology because of the gate clocked nature [12]. Furthermore, possible noise paths that could jeopardize the adder are discussed [13].

Section II gives a clear understanding about QCA technology. Section III explains the algorithm of PPA and shows the implementation of this adder using QCA technology. In Section IV, the analysis of each gate implemented is presented. The conclusions and future works are outlined in Section V.

II. QUANTUM CELLULAR AUTOMATA

The following explanations are qualitative, having in mind the computer architecture. Fig. 1 depicts the basic QCA cell, which is composed by 4 dots, one in each corner. QCA has two electrons confined in one of the 4 corner dots. These two electrons are allowed to tunnel between these dots. Nevertheless, tunneling to out of the cell is assumed to be impossible [3]. The configuration with the lowest energy sum takes place when the electrons have the largest possible distance between each other happens in the opposite corners because of the Coulomb forces between the electrons. The two possible states are the ones depicted in Fig. 1. Each one of the states has a polarization number as P = 1 and P = -1.

The basic devices in QCA architecture are the wire (shown in Fig. 2), the majority gate (shown in Fig. 4) and the inverter (shown in Fig. 3). The inverter and wire have the same proposal as the conventional circuits in CMOS. The majority gate is a well-known gate in the fault tolerance area as a majority voter. The majority gate Boolean equation is

AB+BC+AC. Using this equation, it is possible to deduce the following: if C = 0, so the majority gate represents the AND between the inputs A and B; if C = 1, it is possible to eliminate the AB minterm, and the majority gate represents the OR between the inputs A and B. It is known through De Morgan theorem that if we have available inverters and only AND gates or OR gates, it is possible to build any Boolean function.

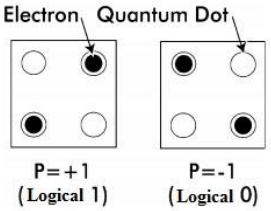

Fig. 1 –Binary information codification in QCA [17].

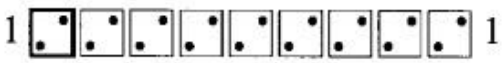

Fig. 2 – Wire structure [12].

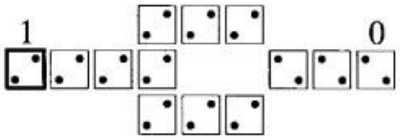

Fig. 3 – QCA inverter gate [12].

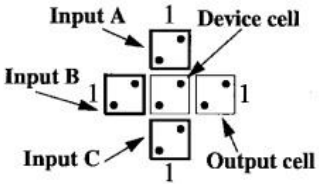

Fig. 4 – Majority gate topology [12].

Basically, it is possible to build any circuit in QCA but, due to its duplex nature, it is necessary some design strategies to ensure that the signal does not go back to the input. In [12], the solution proposed in order to solve the problem is the introduction of the circuit temporal barrier, called clock. The four possible states are described below, and depicted in Fig. 5 [12]:

1) Switch: When the clock is going from high to low or the barrier is rising, the cell becomes polarized with the polarization of its drivers.
2) Hold: When the clock is low or the barrier is raised, the cell retains its state.
3) Release: When the clock is going from low to high or the barrier is lowering, the cell is allowed to relax, losing its polarization.
4) Relax: When the clock is high or the barrier is low, the cell remains not polarized.

III. Parallel Prefix Adder

Parallel-prefix adder (PPA) algorithm is based on the principle of *generate* (G) and *propagate* (P) signals. The principle of P and G is to predict if during a sum of 2 vectors, A and B, occurs carry-out (Cout). P signal predicts if the sum between two bits a_i and b_i propagates the carry arriving in this index to the next one. It can be done by checking if a_i or b_i presents the logic value '1' (OR or exclusive-OR operations). G signal, in turn, predicts if the sum between two bits a_i and b_i generates a carry signal, without considering the carry arriving in this index (cin_i). It is possible by just verifying if both a_i and b_i presents the logic value '1' (AND operation). PPA uses initially *propagate* and *generate* signals individually for each bit index. The individual signals area used in order to calculate the array propagation, $P_{i,0}$, and array generation, $G_{i,0}$. With the array propagation and generation for $0 < i \leq n$, where 'n' is the number of bits, it is possible to calculate the Cout for each bit. Finally, using all the individual propagation and individual Couts, it is possible to evaluate the final sum for each bit.

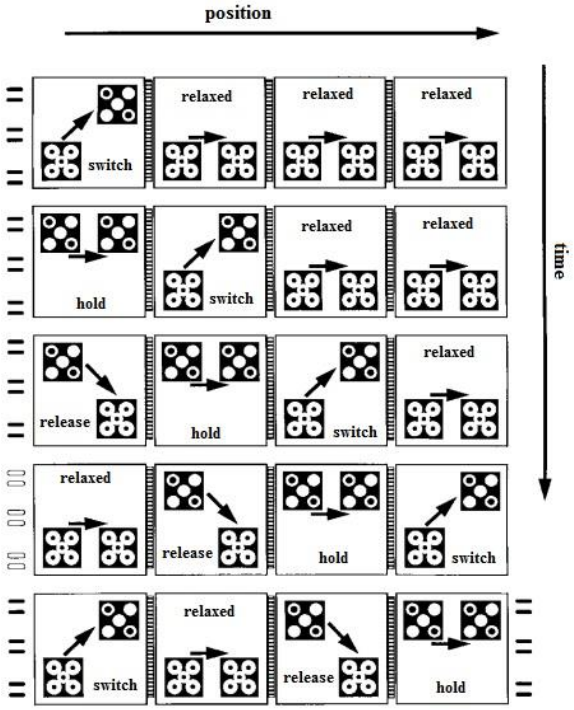

Fig. 5 –Four possible clock states [12].

Bellow, each step is enumerate and desbribed using Boolean expressions:

1) Calculate P_i and G_i: $P_i = a_i \oplus b_i$ and $G_i = a_i \bullet b_i$
2) Calculate the group generation and the group propagation using the following expressions: $G_{i+1,i} = G_{i+1} + P_{i+1} \bullet G_i$ and $P_{i+1,i} = P_{i+1} \bullet P_i$
3) Calculate the Couts for each bit: $Cout_{i-1} = G_{i,0} + P_{i,0} \bullet C_{in}$
4) Calculate the sum for each bit $S_i = P_i \oplus Cout_{i-1}$

IV. QCA PPA Circuit Implementation

In the first step, in order to implement the PPA, is defined the expressions using devices defined in Section II. In order to keep it simple, the majority gate (with A, B and C as input) is denoted by M(A,B,C), and the symbol ~A represents the input A negated. Analyzing the 4 steps, it is possible to notice that we have some repeated expressions. The basic gates used are AND, OR, XOR and F=A+B•C.

AND and OR expressions are translated as M(A,B,0) and M(A,B,1), respectively. Exclusive-OR (XOR) expression can be achieved based on its definition XOR=A•(~B) + (~A)•B, using majority gates. The expressions are M(M(A,~B,0), M(~A,B,0),1). F expression looks like M(A,M(B,C,0),1).

Now, with the majority expressions available, it is possible to show the layout for each one. AND and OR expressions are very simple. In Fig. 6, it is shown just the AND gate, whereas to have the OR gate one can just replace the -1 by 1.

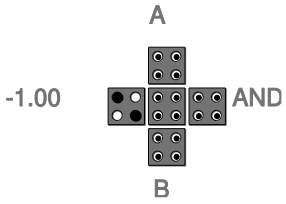

Fig. 6 – QCA AND gate implementation using a single majority gate.

XOR can be implemented using a cascade association of majority gates, as show in Fig. 7. The cells within the squares are the majority gates and the ones within the circles are inverters.

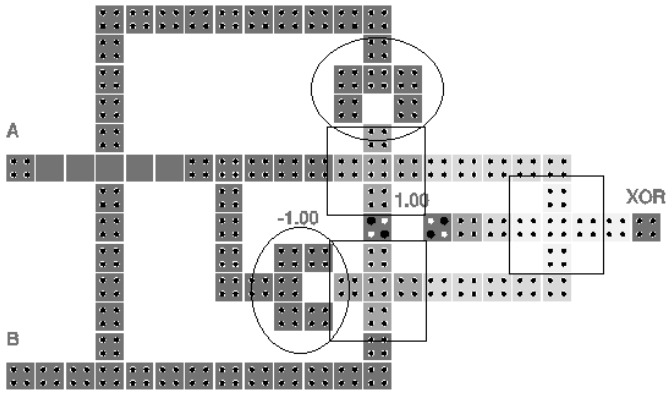

Fig. 7 – QCA XOR gate implementation.

The expression F can be implemented using a cascade association of majority gates as well as a complex gate, as shown in Fig. 8. The complex gate is indeed the association of majority gates. The difference between the cascade and the complex gate are the clock zone. In Fig. 9, it is possible to see the F expression implemented with cascade majority. Moreover, it is possible to notice that there are two additional clock zones to perform the same logic behavior. The decision of what gate to choose in the circuit is further discussed in the next section, because the complex gate has problems with noise paths.

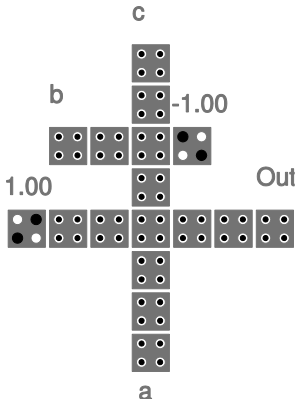

Fig. 8 – QCA complex gate (F=A+B•C).

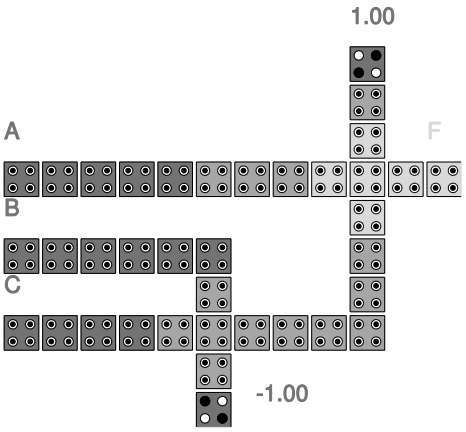

Fig. 9 – F=A+B•C gate with cascade association of majority gates.

V. QCA Gate Analysis

In this paper, three design analysis are done, using the QCADesigner tool [16]. The first one is the verification of the functional correctness of the circuit. The second analysis is the timing of the circuit. The final analysis is to check the noise path.

In Fig. 10, the waveform of OR expression can be seen. The waveform in “blue” are the generated signals (the inputs) and the “yellow” is the result of the expression OR, whereas is possible to analyze the timing using the “red” waveform that is the clock signal. The conclusion about timing is that this circuits calculate the OR logic within one clock zone (each clock zone is one quarter of the total clock).

The noise path analysis is done in the F complex gate. In Fig. 11, it is shown partial results for the F gate. Within the circles are the problematic signals, the path for which this signals pass through is called noise path. The noise path can propagate a wrong signal in larger circuits. The decision about use one circuit with a noise path have to be done carefully and the verification of the functional correctness of the largest circuit have to be done to certify that this noise do not cause a failure in the system.

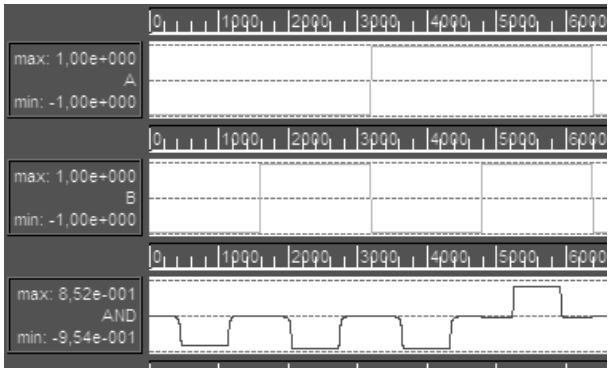

Fig. 10 – Waveform of QCA AND gate.

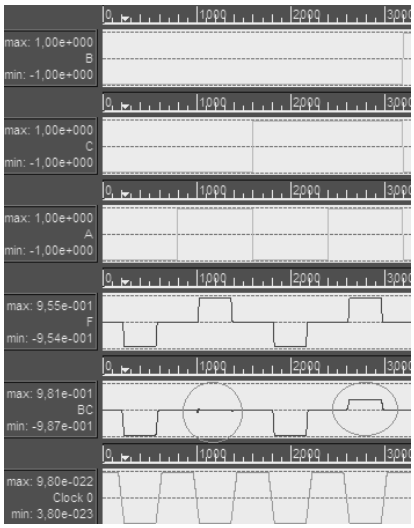

Fig. 11 – Noise path in the F=A+B•C complex gate.

VI. Conclusions

PPA is an flexible circuit that can be optimized for performance are area criteria. It is important to have this option in design for using QCA technology in the future. This paper introduced how to build a PPA using QCA technology. Even though to build a PPA is just to connect the circuits presented herein, the routing is a whole challenge because of many problems like the bidirectional nature of QCA, the proximity of the cell (the Coulomb force interaction), the noise propagation or the decision of how much clock zones to use in each step of the circuit. Future works include the construction of a PPA and the comparison to other adders, like RCA and CLA. Other goal to future work is to set up a formal way to analyze and compare the circuits.

Acknowledgment

Research partially supported by Brazilian funding agencies CNPq and FAPERGS, under grant 11/2053-9 (Pronem).

References

[1] K. Kong, Y. Shang and R. Lu, "An Optimized Majority Logic Synthesis Methodology for Quantum-dot Cellular Automata". VLSI Design, Vol 8, pp. 170-183, 2010.

[2] R. Zhang, K. Walus, W. Wang and G. Jullien, "A Method of Majority Logic Reduction for Quantum Cellular Automata". IEEE Transactions on Nanotechnology, Vol 3, pp. 443-450, 2004.

[3] C. S. Lent, D. Tougaw, W. Parod and C. Bernstein, "Quantum Cellular Automata". Nanotechnology, pp 49-57, 1993.

[4] D. Tougaw and C. Lent, "Logical Devices Implemented Using Quantum Cellular Automata". Applied Physiscs, Vol. 75, pp. 1818-1825, 1994.

[5] K. Walus, A. Vetteth, G. A. Jullien and V. S. Dimitrov, "RAM Design Using Quantum-Dot Cellular Automata". Proc. 2003 Nanotechnology Conf., Vol. 2, pp. 160-163, 2003.

[6] Mohammad Torabi, "A New Architecture for T Flip Flop Using Quantum-Dot Cellular Automata". 3rd Asia Symposium on Quality Eletronic Design, pp. 269-300, 2011.

[7] Heumpil Cho, "Adder Designs and Analyses for Quantum-Dot Cellular Automata". IEEE Transactions on Nanotechnology, Vol. 6, May 2007.

[8] I. Koren, "Computer Arithmetic Algorithms". A. K. Paters, 2ª edition, pp. 93-132, 2002.

[9] K. A Escobar, L. C. Manique and R. P. Ribas, "Optimal Arrangement of Parallel Prefix Adder (PPA) Trees According to the Area and Performance Criteria". South Symposium on Microelectronics, pp. 161-165, 2010.

[10] R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders". IEEE Trans. Computers, Vol. C-31, pp. 260-264, 1982.

[11] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class Recurrence Equations". IEEE Trans. Computers, Vol. C-22, pp. 786-793, 1973.

[12] C. S. Lent and P. D. Tougaw, "A Device Architecture for Computing with Quantum Dots". Proc. IEEE, Vol. 85, April 1997.

[13] K. Kim, K, Wu and R. Karri, "The Robust QCA Adder Designs Using Composable QCA Building Blocks". IEEE Trans. On Computer-Aided Design Integrated Circuits and Systems, Vol. 26, 2007.

[14] R. E. Ladner and M. J. Fischer, "Parallel Prefix Computation". JACM, Vol. 27, pp. 831-838, 1980.

[15] T. Hans and D. A. Carlson, "Fast Area-Efficient VLSI Adders". Proc. 8th IEEE Symposium on Computer Arithmetic, pp. 49-56, 1987.

[16] K. Walus, T. J. Dysart, G. A. Jullien and R. A. Budiman, "QCADesigner: A Rapid Design and Simulation Tool for Quantum-Dot Cellular Automata". IEEE Transaction on Nanotechnology, Vol. 3, pp. 26-31, 2004.

[17] R. Zhang, K. Walus, W. Wang and G. A. Jullien, "A Method of Majority Logic Reduction for Quantum Cellular Automata". IEEE Trans.on Nanotechnology, Vol. 3, pp 443-450, 2004.