Design of a Hybrid Adder using QCA in MAC unit

VIGNESHWARI.R (PG scholar),
Department of ECE,
PSN College of Engineering and Technology,
Tirunelveli.
vigneshwari.raji@gmail.com

JENOPAUL.P (Professor)
Department of ECE,
PSN College of Engineering and Technology,
Tirunelveli.
jenopaul1@rediffmail.com

Abstract - Quantum dot cellular automata design of a hybrid adder is proposed to reduce the area, delay and power consumption of an adder. In previous Quantum dot cellular automata design of a ladner-fischer prefix adder has been implemented. Hybrid adder has a better performance than the ladner-fischer adder. Hybrid adder has a minimum area, delay and power consumption. In this paper we are going to implement the hybrid adder in the application of multiple accumulator units.

Index Terms—Hybrid adder, Ladner–Fischer adder, nanoelectronics, quantum-dot cellular automata (QCA), MAC unit.

I. I NTRODUCTION

Nanoelectronics devices have been of interest to the research community during the last decade. These include carbon nano tubes, silicon nanowires, resonant tunneling diodes, and others. These devices have emerged as alternatives to the traditional VLSI technology based on CMOS. Conventional device physics is based on a free electron model and as device dimensions shrink (to be of the order of the wavelength of an electron), this model is not appropriate since the energies an electron is allowed to have become discrete. A recent book on nano electronics [1] provides a good introduction to the quantum mechanics of electrons, notions of free and confined electrons as well as single electron and many electron devices. One of the devices suggested in the literature as an alternative to the traditional CMOS-based technology is the quantum-dot cellular automata (QCA). In QCA, the device used for logic is also used for interconnect. The basic logic gates in the QCA architecture are the majority gate (also referred to as the majority voter) and the inverter. The focus of this paper is on design of arithmetic circuits in QCA. In particular, our interest is in efficient design of multi-bit adders. One possible approach is based on examination of the best adders (meeting some criteria) developed for existing technologies such as CMOS for adaptation to new ones such as QCA. However, algorithms that have been optimally implemented in one technology may not necessarily be the best in a different technology [2].In this paper, we pursue the following two directions. We consider area occupied by basic logic elements in a QCA design and examine optimization of logic. We also examine development of special adders for the QCA model. The "quantity" of logic also indirectly determines the "amount" of QCA wires in a design. Prior work on adder designs has examined a few directions. Wang et al. [3] present an efficient design of a 1-bit QCA adder that uses three majority gates and two inverters. Majority logic reduction for several three variable Boolean functions is studied in [4] a performance comparison of some QCA adders is presented in [5]. Modular design of conditional sum adders is studied in [6]. Tang et al.[7] have presented a QCA circuits design methodology based on traditional CMOS circuits design flow and a SPICE model. Ripple carry and carry look ahead adder designs in QCA are presented in [8]. Robust QCA adder designs that exploit proper clocking schemes are proposed in [9]. Probabilistic analysis of molecular quantum-dot cellular adders is presented in [10]. Reliability of magnetic QCA adders and electrostatic QCA adders is studied via probabilistic transfer matrices in [11]. Robust adders based on QCA are described [12]. A model of QCA circuits using Bayesian networks is presented in [13] .Hierarchical probabilistic macro modeling for QCA circuits is described in [14] . Energy dissipation per clock cycle in QCA adder circuits is studied in [15]. Cho and Swartz ladner [16] have presented design of a carry flow adder and a multiplier in QCA. Synthesis tools for QCA have been reported in [17] and [18].

To the best of our knowledge, there is no prior work that has examined in detail properties of basic logic elements in QCA for obtaining very efficient adder designs. Further, special algorithms for addition in the QCA model are limited. This paper begins by developing a QCA-based solution for the Ladner–Fischer prefix adder. The paper then proceeds to develop an adder that is a hybrid of Ladner–Fischer and ripple carry adder. This hybrid adder is shown to be well-suited to the QCA model. We demonstrate that the hybrid adder has lower delay (in view of parallelism) than the best existing adder designs in the literature. Further, the hybrid adder compares well with

existing adders in terms of area of the QCA design (since it incorporates best features of ripple carry adders). We also show that the hybrid adder has a smaller area-delay product than existing adder designs in QCA. Results of simulation using QCA Designer [17] support the theory presented. The remainder of this paper is organized as follows. Section II provides the basic notations pertaining to QCA. Section III presents QCA design of QCA design of the proposed hybrid adder. Section IV presents the details of simulation in QCA Designer. Section V presents comparisons with prior work.VI presents the details of multiplier accumulator unit. Conclusions are presented in Section VII.

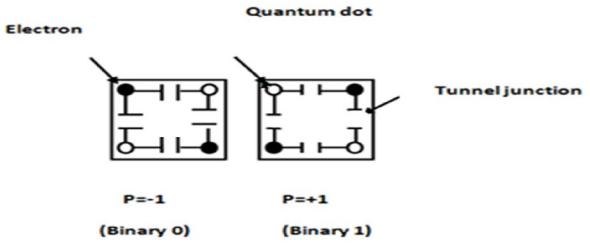


Fig. 1. QCA cells with electrons indicating possible polarizations.

II. BASICS OF QCA

In QCA, the logic states are not stored as voltage levels [19]. Instead, the location of individual electrons determines the bi-nary state. Fig. 1 depicts QCA cells. Each QCA cell is a set of four dots positioned at the four corners of a square. Each QCA cell is occupied by two electrons. Several approaches have been suggested for computation with an array of QCA cells. One approach is based on transferring the array to an excited state from a ground state by merely applying input data (without explicit clocking). The array is expected to settle to a new ground state. However, sometimes the transition may result in a meta stable intermediate state. To facilitate transfer to a new ground state, another approach based on clocking has been suggested. Clocking (by application of an appropriate voltage to a cell) leads to adjustment of tunneling barriers between quantum dots for transfer of electrons between the dots.

Clocking is performed in one of two ways: zone clocking and continuous clocking. In zone clocking , each QCA cell is clocked using a four-phase clocking scheme as shown in Fig. 2. The four phases correspond to switch, hold , and release and relax. In the switch phase, cells begin un polarized and with low potential barriers but the bar riers are raised during this phase. In the hold phase, the barriers are held high while in the release phase, the barriers are lowered. In the last phase, namely relax, the barriers remain lowered and keep the cells in a un polarized state. An alternative to zone clocking, called continuous clocking, involves generation of a potential field by a system of submerged electrodes. The former clocking scheme is adopted in this paper since the CAD tool mused for simulation supports zone clocking and further, prior works on adders are only based on this clocking scheme. Primitives in the QCA model consist of a wire, inverter and majority gate and are depicted in Fig. 3. Fig. 4 shows the operation of a wire in different clock zones. A majority gate in QCA takes three inputs and implements the majority function of three Boolean variables a, b and c. The majority function is denoted by M (a,b,c) is defined as M (a,b,c) =ab+bc+ca.

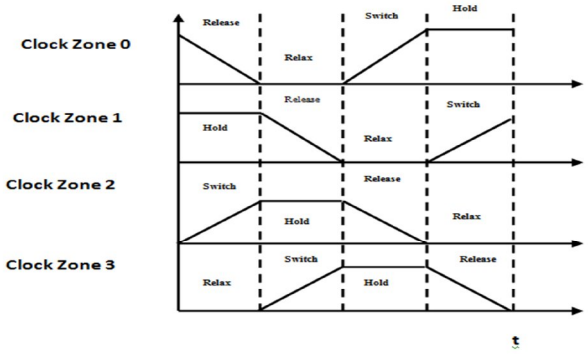


Fig. 2. QCA clock zones.

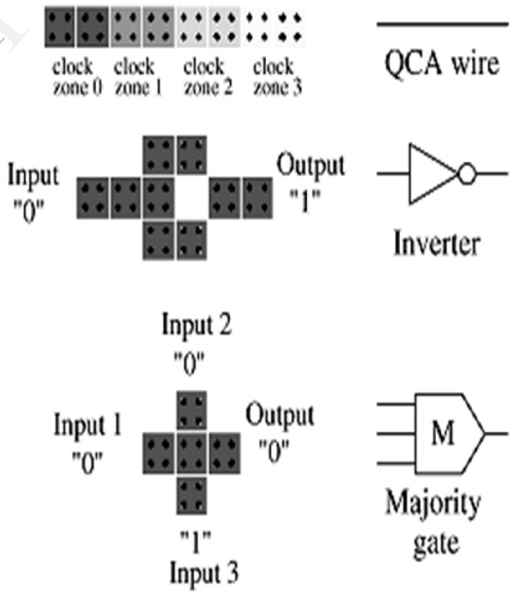


Fig. 3. QCA wire, inverter, and majority gate.

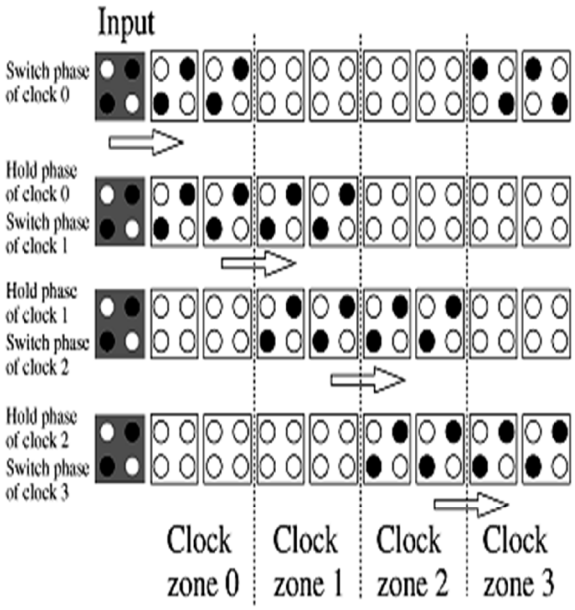


Fig. 4. Operation of a wire in different clock zones.

A QCA design permits two options for crossover, termed coplanar crossover and multilayer crossover. While the coplanar crossover uses only one layer but involves usage of two cell types (termed regular and rotated), the multilayer crossover uses more than one layer of cells (analogous to multiple metal layers in a conventional IC). Multilayer crossover is predominantly used in this paper for wire crossings since the design is fairly simple. Some results for coplanar crossover are also presented for comparison purposes.

III. NEW HYBRID ADDER IN QCA

Design of 8- and 16-bit Hybrid Adder in QCA While the Ladner–Fischer adder supports parallelism, the requirement of majority gates (which contributes to the overall area) is quite high. The large number of majority gates has an indirect effect on the wire (delay and amount). It is therefore of interest to explore ways of reducing the area. From the literal true, it is known that ripple carry adders are simple and have low area requirement. This fact is taken advantage of in our design of a hybrid adder. In this section, we present an adder which is a hybrid of the Ladner–Fischer adder and a ripple carry adder. We show that this hybrid adder has advantages especially in the QCA domain over the Ladner–Fischer adder as well as the ripple carry adder in terms of delay. Further, the hybrid adder requires a substantially lower number of majority gates for different adder sizes in comparison to a Ladner–Fischer adder.

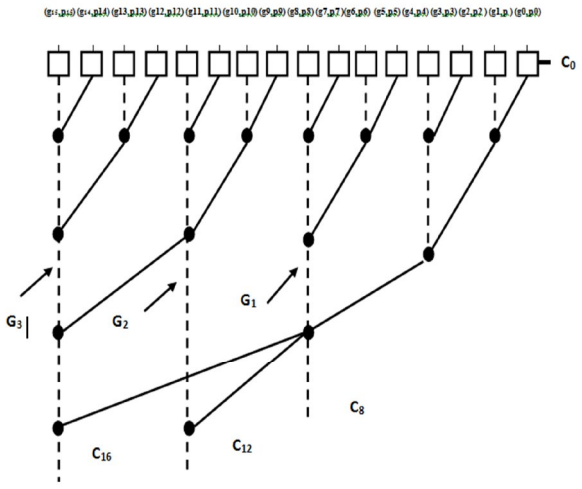


Fig. 5. Tree structure for C_8,C_{12} and C_{16} carries.

Fig. 4 shows a general tree structure for computation of C_{16},C_{12} ,C_8 and C_4 Parallel prefix graphs for various adders can be derived from this. The proposed hybrid adder is based on the idea that a number of carries not explicitly labeled in Fig.4 (in particular,C_1,C_2,C_3,C_4,C_5,C_6,C_9,C_{10},C_{13},C_{14}) can be computed using a ripple carry style leading to a highly efficient adder in terms of majority gates and delay. We will investigate first the majority gate requirement for carries C_{16}, C_{12} ,C_8 and C_4 With reference to Fig. 4, we note that C_{16}, C_{12} and C_8 depend on G_1,G_2 and G_3 respectively. Let G_j,j = 1, 2,3 represent generate $(g_{i+3},p_{i+3})o(g_{i+2},p_{i+2})o(g_{i+1},p_{i+1})o(g_i,p_i)$, where I = 4, 8, and 12, respectively. G_j can be expanded as G_j can be expressed using majority gates. Using Proposition 1, we note that g_{i+1} + $p_{i+1}g_i$ can be expressed as $M(g_{i+1},p_{i+1},g_i)$. Direct evaluation of G_j using above equation requires up to ten majority gates since g_{i+3},p_{i+3},g_{i+2},p_{i+2},g_{i+1},p_{i+1} and g_i require one majority gate each besides the three majority gates shown explicitly in the expression for Gj. Consequently, a total of 30 majority gates is required for G_1,G_2 and G_3 We now present a new result that reduces the majority gate requirement substantially.

Proposition 1: Let f_1,f_2 and f_3 be three Boolean functions such that $f_1=x_1y_1$ and $f_2=x_2+y_2$ where x_1 and y_1 are two binary inputs.Then

$M(f_1,f_2,f_3) = M(x_1,y_1,f_3).$

Proof: $M(f_1,f_2,f_3)$ is given by

$M(f_1,f_2,f_3) = f_1f_2 + f_1f_3 + f_2f_3$

$= f_1f_2 + (f_1 + f_2)f_3$

$= x_1y_1(x_1 + y_1) + (x_1y_1 + x_1 + y_1)f_3$

$= x_1y_1 + (x_1 + y_1)f_3$

$= M(x_1,y_1,f_3).$

Majority logic expressions for compute c_8,c_{12} and c_{16} can now be given as shown in below equation. The details are given in the equation

$C_8 = M(G_1,p_7,p_6p_5p_4c_4)$
$C_{12} = M(G_2,p_{11},p_{10}p_9p_8c_8)$
$C_{16} = M(G_{3+} p_{15}p_{14}p_{13}p_{12}\ G2,_+ p_{15},p_{14}\ldots c8).$

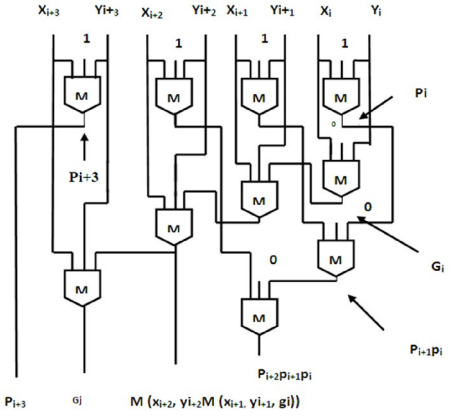


Fig. 6. Four-bit hybrid adder block

The requirements for majority gates for c_7,c_8,c_{11},c_{12}, c_{15} and gates are required (including the requirements for p_i's and g_4 , majority gate for AND of p_6 and p_5 , AND of p_4 and c_4 , AND of p_6 p_5 and p_4c_4 theOR operation). For in c_8 of above equation three majority gates are required (by reutilization of majority gates for p_6, p_5, etc.) since p_7,G_1 and M() require one majority gate each (G_1 is given by $M(x_7,y_7,M(x_6,y_6,M(x_5,y_5,g_4)))$ and the majority gates involved in computation of c_7 , namely $M(x_6,y_6,M(x_5,y_5,g_4))$, $p_6p_5p_4c_4$) are reutilized for $G_1$and c_8). For c_{11}, the calculations are similar to c_7 and so ten majority gates are required.

For c_{12} , the calculations are similar to c_8 , therefore three majority gates are required. For c_{15}, the calculations for the part given by $M(x_{14},y_{14},M(x_{13},y_{13},g_{12}))+$ $p_{14}p_{13}p_{12}G_2$ are similar to c_{11} hence ten majority gates are required. For … + $p_{14}p_{13}p_{12}p_{11}p_{10}p_9p_8c_8$, four majority gates are required (one majority gate for AND of p_{11} with $p_{10}p_9p_8$, one majority gate for AND of $p_{14}p_{13}p_9p_{12}$ with $p_{11}p_{10}p_9p_8$, one majority gate for AND of $p_{14}p_{13}p_{12}p_{11}p_{10}p_9p_8$ with c_8 and one for the overall OR operation). Hence, altogether 14 majority gates are required. For c_{16}, four majority gates are required (one for M(.) and three for above equation and this is in view of the following facts: 1)c_{16} computation reutilizes majority gates involved in c_{15} and this is similar to the manner in which c_8 computation reutilizes majority logic used for c_7 ; 2) the first term within M() in c_{16} equation is realized as shown in above equation; and 3) In above equation G_3,p_{15} and $M(G_3,\ldots)$ require one majority gate each. Fig. 5 illustrates the generation of carries of a 16-bit hybrid adder. The calculation of the remaining carries, namely ,c_i,i = 1, 2, 3, 4, 5, 6, 9, 10, 13, 14 can be done in ripple-carry style (each $c_i$can be expressed in terms of $M(x_{i-1},y_{i-1},c_{i-1})$). For example, carry c_5 can be calculated as $M(x_4,y_4,c_4)$. So ten majority gates are required for these ten carries. The requirements for sum are the same as for the Ladner–Fischer adder. In particular, 32 majority gates, and 16 inverters are required for a 16-bit hybrid adder. The total requirements are summarized by Proposition 2.

Proposition 2: A 16-bit hybrid adder requires 86 majority gates and 16 inverters.

Remark :In Fig. 5, some of the carries computed in ripple carry style are shown enclosed in ellipses (& labelled with $).

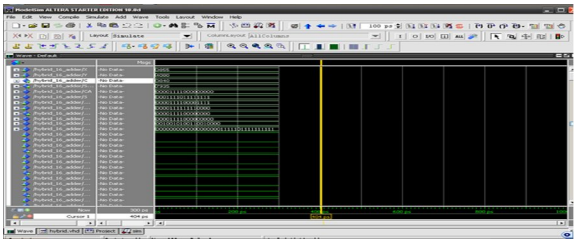
For example C_5 and C_6 in Stage 3 are computed in ripple carry style with one majority gate (delay) difference between C_5 and C_6 . However, C_7 and C_8 are computed in prefix style and in parallel with C_6 . Similarly, in Stage 4, C_9 and C_{10} are computed in ripple carry style but C_{10} is computed in parallel with C_{11} and C_{12} (the latter are obtained in prefix style). The result on majority gates for a 16-bit hybrid adder can be compared with that of the 16-bit Ladner–Fischer adder. In particular, 45 majority gates are reduced here which is approx-imately reduction by 35%.

Remark : One can comment on the latency reduction for 8-bit Hybrid adder in comparison to an 8-bit Ladner–Fischer adder. In the former, the critical path for c_8 involves six majority gates while the same is required for c_8 in Ladner–Fischer .

However, in the latter, one clock zone delay is incurred in obtaining g_0 and p_0 (they can be obtained in parallel, hence just one zone delay) whereas in the hybrid adder, g_0 's and p_0 's are not required at all thereby a reduction of 0.25 units (corresponding to one majority gate) of delay is possible. We also note that besides g_0 , computation of several other g's is skipped in the case of a hybrid adder. These include g_1, g_2, g_3, g_4, g_5, g_6 and g_7 for an 8-bit hybrid adder. This leads to a substantial reduction in majority logic. However, the reduction is delay is only 0.25 units (due to the computation of g 's happening in the "same" stage). Further reduction in delay (in a hybrid adder) is dependent isdependent on the clock zone for wires.

IV . SIMULATION RESULTS OF HYBRID ADDER

In this section, we present the simulation results for hybrid adder.



V . COMPARISON OF DIFFERENT ADDER

Approach	Cell-count	Area ($\mu m \times \mu m$)	Delay (clocks)	Area-Delay Product
Ladner-Fischer4	698	0.87 × 0.71	2	1.24
Ladner-Fischer8	1994	1.67 × 1.06	2 (3/4)	4.86
Ladner-Fischer16	5376	3.29 × 1.46	4 (1/4)	20.41
Ladner-Fischer32	13552	6.56 × 2.02	7 (2/4)	99.38
Ladner-Fischer64	35850	13.4 × 2.81	13 (2/4)	506.43
Hybrid4	475	0.80 × 0.53	1 (3/4)	0.742
Hybrid8	1422	1.34 × 0.83	2 (1/4)	2.5
Hybrid16	3555	2.61 × 1.02	3 (1/4)	8.65
Hybrid32	8946	5.55 × 1.48	5 (2/4)	45.18
Hybrid64	22427	11.6 × 2.18	9 (2/4)	240.65
RCA4 [8]	651	1.67 × 0.72	4 (1/4)	5.11
RCA8 [8]	1499	3.43 × 1.04	8 (1/4)	29.43
RCA16 [8]	3771	6.97 × 1.69	16 (1/4)	191.41
RCA32 [8]	10619	14.0 × 3.01	32 (1/4)	1361.93
RCA64 [8]	33531	28.2 × 5.65	64 (1/4)	10229.69
CLA4 [8]	1575	1.74 × 1.09	3 (2/4)	6.64
CLA8 [8]	3988	3.5 × 1.58	6 (2/4)	35.9
CLA16 [8]	10217	7.02 × 2.21	10 (1/4)	159
CLA32 [8]	25308	14.06 × 3.05	19	814.8
CLA64 [8]	59030	28.2 × 3.73	31 (2/4)	3313.4
CFA4 [16]	371	0.90 × 0.45	1 (2/4)	0.61
CFA8 [16]	789	1.79 × 0.53	2 (2/4)	2.37
CFA16 [16]	1769	3.55 × 0.69	4 (2/4)	11.23
CFA32 [16]	4305	7.09 × 1.03	8 (2/4)	62.07
CFA64 [16]	11681	14.2 × 1.71	16 (2/4)	399.2

VI. MULTIPLIER- ACCUMULATOR UNIT

MAC is composed of an adder, multiplier and an accumulator. Usually adders implemented are Carry-Select or Carry-Save adders, as speed is of utmost importance in DSP One implementation of the adder could be as a quantum dot design of a hybrid addder . The inputs for the MAC are to be fetched from me mory location and fed to the multiplier block of the MAC, which will perform multiplication and give the result to adder which will accumulate the result and then will store the result into a memory location. This entire process is to be achieved in a single clock cycle. In the majority of digital si gnal processing (DSP) applications the critical operations are the multiplication and accumulation. Real-time signal processing requires high speed and high throughput Multiplier-Accumulator (MAC) unit that consumes low power, which is always a key to achieve a high perfor mance digital signal processing system.

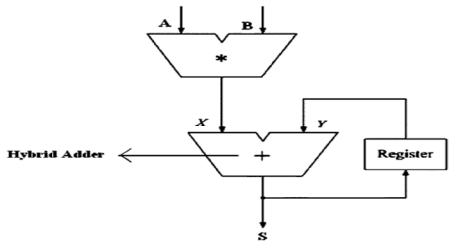


Fig. 8. Basic structure of MAC

VII. CONCLUSION

Many previous designs have claimed to provide high performance,but it consumes more power and the area-delay product had increased.To overcome this problem an efficient design of a hybrid adder is proposed. The designs are based on new results concerning majority logic. The hybrid adder is shown to be particularly well suited to the QCA model.For best improvements,further more the efficient design of hybrid adder is implemented in MAC unit.The MAC unit consist of multiplier and adder.Hybrid adder will be implemented in the adder part of the MAC unit and compare the performance of the previous adder.

REFERENCES

[1] Vikramkumar Pudi and K. Sridharan, "*Efficient Design of a Hybrid Adder in Quantum-Dot Cellular Automata,*"IEEE Trans.,Very Large Scale Integr.(VLSI) Syst., Vol. 19, no. 9, Sep. 2011.

[2] G. W. Hanson, "*Fundamentals of Nanoelectronic,"s* . Englewood Cliffs, NJ: Prentice-Hall, 2008.

[3] I. Koren , "*Computer Arithmetic Algorithms*," . Natick, MA: A.K. Peters Ltd., 2002.

[4] W. Wang, K. Walus, and G. A. Jullien, "*Quantum-dot cellular automata adders,*" in Proc. 3rd IEEE Conf. Nanotechnol. (IEEE-NANO), 2003, pp. 461–464.

[5] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "*A method of majority logic reduction for quantum cellular automata*," IEEE Trans. Nanotechnol., vol. 3, no. 4, pp. 443–450, Dec. 2004.

[6] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "*Performance com parison of quantum-dot cellular automata adders,"* in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS) , 2005, pp. 2522–2526.

[7] H. Cho and E. Swartzlander, "*Modular design of conditional sum adders using quantum-dot cellular automata,*" in Proc. 6th IEEE Conf. Nanotechnol. (IEEE-NANO), 2006, pp. 363–366.

[8] R. Tang, F. Zheng, and Y.-B. Kim, "*QCA-based nano circuits design [adder design example],"* in Proc. IEEE Int. Symp. Circuits Syst., 2005, pp. 252 -2530.

[9] H. Cho and E. E. Swartzlander, "*Adder designs and analyses for quantum-dot cellular automata,*" IEEE Trans. Nanotechnol. , vol. 6, no. 3, pp. 374–383, May 2007.

[10] K. Kim, K. Wu, and R. Karri, "*The robust QCA adder designs using composable QCA building blocks,*" IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 26, no. 1, pp. 176–183, Jan. 2007.

[11] T. J. Dysart and P. M. Kogge, "*Probabilistic analysis of a molecular quantum-dot cellular automata adder,"* inProc. IEEE Int. Symp. Defect Fault-Toler. VLSI Syst., 2007, pp. 478–486.

[12] T. Dysart and P. M. Kogge, "*Analyzing the inherent reliability of mod erately sized magnetic and electrostatic QCA circuits via probabilistic transfer matrices,*" IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 4, pp. 507–516, Apr. 2009.

[13] I. Hanninen and J. Takala, "*Robust adders based on quantum-dot cel lular automata*," in Proc. IEEE Int. Conf. Appl.-Specific Syst., Arch. Processors, 2007, pp. 391–396.

[14] S. Srivastava and S. Bhanja, "Hierarchical probabilistic macromodeling for QCA circuits," IEEE Trans. Comput., vol. 56, no. 2, pp. 174–190, Feb. 2007.

[15] S. Srivastava, S. Sarkar, and S. Bhanja, "Estimation of upper bound of power dissipation in QCA circuits," IEEE Trans. Nanotechnol., vol. 8, no. 1, pp. 116–127, Jan. 2009.

[16] H. Cho and E. E. Swartzlander, "Adder and multiplier designs in quantum-dot cellular automata," IEEE Trans. Comput., vol. 58, no. 6, pp. 721–727, Jun. 2009.

[17] K. Walus, T. Dysart, G. Jullien, and R. Budiman, "QCADesigner: A rapid design and simulation tool for quantum-dot cellular automata," IEEE Trans. Nanotechnol., vol. 3, no. 1, pp. 26–29, Jan. 2004.

[18] R. Zhang, P. Gupta, and N. K. Jha, "Synthesis of majority and minority networks and its applications to QCA, TPL, and SET based nanotechnologies," in Proc. Int. Conf. VLSI Des., 2005, pp. 229–234.