Fast parallel-prefix Ling-carry adders in QCA nanotechnology

A. Thanos and H. T. Vergos
Computer Engineering & Informatics Department
University of Patras, 26500, Greece
E-mail: {athanos, vergos}@ceid.upatras.gr

***Abstract* - Ling-carry adders that use a Ladner-Fischer parallel-prefix algorithm for carry computation are explored for quantum-dot cellular automata (QCA) nanotechnology implementations. The results derived from drawn layouts, indicate that the proposed adders outperform in terms of delay all previously reported architectures explored for QCA implementations. The delay difference is becoming bigger in favor of the proposed adders as we move to larger wordlengths.**

1. INTRODUCTION

CMOS technology's role as the dominant VLSI implementation technology is expected to face serious problems in the near future due to limitations such as short channel effects, doping fluctuations and increasingly difficult and expensive lithography at nano scale. Novel innovative technologies need therefore to be investigated as possible alternatives. To this end, several nanotechnologies have been considered, including carbon nanotubes, silicon nanowires, resonant tunneling diodes etc. Quantum-dot Cellular Automata (QCA) has been introduced as one of them [1] and is considered a revolutionary and a very promising nanotechnology due to outstanding energy efficiency, high density, fast computing performance and room temperature operation.

Although QCA devices implementation is still under development, circuit architectures in QCA need to be researched, taking into account that the algorithms proposed for implementing circuits in CMOS may not necessarily be optimal for other technologies due to the specific characteristics of each technology.

In this manuscript we focus on adder implementations in QCA since addition is the most fundamental operation in every digital system. Several adder designs have been proposed for QCA nanotechnology [2-4], including carry look ahead, ripple carry [2] and carry flow adders [3]. Ladner-Fischer parallel-prefix adders and a hybrid structure built by mixing Ladner-Fischer and ripple-carry stages have also been considered in [4]. In this paper, we investigate the suitability for QCA of an adder, which uses the Ling carry definition and a parallel-prefix computation unit. We present QCA drawn layouts and simulation results for the proposed adder. Our results indicate that this proposed adder architecture offers the least execution delay among all known proposals.

The rest of the paper is organized as follows. Section 2 provides some background on QCA nanotechnology. A brief overview of the parallel-prefix addition formulation, along with Ling carry definition and a theoretical delay analysis of the parallel-prefix Ling adder in QCA are given in Section 3. Section 4 presents implementations of the proposed adder with example layouts drawn and simulation results as well as comparisons with previously proposed architectures.

2. QCA OVERVIEW

A QCA circuit is built using cells. Each cell is a square nanostructure. The basic cell or cross cell has four dots at the four corners of the square. In each cell two electrons are injected. Each dot can hold only a single electron. As the two electrons repulse each other because of the Coulombic interaction, they occupy dots that are in maximal distance. As a consequence, the two electrons inside the cell can be in two possible polarization states that are used to represent binary values. Fig. 1 depicts the cross cell and the two polarizations states for binary "0" and "1".

Cells are then used to construct wires, majority gates and inverters. A non-inverting wire (see Fig. 3a) is made up by an array of cells. A four phase clocking scheme is used in QCA for signal propagation. The four phases of the clock are namely switch, hold, release, and relax which correspond to the high to low, low, low to high and high states of the clock respectively. In switch state the cell begins computing, in hold state holds its value, in release is released and in relax state is inactive. Fig. 2 presents the propagation of a signal across a wire made by cross cells and how signal flow is accomplished with the four-phase clocking scheme. Cells in different clock zones are usually indicated by different colors. Due to Coulombic repulsion adjacent cells are forced to the same polarization, propagating this way the binary value with a direction of input to output depending on the clock zones of cells.

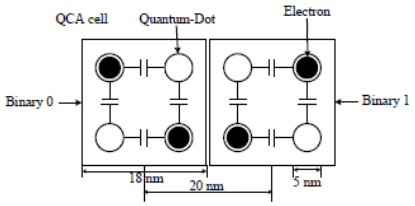


Fig.1. QCA cross cells, polarizations and encodings.

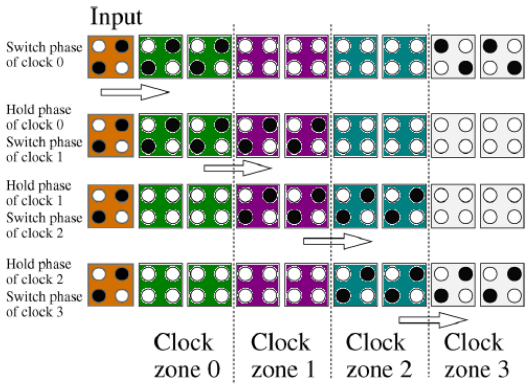


Fig. 2. Signal propagation using the clocking scheme

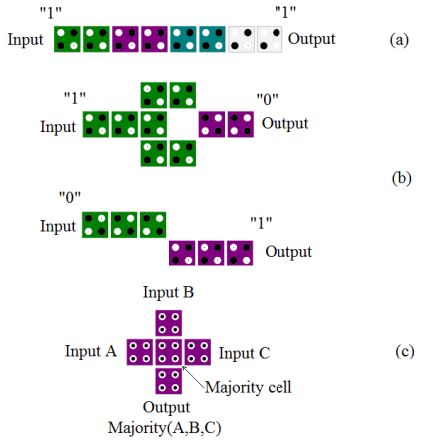


Fig.3. Basic elements: (a) QCA binary wire (b) Inverter and (c) The majority gate.

Inverters can be built in different ways. Fig. 3b indicates two of them. More complex logic gates are built around the majority gate. A majority gate (Fig. 3c) is a cross-shaped structure of five cells, with three input cells, an output cell and the centre cell of the structure called majority cell. In order to ensure proper operation for the majority gate, all five cells of the gate must be clock zone shifted by one in relation to the wires that carry the input values. This stands on the fact that input values must reach on the cells at the same time in order to establish a valid value on the majority cell. This gate implements the majority function of three input binary values a, b and c defined as M(a,b,c)=a·b + b·c + c·a. Two-input AND and OR gates are derived by setting one input of a majority gate to "0" or to "1", respectively.

Multilayer crossovers are used for wire crossings in QCA layouts. This approach is similar to multiple metal layers in CMOS technology. In multilayer crossovers signals in two basic layers get connected by an intermediate one that only uses vertical cells for interconnection.

3. PARALLEL-PREFIX LING ADDERS

An n-bit adder which produces the sum S (S=$S_{n-1}S_{n-2}\ldots S_1S_0$) of two n-bit operands A and B (A=$A_{n-1}A_{n-2}\ldots A_1A_0$ and B=$B_{n-1}B_{n-2}\ldots B_1B_0$) is a three stage circuit.

The first called preprocessing stage computes the carry-generate bits g_i=a_i·b_i, the carry-propagate bits p_i=a_i+b_i and the half sum bits d_i=$a_i \oplus b_i$, where ·, +, ⊕ represent the logical AND, OR and XOR operation, respectively. The second stage computes the carry bits c_i according to

$$c_i = g_i + p_i \cdot c_{i-1} \quad (1)$$

The final stage produces the sum bits according to

$$s_i = d_i \oplus c_{i-1} \quad (2)$$

Carry computation which takes place in the second stage can be transformed into a prefix problem by using the associative operator ∘, which associates pairs of generate and propagate signals according to:

$$(g_i,p_i) \circ (g_{i-1},p_{i-1}) = (g_i + p_i \cdot g_{i-1}, p_i \cdot p_{i-1}) \quad (3)$$

In case of successive associations of the generate and propagate pairs at the k, k-1,…,j bit positions, the notation ($g_{k:j}$, $p_{k:j}$), is often used to represent the group generate and propagate term produced, that is,

$$(g_{k:j}, p_{k:j}) = (g_k,p_k) \circ (g_{k-1},p_{k-1}) \circ \ldots \circ (g_j,p_j) \quad (4)$$

Ling [5] by considering adjacent pairs of the input operands proposed a new carry formulation that speeds up the computation. The Ling carry H_i is defined as:

$$H_i = c_i + c_{i-1} \quad (5)$$

Using (1) and (5) it holds that:

$$H_i = g_i + g_{i-1} + p_{i-1} \cdot g_{i-2} + \cdots + p_{i-1} \cdot p_{i-2} \cdot \ldots \cdot p_1 \cdot g_0 \quad (6)$$

Ling carries can be computed in less logic levels than the normal carries. For example, c_3 is given by:

$$c_3 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0$$

whereas H_3 is :

$$H_3 = g_3 + g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0$$

The derivation of the sum bits in a Ling adder may at first seem more complicated. But since from (2) we have $s_i = d_i \oplus c_{i-1}$, and $c_i = p_i \cdot H_i$ holds, we can rewrite (2), as:

$$s_i = d_i \oplus (p_{i-1} \cdot H_{i-1}) \quad (7)$$

which can further be written [6] as:

$$s_i = \bar{H}_{i-1} \cdot d_i + H_{i-1} \cdot (d_i \oplus p_{i-1}) \quad (8)$$

Therefore, the final sum bits are obtained by using 2 to 1 multiplexers, from signals already available when the selecting Ling carry is computed.

Ling carry computation can also be transformed into a parallel-prefix problem [7, 8]. For example, in an 8-bit adder, H_4 and H_5 are computed by:

$$H_4 = g_4 + g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0$$

$$H_5 = g_5 + g_4 + p_4 \cdot g_3 + p_4 \cdot p_3 \cdot g_2 + p_4 \cdot p_3 \cdot p_2 \cdot g_1 + p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot g_0$$

Using that $g_i \cdot p_i = g_i$, the above equations can be written as:

$$H_4 = (g_4 + g_3) + (p_3 \cdot p_2) \cdot (g_2 + g_1) + (p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot g_0)$$

$$H_5 = (g_5 + g_4) + \big((p_4 \cdot p_3) \cdot (g_3 + g_2)\big) + (p_4 \cdot p_3 \cdot p_2 \cdot p_1) \cdot (g_1 + g_0)$$

Therefore, by defining $G_{i:i-1}$=g_i+g_{i-1} and $P_{i:i-1}$=p_i·p_{i-1}, H_4 and H_5 are expressed as:

$$H_4 = G_{4:3} + P_{3:2} \cdot G_{2:1} + P_{3:2} \cdot P_{1:0} \cdot G_{0:-1}$$

$$H_5 = G_{5:4} + P_{4:3} \cdot G_{3:2} + P_{4:3} \cdot P_{2:1} \cdot G_{1:0}$$

or equivalently in parallel-prefix notation as:

$$H_4 = (G_{4:3}, P_{3:2}) \circ (G_{2:1}, P_{1:0}) \circ (G_{0:-1}, P_{-1:-2})$$

$$H_5 = (G_{5:4}, P_{4:3}) \circ (G_{3:2}, P_{2:1}) \circ (G_{1:0}, P_{0:-1})$$

Similar forms can be derived for all the Ling carries of an n-bit adder. Any developed algorithm (for example [9], [10]) can be used for the parallel-prefix computation. Most often these algorithms are expressed by acyclic graphs in which the nodes represent the prefix operator. For example Fig.4(a) presents the Ladner-Fischer graph for an 8-bit adder, while Fig.4(b) depicts the implementation of the prefix operator.

In general, in an n-bit Ling adder, the Ling carries H_i and H_{i+1} are formulated as:

$$H_i = (G_{i:i-1}, P_{i-1:i-2}) \circ (G_{i-2:i-3}, P_{i-3:i-4}) \circ \ldots \circ (G_{0:-1}, P_{-1:-2})$$

$$H_{i+1} = (G_{i+1:i}, P_{i:i-1}) \circ (G_{i-1:i-2}, P_{i-2:i-3}) \circ \ldots \circ (G_{1:0}, P_{0:-1})$$

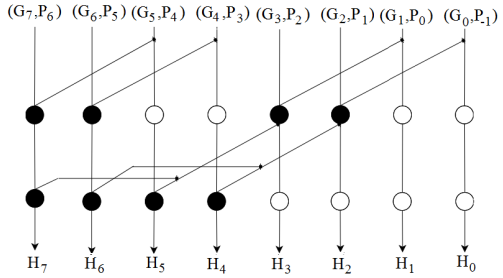
A carry-in signal, c_{in}, may also have to be incorporated to a parallel-prefix Ling adder. A straightforward way is to set g_{-1}=c_{in}. In this way one OR gate is added to the adder that computes the G_0 signal as: G_0=g_0+c_{in}.

The design of an n-bit parallel-prefix adder that uses the above Ling-carry formulation can be performed by the following three steps:

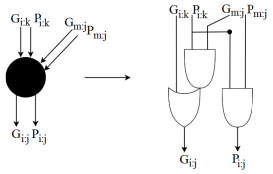
First step: Produce the g_i, p_i, d_i signals using AND, OR and XOR gates from the inputs A=$A_{n-1}A_{n-2}\ldots A_1A_0$ and B=$B_{n-1}B_{n-2}\ldots B_1B_0$. Then, use the g_i and p_i signals to generate the pair G_i, P_i signals using OR and AND gates respectively. XOR gates are also required to compute $(d_i \oplus p_{i-1})$ which will be used at the third step.

Second step: Use the G_i, P_{i-1} pairs generated from step one and prefix operators, to compute the Ling carries. In the case of an n-bit adder this requires $\log_2 n$-1 parallel prefix stages.

Third step: Obtain the sum bits s_i by using multiplexers that implement (8) and are controlled by the Ling carries



(a) Parallel-prefix graph



(b) Prefix operator implementation

Fig. 4. Ladner-Fischer parallel-prefix algorithm for an 8-bit adder.

computed in step two. Derive the carry-out bit by using an AND gate that takes as inputs the H_{n-1} carry and the p_{n-1} bit.

Table 1 presents the clock zones required for implementing the above steps in QCA. Step 1 requires AND and OR gates that operate in parallel for computing the p_i and g_i signals followed by OR and AND gates, respectively, that compute the G_i and P_i pairs. All these gates in QCA have a latency of one clock zone (are implemented by a single majority gate). The computation of d_i and $p_{i-1} \oplus d_i$ signals is performed by XOR gates in two clock zones each (three majority gates are required per XOR gate with the two of them operating in parallel) but does not contribute to the critical path of the adder. Along with the clock zone required for obtaining the inputs, step 1 can be implemented in QCA with a latency of 3 clock zones. For the second step $\log_2 n$-1 parallel-prefix stages are utilized. Each stage consists of prefix operators (implementation shown in Fig. 4.b). Each operator contributes a delay of 2 clock zones. Therefore, the delay of the second step is 2 x ($\log_2 n$-1) clock zones. Finally, the implementation of the multiplexers of the third step can be performed by three majority gates, with two of them operating in parallel, which leads to a delay contribution of 2 clock zones. Summing all the above we conclude that the parallel-prefix Ling carry adders have an execution delay of T = 5 + 2 x ($\log_2$n-1) clock zones.

Unfortunately, the above estimation is a lower limit of the actual delay since it totally ignores wire delay. Wire delay in QCA stems from long wires that require more cells than the maximum allowed in a single clock zone. In such wires, some cells must be clock zone shifted for satisfying the maximum cell count restriction set for robust design and for ensuring proper signal flow through wires.

Table 1. Latency in QCA parallel-prefix Ling adder

Step	Signal Computation	Gate required	Clock zones	On Critical path
1	g_i	AND	1	Yes
	p_i	OR	1	Yes
	d_i	XOR	2	No
	d_i ,pi_{-1}	XOR	2	No
	G_i	OR	1	Yes
	P_i	AND	1	Yes
2	H_i	$\log_2 n$-1 stages of prefix operator	2*($\log_2 n$-1)	Yes
3	s_i	2->1 multiplexer	2	Yes

4. IMPLEMENTATION AND COMPARISONS

For our experiments we used the QCA Designer tool [11] which has been specifically developed and commonly adopted for layout drawing and simulating of QCA designs. Using the QCA Designer, we have drawn layouts for the proposed n-bit adders for n = 4, 8, 16, 32 and 64. For our implementations, we considered the Ladner-Fischer parallel-prefix algorithm [10] since it leads to more than 20% less majority gates than the Kogge-Stone [9] algorithm in all n > 4 adder cases.

Each cell used is 18nm in both width and height with its quantum dots have a diameter equal to 5nm. Cells are placed on a grid with a cell center-to-center distance of 20nm. In order to limit the propagation delays between cell-to-cell reactions and ensure reliable signal transmission, a maximum and minimum cell count in each clock zone must be defined. For our experiments, we assumed a maximum and minimum count of 30 and 2 cells, respectively. This resulted in zero wire delay for the 4-, 8- and 16-bit adder cases. In the 32- and 64-bit adder cases however, clock zone shifting was necessary, imposing a maximum wire delay of 0.75 (3 zones) and 2.5 (10 zones) clocks, respectively.

Figs. 5 and 6 present the drawn layouts for n = 8 and 16, respectively, while Figs. 7 and 8 present simulation results that verify the correctness and indicate the addition delay.

Table 2 presents comparison results against previous adder proposals in QCA. The metrics considered are cell count, area and delay. The results indicate that ripple-carry adders (RCA) [2] only offer small implementations for narrow operands. The carry lookahead (CLA) [2] architecture is also not a good alternative for QCA, since it requires large area without offering significant savings in delay. The carry-flow adder (CFA) architecture [3] offers the smallest implementations with low delay in the narrow operands' cases. The Ladner-Fischer [4] adders use approximately half the cells of the CLA adder, but more

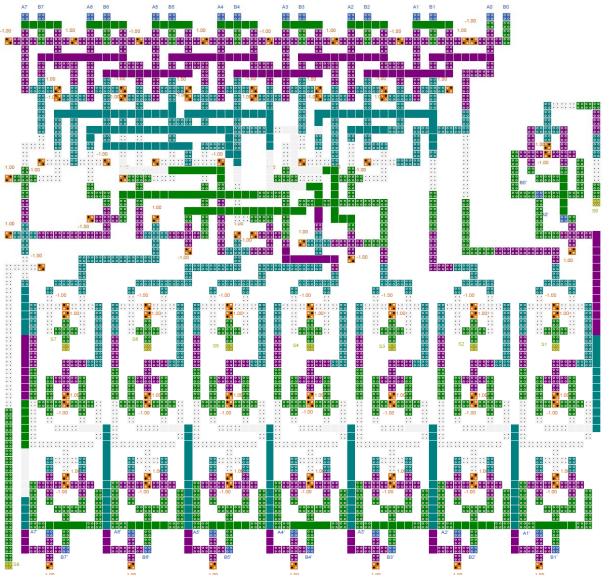


Fig. 5. Drawn layout for the proposed 8-bit adder.

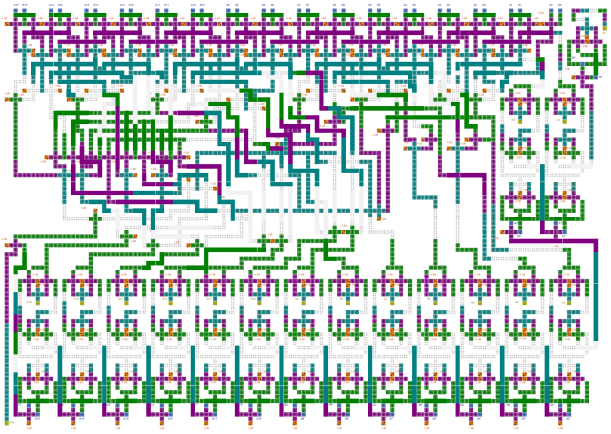


Fig. 6. Drawn layout for the proposed 16-bit adder.

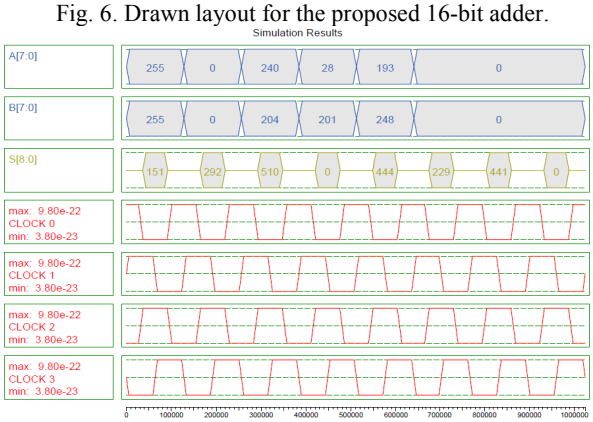


Fig. 7. Waveforms from the 8-bit adder simulation.

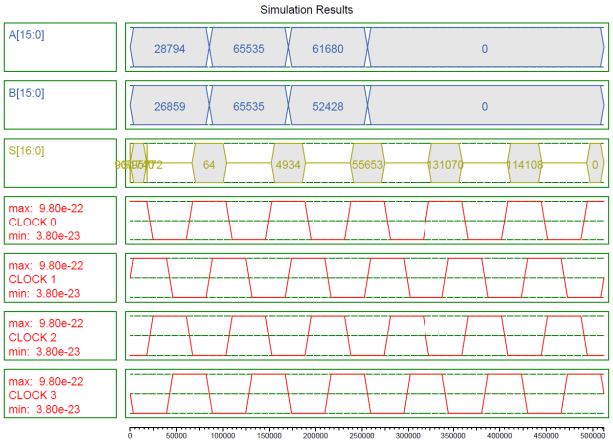


Fig. 8. Waveforms from the 16-bit adder simulation

than twice the cells of an CFA. While their delay is bigger than that of the CFA in the 4 and 8-bit adder cases, they are faster in wider wordlengths. Finally, the hybrid adder solution [4] offers both small implementations and low delay for sufficiently wide operands.

The proposed adders require smaller area than the CLA adders. Their most significant merit however, is that they offer the least delay compared to all other approaches excluding the 4-bit adder case. The difference in delay savings offered by the proposed adders grows as the wordlength increases, despite the added wire delay.

5. CONCLUSION

In this manuscript, parallel-prefix Ling-carry adders that achieve very low execution delay in QCA nanotechnology were investigated. The examined adders use the Ladner-Fischer parallel-prefix structure assuming the Ling carry formulation. Layouts for the proposed adders were drawn to verify their correctness and estimate the wire delay. The comparisons against previous solutions indicate significant delay savings that grow at larger wordlengths.

Table 2. Complexity of different adders

Word-length	Adders	Cell-Count	Area ($\mu m*\mu m$)	Delay (clocks)
4	Proposed adder	977	0.94*1.1	1.75
	LF[4]	698	0.87*0.71	2
	Hybrid[4]	475	0.8*0.53	1.75
	RCA[2]	651	1.67*0.72	4.25
	CLA[2]	1575	1.74*1.09	3.5
	CFA[3]	371	0.9*0.45	1.5
8	Proposed adder	2562	1.47*1.43	2.25
	LF[4]	1994	1.67*1.06	2.75
	Hybrid[4]	1422	1.34*0.83	2.25
	RCA[2]	1499	3.43*1.04	8.25
	CLA[2]	3988	3.5*1.58	6.5
	CFA[3]	789	1.79*0.53	2.5
16	Proposed adder	6592	2.75*1.95	2.75
	LF[4]	5376	3.29*1.46	4.25
	Hybrid[4]	3555	2.61*1.02	3.25
	RCA[2]	3771	6.97*1.69	16.25
	CLA[2]	10217	7.02*2.21	10.25
	CFA[3]	1769	3.55*0.69	4.5
32	Proposed adder	17139	5.3*2.93	4
	LF[4]	13552	6.56*2.02	7.5
	Hybrid[4]	8946	5.55*1.48	5.5
	RCA[2]	10619	14.0*3.01	32.25
	CLA[2]	25308	14.06*3.05	19
	CFA[3]	4305	7.09*1.03	8.5
64	Proposed adder	44047	10.34*4.58	6.25
	LF[4]	35850	13.4*2.81	13.5
	Hybrid[4]	22427	11.6*2.18	9.5
	RCA[2]	33531	28.2*5.65	64.25
	CLA[2]	59030	28.2*3.73	31.5
	CFA[3]	11681	14.2*1.71	16.5

REFERENCES

[1] C.S. Lent, et al., "Quantum Cellular Automata," *Nanotechnology*, vol. 4, no. 1 pp. 49-57, Jan. 1993.

[2] H. Cho and E. E. Swartzlander, "Adder Designs and Analyses for Quantum-Dot Cellular Automata," *IEEE Trans. Nanotechnology*, vol. 6, no. 3, pp. 374-383, May 2007.

[3] H. Cho, and E. E. Swartzlander, "Adder and Multiplier Design in Quantum-Dot Cellular Automata," *IEEE Trans. On Computers,* vol. 58, no. 6, pp. 721-727, June 2009.

[4] V. Pudi, and K. Sridharan, "Efficient Design of a Hybrid Adder in Quantum-Dot Cellular Automata," *IEEE Trans. on VLSI,* vol. 19, no. 9, pp. 1535-1548, Sept. 2011.

[5] H. Ling, "High-speed Binary Adder", *IBM J. Research & Development*, vol. 25, pp .156-166, May 1981.

[6] S. Vassiliadis, "Recursive Equations for Hardwired Binary Adders", *Int. J. of Electronics*, vol. 67, no. 2, pp. 201-213, August 1989.

[7] C. Efstathiou, H. T. Vergos and D. Nikolos, "Ling adders in CMOS standard cell technologies," *in Proc. of the 9th ICECS*, pp. 485-488, vol. 2, 2002.

[8] G. Dimitrakopoulos and D. Nikolos, D., "High-speed parallel-prefix VLSI Ling adders," *IEEE Trans. On Computers,* vol. 54, no. 2, pp.225 -231, Feb. 2005.

[9] P. Kogge, and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Trans. On Computers*, vol. C-22, no. 8, pp. 786-793, Aug. 1973.

[10] R. E. Ladner and M. J. Fischer, "Parallel Prefix computation", *J. ACM*, vol. 27, no. 4, pp 831-838, Oct. 1980.

[11] K. Walus, et al., "QCADesigner : a Rapid Design and Simulation Tool for Quantum-Dot Cellular Automata," *IEEE Trans. Nanotechnology*, vol. 3, no. 1, pp. 26–31, March 2004.