Squarers in QCA Nanotechnology

O. Giannou, H. T. Vergos, and
Computer Engineering & Informatics Dept.,
University of Patras, 26 500, Greece
Email: {ogiannou, vergos}@ceid.upatras.gr

D. Bakalis
Physics Department,
University of Patras, 26 500, Greece
Email: bakalis@physics.upatras.gr

***Abstract*—A novel architecture suitable for designing squaring circuits in Quantum-dot Cellular Automata (QCA) nanotechnology is explored in this manuscript. It consists of a reduced partial product array followed by a Dadda adder tree and a final carry-flow adder. A new full adder (FA) layout is also introduced that is more area efficient than the already proposed ones. The proposed squarers offer an execution delay of $(2n-1)$ clock zones. Layouts of the proposed squarers using multilayer design in the QCAdesigner toolset are presented and their area and delay complexities are compared against previously proposed multipliers.**

I. Introduction

The computation of the square of a binary number can be implemented in hardware by a multiplier having the same value on its two inputs. However, a large number of applications require a very large number of squaring operations and therefore profit from the use of a dedicated squaring circuit, capable of performing hundreds of millions of squaring operations per second. Such applications include Euclidean distance and vector normalization [1]–[3], image compression and pattern recognition [4]–[6], Viterbi decoding, graphic processors [7], polynomial and function evaluation [8]–[10], digital synthesizers [11] and so on. It is therefore no surprise that a significant number of architectures have been presented for the design of efficient dedicated squaring circuits in CMOS technology [3], [12]–[15].

It is expected that the role of CMOS as the dominant VLSI technology will 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. The projected expectations in terms of device density, power dissipation and performance radically necessitate different technologies that provide innovative solutions to integration and computations. To this end, various nanoelectronic devices have been of interest to the research community during the last decade. These include carbon nanotubes, silicon nanowires, resonant tunneling diodes, and others. One of the devices suggested in the literature as an alternative to the traditional CMOS-based technology is the quantum-dot cellular automata (QCA) which is considered to provide new possibilities for computing owing to its unique properties [16], such as that the device used for logic is also used for interconnect. QCA relies on the Coulombic interaction of electrons and its logic states are not stored as voltage levels, but rather as the position of individual electrons. Even though the physical implementation of devices is still being developed, it is necessary to research about efficient QCA circuit architectures. Although the architectures already proposed for CMOS may be translated into QCA implementations by mapping CMOS gates into QCA basic logic gates (the majority, the minority and the inverter are the basic logic elements efficiently implemented in QCA) this does not mean that algorithms that have been optimally implemented in one technology are necessarily the best choice in the new one. To this end, a lot of recent research has focused in efficient circuit design and arithmetic components design in QCA. Examples include parallel adders [17]–[20], serial-parallel and parallel multipliers [21]–[24].

In this paper, we consider the problem of designing fast and area efficient squarers in QCA which has not received any research efforts yet. We propose an architecture based on reducing the partial products array, an adder tree that reduces the partial products in two final summands and a final parallel adder that produces the result. We present experimental results for our squarers based on drawn layouts and compare them against previously proposed parallel multipliers.

The rest of this paper is organized as follows. Section II is a brief overview of QCA and provides the basic notations pertaining to it. The proposed squarers are introduced in Section III. Delay and area results from drawn layouts are given in Section IV. Conclusions are drawn in the last section.

II. QCA overview

A QCA is a square structure of electron wells confining two free electrons. Each cell has four quantum dots which can hold a single electron per dot. Two different positions of the dots are possible; they can be positioned near each vertex of the cell, or close to the medium point of each side, to form, respectively, *cross cells* and *plus cells*. Each of the two excess electrons within each cell can be in any one of the four dots, but due to electric repulsion they can not occupy the same dot and must reside in maximally distant, that is opposed, dots. Therefore, only two polarizations are possible for each case of cells, enabling us to encode a binary value at each. Figs 1(a) and 1(b) show the possible polarizations and the binary encodings commonly adopted for cross and plus cells, respectively. In this paper we assume a nominal cell size of 20 nm by 20 nm. The cell has a width and height of 18 nm and 5-nm-diameter quantum-dots. The cells are placed on a grid with a cell center-to-center distance of 20 nm.

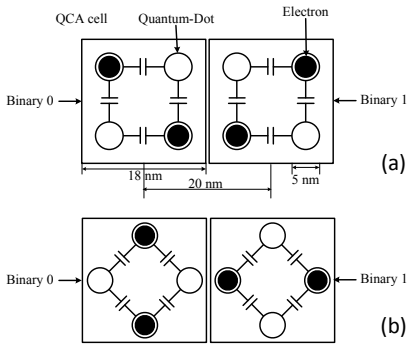


Fig. 1. QCA cells : polarizations and encodings.

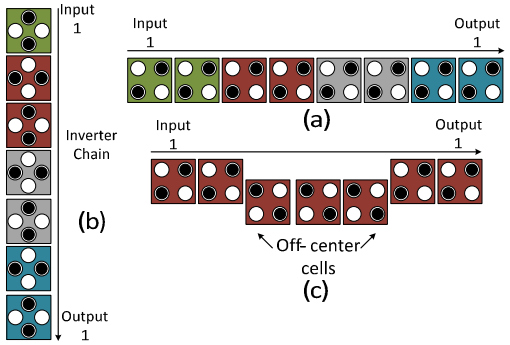


Fig. 2. Wires in QCA.

By carefully choosing the interdot tunneling barrier of the QCA cell, adiabatic switching is accomplished. For switching, the cells are grouped together into pipeline zones and a four phase clocking scheme is used in QCA. In this way, the circuit area is divided into four sections driven by a distinct phase of the clock. In each zone, the clock signal has four states: high-to-low, low, low-to-high, and high. The cell begins computing during the low-to-high state and holds the value during the high state. The cell is released when the clock is in the high-to-low state and inactive during the low state. During each clock cycle, half of the cells are active for signal propagation, while the other half is unpolarized. During the next clock cycle, half of the previous active clock zone is deactivated and the remaining active zone cells trigger the newly activated cells to be polarized. Thus, signals propagate from one clock zone to the next. This clocking scheme allows each zone of cells successively to perform its calculation and then to hold its state by raising the interdot barriers and leads to inherent self latching in QCA.

The basic elements in QCA are the wire, the inverter and the majority / minority gates. A wire is formed by cascading cells horizontally or vertically. Due to Coulombic interactions between adjacent cells a binary value or its complement may propagate from one end (input) of the wire to the other (output). A straight line of cross cells can be used to propagate a binary value (Fig. 2(a)) whereas a line of plus cells is an inverter chain (Fig. 2(b)). Furthermore QCA cells do not have to be in a completely straight line for binary signal propagation. Off center cells can be used. The off center cells in Fig. 2(c) perform a double inversion. Fig. 3 presents the two most popular ways to construct an inverter in QCA. Because

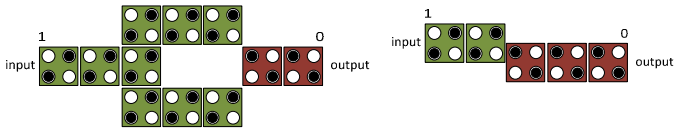


Fig. 3. Inverters in QCA.

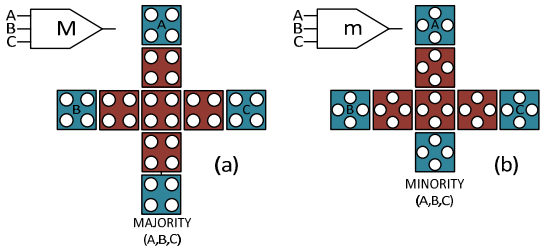


Fig. 4. Structure of a majority (a) and a minority (b) gate.

there are propagation delays between cell-to-cell reactions, there should be a limit on the maximum cell count in a clock zone. This ensures proper propagation and reliable signal transmission. In this paper, a maximum length of 30 cells is considered. The minimum separation between two different signal wires is the width of two cells.

In CMOS technology two different metal layers must be used for routing of two signals when the wires that propagate them cross. A similar approach can be used in QCA, called multilayer crossover. It utilizes several cell layers and uses intermediate layers of vertical cells that interconnect them. Another wire crossing unique to QCA technology, called coplanar wire crossing is also available. Coplanar crossovers can be realized by using the two types of cells (cross and plus) to propagate two signals on the same layer in QCA. In this paper, we use both routing strategies.

Apart from wires and inverters, three-input majority / minority gates serve as the fundamental gates. The structure of the majority and minority gates is similar with the only exceptions being that cross cells are used for the former and plus cells for the latter and that the minority gate requires one cell less. Figs 4(a) and 4(b) show the logic symbols and the layouts of the three-input majority and minority gates, respectively. AND and OR gates can easily be derived by fixing one of the inputs of these gates to zero or one.

Based on the inverters and the majority or minority gates introduced earlier, one can design larger circuits. Circuit diagrams for a full adder (FA) using inverters and majority or minority gates are given in Figs 5(a) and 5(b) respectively. Two layouts with cross cells for the QCA implementation of the first diagram are also presented in Fig. 5(a). The left one has been proposed by [22] while the one on the right presents a more area efficient solution which we propose and use in this paper. The layout of Fig. 5(b) [19], uses plus cells and is also slightly more area efficient than that of [22]. It is noted that all FA cells require a single clock zone to produce their carry output, while two clock zones are required for their sum output and the same holds for the corresponding half adders (HA).

Several architectures already proposed for CMOS parallel

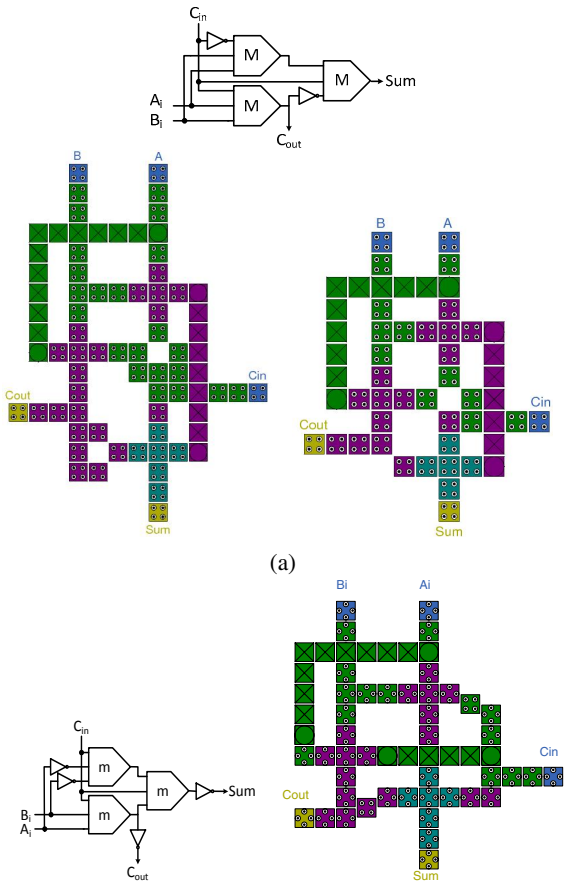


Fig. 5. FA cells in QCA.

adders have also been investigated for implementation in QCA including the ripple carry, carry lookahead, conditional sum, parallel-prefix and hybrid [18], [19], [22], [25], [26] architectures. Due to the available building blocks in QCA, the ripple carry architecture (also known as the carry-flow architecture) offers, apart from small area, the least delay (equal to $n+2$ clock zones for an n-bit parallel adder), for operand sizes less than 16 bits and is also the clear choice among the available adder architectures when the operand bits are not all derived at the same time, such as in the case of the parallel adder used in multipliers and squarers.

Several architectures have also been proposed for designing a multiplier in QCA; some for the serial-parallel multipliers [22], [27] case while others for parallel multipliers [21], [23], [24]. To the best of our knowledge, no architecture has been explored for the design of a squarer in QCA. We fill this gap by proposing a novel architecture in the next Section.

III. Proposed Squarers

Let $A = a_{n-1}a_{n-2}\dots a_1a_0$ denote an n-bit operand. For the clarity of our presentation in the examples below we consider that $n = 8$. A squarer circuit that produces A^2 can be thought as one consisting of three stages :

- the generation stage, which produces the partial product bits,
- the reduction stage, which reduces the partial products in two final addends and
- the final addition stage in which the sum of the two addends is computed.

A. Generation stage

The purpose of the generation stage is twofold; to produce as less partial product bits as possible and to minimize the maximum height of each column of partial product bits. Less partial product bits means small implementation area but is even more important in QCA than in CMOS due to the limitations of the maximum length of cells a wire can have before a clock zone change is required. Minimizing the maximum height of the columns also minimizes the stages of the adder tree required for their reduction.

Two approaches can be followed to minimize the number of partial product bits; to use Booth encoding at the input operand and to fold the partial product bits using mathematical identities. Since Booth encoding does not lead to faster designs [28] we only concentrate below on the second technique. The following identities [12] can be applied to the initial partial product array to derive a folded one whose size is approximately half of the original : $a_ia_i = a_i$ and $a_ia_j + a_ja_i = 2a_ia_j$. The second identity means that we can move pairs of a_ia_j and a_ja_i partial product bits to the next to the left column. Regions marked A and B in the example case in Fig. 6 indicate the initial and the folded product array derived by the application of the above identities. In the general case, after the above identities have been exploited, the square of A takes the form :

$$A^2 = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ia_j2^{i+j} = \sum_{i=0}^{\lfloor\frac{n-1}{2}\rfloor}\sum_{j=i+1}^{n-1} a_ia_j2^{i+j+1} + \sum_{i=0}^{n-1} a_i2^{2i}$$

Further reductions in the partial products array are possible by using identities of precalculated sums such as those proposed in [3], [13], [15]. However, most of these identities are very costly to implement in QCA and therefore we only consider the half-adder relation : $a_i + a_ia_{i-1} = 2a_ia_{i-1} + a_i\overline{a_{i-1}}$. Using this relation we can substitute a pair of a_i and a_ia_{i-1} bits residing in the column with weight 2^{2i} with the $a_i\overline{a_{i-1}}$ bit in the same column and the a_ia_{i-1} bit in the next to the left column, that is, the one with weight 2^{2i+1}. When n is even, the maximum depth of the array occurs at the column with weight 2^{n+1} and the above substitution reduces the depth by one. We propose that the half-adder relation is applied only to pairs of bits residing in the columns with weight 2^2 up to 2^{n+1} (in the example case in the shaded area of region B in Fig. 6) for three reasons :

- each application increases the number of distinct partial products that should be generated. For example, both a_1a_0 and $a_1\overline{a_0}$ need to be generated compared to just a_1a_0 before the application of the relation. Since we limit the length of each wire to a maximum of 30 cells before a new clock zone is used, for forming more partial products

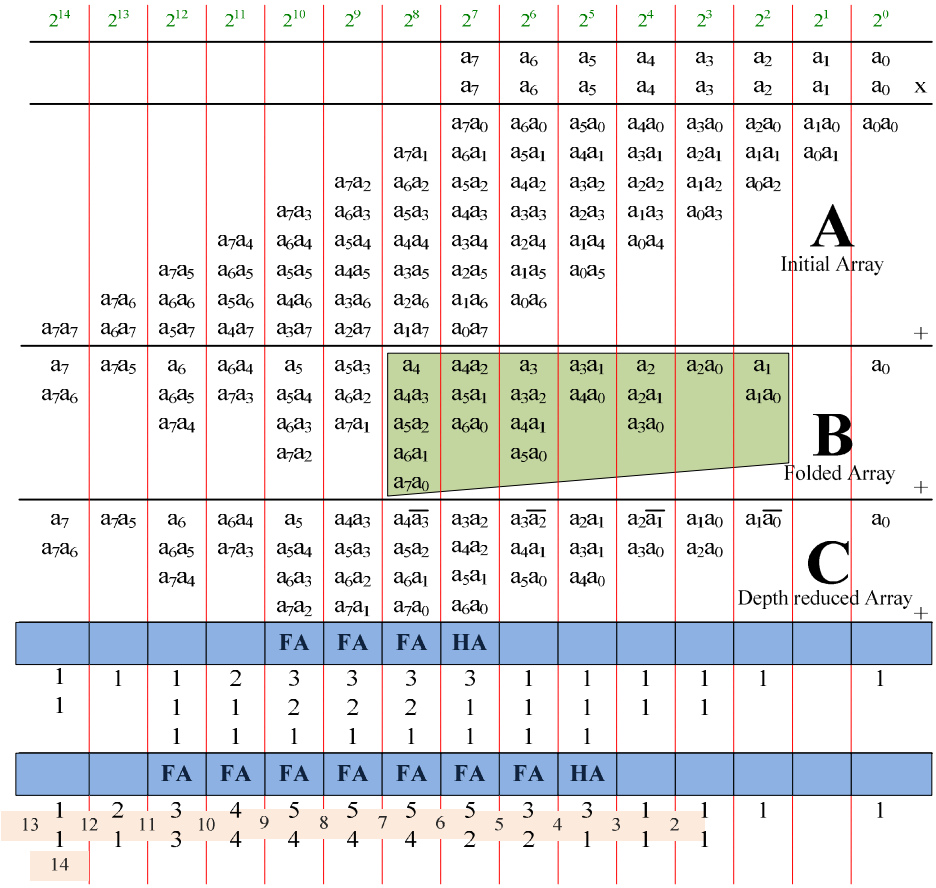


Fig. 6. Derivation of the 8x8 squarer.

more clock zones would be required resulting in partial products produced with delay.

- the application to the column with weight 2^{2n-1} would increase the size of the final adder by one.
- the application to the columns with weight more than 2^{n+1} will not reduce the depth of the array.

Region C in Fig. 6 presents the final form of the partial products in the example case.

B. Reduction stage

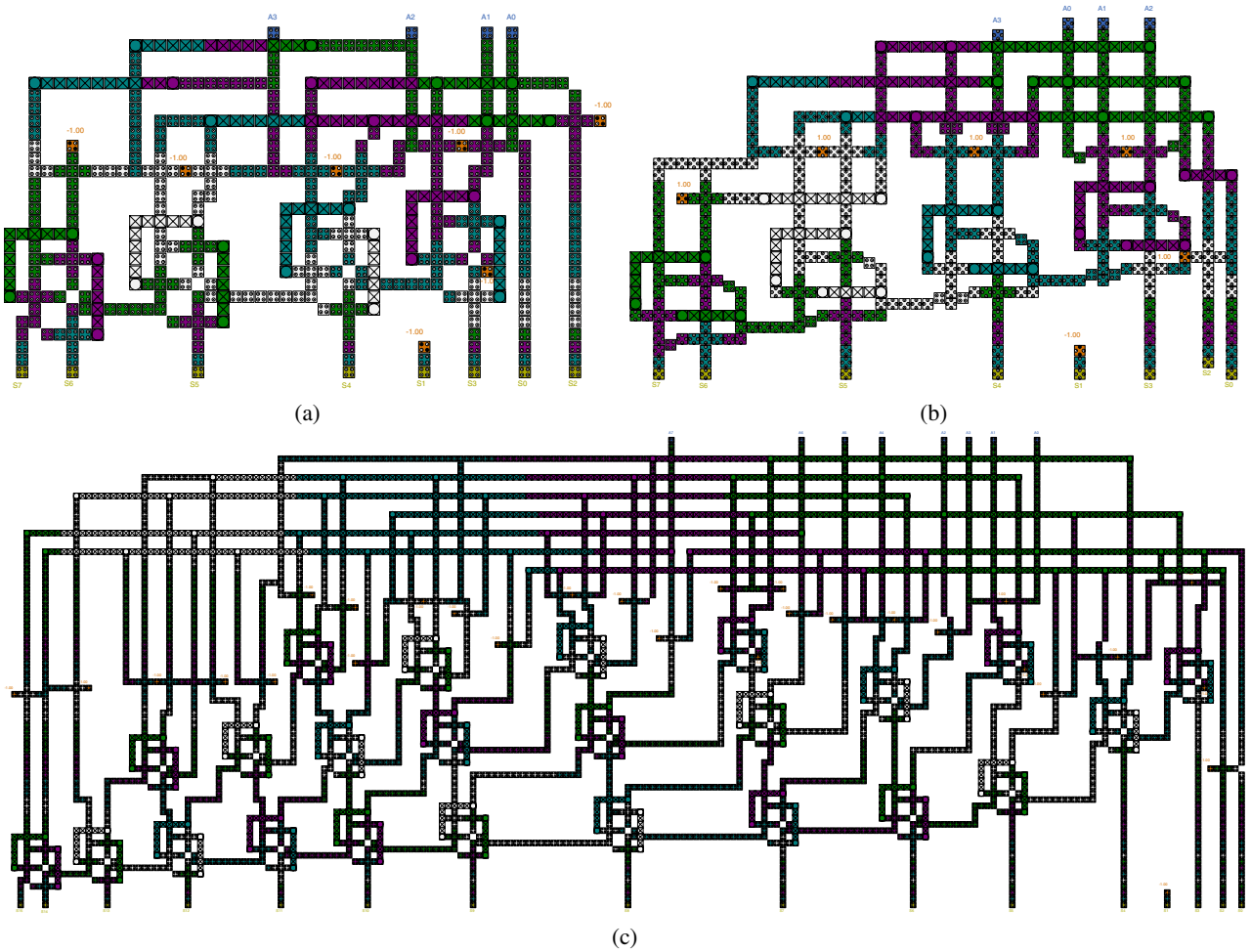
For reducing the partial products into two final addends an adder array or an adder tree can be used. Since an adder tree uses fewer stages we adopt a tree designed according to the Dadda proposal [29] for our squarers. A Wallace [30] tree for partial product reduction was also considered as an alternative solution. Although Wallace's strategy (combine partial product bits at the earliest opportunity) leads to a slightly narrower final adder, it also has a more complex reduction tree leading to delayed derivation of the final addends. Dadda's method performs combining as late as possible, while maintaining the same stages in the adder tree.

The two shaded lines following region C in Fig. 6 indicate the required HA and FA in every column of each of the two stages of the adder tree. A total of 10 FAs and 2 HAs are required in the example case. The numbers in between represent the number of clock zones required for deriving the inputs of the second stage of the adder tree. For example, the FA in the column with weight 2^7 in the second stage of the tree receives three inputs that are computed after 3, 1 and 1 clock zones, so its sum output is available after 5 clock zones, while its carry after 4 clock zones.

C. Final Addition Stage

The reduction stage provides two addends that should be driven to a $(2n-4)$-bit adder. We adopt the simple ripple-carry (also known as carry-flow in QCA) architecture for it in the proposed squarers which offers a delay of $(2n-2)$ clock zones. Obviously, this starts adding its least significant bit operands (those with weight 2^3) right after their production and not after the completion of the reduction stage.

The last two lines of Fig. 6 indicate the clock zones in which the bits of the final addends are available, while the smaller lightly shaded numbers between them indicate the clock zones required for producing each intermediate carry of the final adder. Interestingly enough, the reduction stage produces all addends' bits either before or at the same clock zone with the intermediate carries at each FA, indicating that the reduction stage does not contribute at all to the delay of the squarer which is only determined by the clock zones required by the final adder and for the partial products formation. By examining all practical squarer cases (from $n = 4$ up to $n = 64$) we verified that the same holds for all of them. Then, by assuming that all partial products can be produced within 1 clock zone we conclude that the proposed n-bit



(a) (b)

(c)

Fig. 7. Layouts of the proposed 4-bit squarer with cross (a) and plus (b) cells and the 8-bit squarer with cross cells.

squarer offers a theoretical delay of $(2n-1)$ clock zones. However, the actual layout implementations may deviate from this theoretical minimum, because :

- of the limits of the length of each wire dictating a shift to a new clock zone. Therefore, a carefully designed layout should first concentrate on producing the partial products for columns with weight 2^3 up to 2^{n+2} in the earliest clock zone possible, while the production of the rest can not delay the squarer since their columns do not have a critical depth.
- in the analysis above, we assume that a FA or a HA will always be able to combine the signals derived earlier. For example, if a FA is used to combine 3 of the 5 signals in a column with arrival times of 5, 4, 2, 1 and 1 clock zones, we assume that this will combine the last three signals for minimum delay. However, in actual layout this may not be always possible.

IV. Results & Comparisons

The presented results are based on layouts drawn with QCADesigner [31] using two cell layers and a maximum of 30 cells before a shift to a new clock zone. The following parameters are used for a bistable approximation: 128000 samples, 0.00001 convergence tolerance, 54 up to 65nm radius

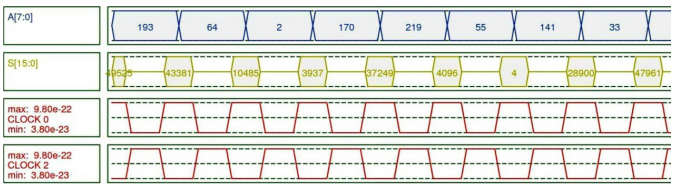


Fig. 8. 8-bit squarer simulation waveforms

effect, 12.9 relative permittivity, $9.8e^{-22}$ clock high, $3.8e^{-23}$ clock low, 2 clock amplitude factor, 11.5 layer separation, 100 maximum iterations per sample.

Figs. 7(a) and 7(b) present the designed multi-layer layouts for the proposed 4-bit squarers using plus and cross cells, respectively. Inverters and majority or minority gates are used for forming the required partial product bits followed by HA and FAs that perform the required additions. Both these designs offer a delay of 7 clock zones with the cross cells design requiring less area for its implementation due to the use of the newly introduced FA layout and because the number of inversions need to be carefully considered in every connection in the plus cells design case. Fig. 7(c) presents the drawn layout for the proposed 8-bit squarer using cross cells while a snapshot of vector table simulation waveforms for it is presented in Fig. 8. A delay of 15 clock zones is offered in

TABLE I
DELAY AND AREA RESULTS

Design	Delay (Clock zones)	Area (μm^2)	Cells
4×4 Array I multiplier	14	5.18	2,956
4×4 Array II multiplier	14	6.02	3,738
4×4 Wallace multiplier	10	7.39	3,295
4×4 Dadda multiplier	12	7.51	3,384
8×8 Array I multiplier	30	22.57	13,839
8×8 Array II multiplier	30	21.49	15,106
8×8 Wallace multiplier	44	87.47	33,894
8×8 Dadda multiplier	47	92.69	34,903
8×8 QM Wallace multiplier	36	82.18	26,499
8×8 QM Dadda multiplier	38	82.19	26,973
4-bit squarer with cross cells	7	0.53	552
4-bit squarer with plus cells	7	0.63	628
8-bit Squarer with cross cells	15	5.99	4,668
8-bit Squarer with plus cells	15	6.3	4,780

the 8-bit squarer case.

The area and the delay of the proposed squarers from the layouts drawn in QCAdesigner are summarized in Table I. No other squarer architecture for QCA has been reported. Table I also includes results of previously proposed multipliers [23], [24] in QCA for which coplanar wire crossings on a single cell layer were only used. As expected, the results indicate that an n-bit squarer is both a smaller and a faster circuit than a $n \times n$ multiplier. This means that although a multiplier can be used for computing the square of a number, such an approach is not efficient for applications with a high rate of squaring operations.

V. CONCLUSIONS

An architecture for designing efficient squarers in QCA nanotechnology was described in this paper. It consists of a reduced partial products array, an adder tree to reduce the array of the partial products into two final summands and a final adder for attaining the result. The proposed n-bit squarer theoretically offers an execution delay of $2n - 1$ clock zones, which has been also achieved by layout experiments performed with QCAdesigner tool.

REFERENCES

[1] R. Jain, A. Madisetti, and R. L. Baker, "An Integrated Circuit Design for Pruned Tree-Search Vector Quantization Encoding with an Off-Chip Controller," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 2, pp. 147–158, June 1992.

[2] Lucent Technologies, *DSP 1628 Datasheet*. Murray Hill, NJ., 1997.

[3] R. K. Kolagotla, W. R. Griesbach, and H. R. Srinivas, "VLSI Implementation of 350MHz 0.35 micron 8 Bit Merged Squarer," *Electronics Letters*, vol. 34, no. 1, pp. 47–48, January 1998.

[4] J. Pihl and E. J. Aas, "A Multiplier and Squarer Generator for High Performance DSP Applications," in *Proc. of* 39^{th} *Midwest Symposium on Circuits and Systems*, vol. I, 1996, pp. 109–112.

[5] J.-T. Yoo, K. Smith, and G. Gopalakrishnan, "A Fast Parallel Squarer Based on Divide-and-Conquer," *IEEE J. Solid-State Circuits*, vol. 32, no. 6, pp. 909 –912, June 1997.

[6] Y. Y. Fengqi and A. N. Wilson, "Multirate Digital Squarer Architectures," in *Proc. of the* 8^{th} *IEEE International Conference on Electronics, Circuits & Systems*, 2001, pp. 177–180.

[7] D. DeCaro, N. Petra, and A. Strollo, "High-Performance Special Function Unit for Programmable 3-D Graphics Processors," *IEEE Trans. Circuits Syst.*, vol. 56, no. 9, pp. 1968–1978, Sept 2009.

[8] A. A. Liddicoat and M. J. Flynn, "Parallel Square and Cube Computations," in *Proc. of the IEEE* 34^{th} *Asilomar Conference on Signals, Systems and Computers*, 2000, pp. 1325 –1329 vol.2.

[9] J.-A. Pineiro, S. Oberman, J.-M. Muller, and J. Bruguera, "High-Speed Function Approximation using a Minimax Quadratic Interpolator," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 304–318, March 2005.

[10] A. Strollo, D. DeCaro, and N. Petra, "Elementary Functions Hardware Implementation using Constrained Piecewise-Polynomial Approximations," *IEEE Trans. Comput.*, vol. 60, no. 3, pp. 418–432, March 2011.

[11] A. Ashrafi, R. Adhami, and A. Milenkovic, "A Direct Digital Frequency Synthesizer based on the Quasi-Linear Interpolation Method," *IEEE Trans. Circuits Syst.*, vol. 57, no. 4, pp. 863–872, April 2010.

[12] T. C. Chen, "A Binary Multiplication Scheme Based on Squaring," *IEEE Trans. Comput.*, vol. C-20, no. 6, pp. 678–680, June 1971.

[13] K. Wires, M. Schulte, L. Marquette, and P. Balzola, "Combined Unsigned and Two's Complement Squarers," in *Proc. of the* 33^{rd} *Asilomar Conf. on Signals, Systems, and Computers*, 1999, pp. 1215–1219 vol.2.

[14] A. Strollo and D. DeCaro, "Booth Folding Encoding for High Performance Squarer Circuits," *IEEE Trans. Circuits Syst. II*, vol. 50, no. 5, pp. 250–254, May 2003.

[15] K. J. Cho and J. G. Chung, "Parallel Squarer Design using Pre-Calculated Sums of Partial Products," *Electronics Letters*, vol. 43, no. 25, pp. 1414–1416, December 6 2007.

[16] C. Lent and P. Tougaw, "A Device Architecture for Computing with Quantum Dots," *Proceeding of the IEEE*, vol. 85, no. 4, pp. 541–557, April 1997.

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

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

[19] F. Bruschi, F. Perini, V. Rana, and D. Sciuto, "An Efficient Quantum-Dot Cellular Automata Adder," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, March 2011, pp. 1–4.

[20] S. Sayedsalehi, M. Moaiyeri, and K. Navi, "Novel Efficient Adder Circuits for Quantum-Dot Cellular Automata," *Journal of Computational and Theoretical Nanoscience*, vol. 8, no. 9, pp. 1769–1775, 2011.

[21] I. Hänninen and J. Takala, "Binary Multipliers on Quantum-Dot Cellular Automata," *Facta Universitatis - Series: Electronics and Energetics*, vol. 20, no. 3, pp. 541–560, 2007.

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

[23] S. Kim and E. Swartzlander, "Parallel multipliers for Quantum-Dot Cellular Automata," in *IEEE Nanotechnology Materials and Devices Conf. (NMDC)*, June 2009, pp. 68–72.

[24] ——, "Multipliers with Coplanar Crossings for Quantum-Dot Cellular Automata," in 10^{th} *IEEE Conf. on Nanotechnology*, 2010, pp. 953–957.

[25] H. Cho and E. Swartzlander, "Modular Design of Conditional Sum Adders Using Quantum-dot Cellular Automata," in 6^{th} *IEEE Conf. on Nanotechnology*, June 2006, pp. 363–366.

[26] V. Pudi and K. Sridharan, "Low Complexity Design of Ripple Carry and Brent-Kung Adders in QCA," *IEEE Trans. Nanotechnol.*, vol. 11, no. 1, pp. 105–119, Jan. 2012.

[27] K. Walus, G. Jullien, and V. Dimitrov, "Computer Arithmetic Structures for Quantum Cellular Automata," in *Proc. of the* 37^{th} *Asilomar Conf. on Signals, Systems and Computers*, Nov. 2003, pp. 1435–1439, Vol.2.

[28] I. Hänninen and J. Takala, "Radix-4 Recoded Multiplier on Quantum-Dot Cellular Automata," in *Proc. of the* 9^{th} *Int. Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 118–127.

[29] L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, 1965.

[30] C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Trans. Electron. Comput.*, vol. EC-13, pp. 14–17, 1964.

[31] 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. 26–31, March 2004.

Back to Index