Serial Parallel Multiplier Design in Quantum-dot Cellular Automata

Heumpil Cho
Qualcomm, Inc.
5775 Morehouse Dr.
San Diego, California 92121
Email: hpcho@qualcomm.com

Earl E. Swartzlander, Jr.
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas 78712
Email: eswartzla@aol.com

Abstract

An emerging nanotechnology, quantum-dot cellular automata (QCA), has the potential for attractive features such as faster speed, smaller size, and lower power consumption than transistor based technology. Quantum-dot cellular automata has a simple cell as the basic element. The cell is used as a building block to construct gates, wires, and memories. Several adder designs have been proposed, but multiplier design in QCA is a rather unexplored research area. This paper utilizes the QCA characteristics to design serial parallel multipliers. Two types of serial parallel multipliers are designed and simulated with several different operand sizes. Those designs are compared in terms of complexity, area, and latency. The serial parallel multipliers have simple and regular structures.

1. Introduction

Current transistor based semiconductor devices are becoming resistent to scaling. Due to the decreasing supply voltage and increasing threshold voltage, the power consumption from leakage current is one of the big challenges of transistor circuits. Nanotechnology is an alternative to these problems and the ITRS report [1] summarizes possible technology solutions. Quantum-dot cellular automata (QCA) is one of the attractive alternatives. Since QCAs were introduced in 1993 [2], several experimental devices have been developed [3–7]. Recent papers show that QCAs can achieve high density [8], fast switching speed [9], and room temperature operation [5, 10].

ALU design is one of the fundamental circuit design issues and several adder designs in QCA have been proposed [11–13], but multiplier design has not been widely considered by QCA designers. There is a QCA multiplier design in [14, 15], but it is not an optimized design. Previous adder designs show that complex designs generally incur longer delays in QCA, so a simple structure is a good choice for the starting point. A large word size parallel multiplier has quite a complex structure. Due to the complex wiring, parallel multipliers in QCA are likely to incur long delays. This paper investigates the serial parallel multiplier. Based on FIR filter equations, the serial parallel multiplication equation is derived and using QCA characteristics, optimized serial parallel multipliers are presented. The final designs show simple and regular structures with an attractive bit slice structure.

The paper is organized as follows. In Section 2, the background of QCA technology and the design approaches are presented. Section 3 shows the algorithmic design of multiplication networks based on filter networks and Section 4 discusses multiplier implementation for QCA circuits. Analyses of simulation results and comparisons follow in Section 5 and conclusions are presented in Section 6.

2. QCA design schemes

2.1. QCA cell

A quantum-dot cellular automata (QCA) is a square nanostructure of electron wells confining free electrons. Each cell has four quantum dots which can hold a single electron per dot. The four dots are located at the corners of the cell and only two electrons are injected into a cell. By the clocking mechanism, the electrons can tunnel through to neighboring cells during the clock transition by the interaction between electrons. A high potential barrier at the settled clock signal locks the state and results in a local polarization which is determined by Coulombic repulsion. The two electrons reside in opposite corners so that two polarizations are possible as seen in Fig. 1. Those two binary states can be used to make QCA cell a storage cell, a computing cell, or a wire.

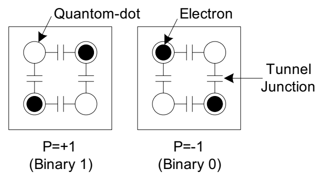


Figure 1. Basic QCA cell and two possible polarizations

2.2. Signal flow

A series of QCA cells act like a wire. An illustration of a QCA wire is shown in Fig. 2. During each clock cycle, half of the wire is active for signal propagation, while the other half is stable. 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.

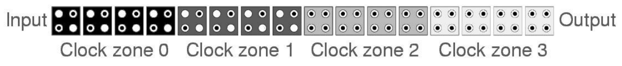


Figure 2. QCA wire

2.3. Clock zones

The circuit area is divided into four sections and they are driven by four phase clock signals. As shown in Fig. 3, there is a 90° phase shift from one section to the next. In each clock zone, the clock signal has four states: high-to-low, low, low-to-high, and high. The cell begins computing during the high-to-low state and holds the value during the low state. The cell is released when the clock is in the low-to-high state and inactive during the high state.

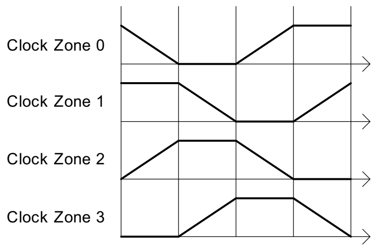


Figure 3. QCA clock zones

The cells in each clock zone behave like a single latch. To be used as a memory cell, a loop of the cells is needed, in which a series of clock zones are used. Because of the signal flow control and synchronization, QCA naturally accepts pipeline designs.

2.4. Logic gates

Logic gates are required to build arithmetic circuits. In QCA, inverters and three-input majority gates serve as the fundamental gates. The governing equation for the majority gate is $M(a, b, c) = ab + bc + ca$. Fig. 4 shows the gate symbols and their layouts. Two input AND and OR gates can be implemented with 3 input majority gates by setting one input to a constant. With ANDs, ORs, and inverters, any logic function can be realized.

$$\begin{aligned} a \cdot b &= M(a, b, 0) \\ a + b &= M(a, b, 1) \end{aligned} \tag{1}$$

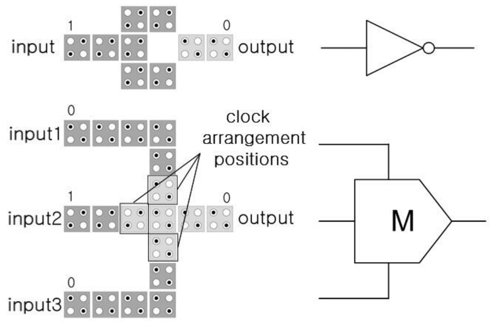


Figure 4. QCA inverter and majority gate

2.5. Design rules

The cell is assumed to have a width and height of $18nm$ and $5nm$ diameter quantum-dots. The cells are placed on a grid with a cell center-to-center distance of $20nm$. Thus the cell size can be defined as $20nm$ for the nominal design.

Because there are propagation delays between cell-to-cell reactions, there should be a limit on the maximum cell count in a clock zone. This insures proper propagation and reliable signal transmission. If there is no restriction on the maximum length, the design might have fewer clock cycles, but due to the increased propagation delays, the operating frequency would be reduced.

From a physical design perspective, there are several circuit operation issues. Long span wires are vulnerable to noise and possibly signal back propagation. The current QCA technology doesn't specifically set the possible operating frequency and actual propagation delays. Thus the maximum cell count can be set as a design parameter. In this paper, 15 cells maximum length is chosen empirically. That was determined to provide freedom to make possible wire routes and a reasonable propagation length for each clock zone. The minimum separation between two different signal wires is the width of two cells.

Multi-layer crossovers are used for wire crossings in this paper. They use more than one layer of cells like a bridge. An example of a multi-layer wire crossing is shown in Fig. 5. The multi-layer crossover design is straightforward although there are questions about how it can be realized in practice, since it requires two overlapping active layers with via connections.

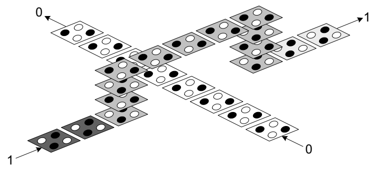


Figure 5. Layout of multi-layer wire crossing

2.6. Simulations

For circuit layout and functionality checking, a simulation tool for QCA circuits, QCADesigner [16], is used. This tool allows users to do a custom layout and then verify QCA circuit functionality by simulations. It includes two different simulation engines such as a bistable approximation and a coherence vector.

3. Algorithmic design

3.1. Filter networks

To consider the multiplication of two numbers, start with a FIR filter example [17]. The filter output is defined by Equation (2).

$$y_i = \sum_{k=0}^{N-1} b_k x_{i-k} \tag{2}$$

Let Z^{-1} be the one cycle delay operator such that $Z^{-1}x_i = x_{i-1}$. Z^0 is defined to be the unit operator and Z^{-n} is defined by $Z^{-n} = Z^{-1}Z^{-n+1}$. The characteristics are as follows.

1. $Z^{-n}x_i = x_{i-n}$.
2. $Z^{-1}F(x) = F(Z^{-1}x)$.
3. If C is time invariant, $Z^{-1}C = C$.

$$\begin{aligned} y_i &= \sum_{k=0}^{N-1} b_k x_{i-k} \\ &= \sum_{k=0}^{N-1} b_k Z^{-k} x_i \\ &= \left(\sum_{k=0}^{N-1} b_k Z^{-k} \right) x_i \end{aligned} \tag{3}$$

Equation (3) can be implemented by the network shown in Fig. 6. The circles in the figure with the b_i's represent multiplication by the constants written inside them and $\bigoplus$ means the addition of two inputs. Data x_i, b_i and y_i are words of arbitrary size.

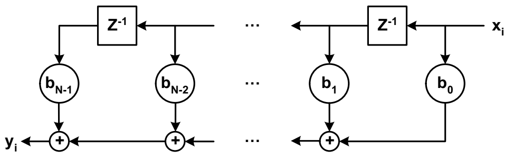


Figure 6. FIR filter network

To use a pipeline design, both upper and lower signal lines include the same additional delay units. Assume that $Z^{-\frac{1}{4}}$ is possible and apply the $Z^{-\frac{1}{2}}$ delay element to each section with upper and lower lines. Equation (4) proves that Fig. 7 gives the correct filter output result with $N/2$ cycle delays.

$$\begin{aligned} &\text{Pipelined FIR filter output} \\ = \; & Z^{-\frac{1}{2}}\Bigg(b_{N-1}Z^{-\frac{3}{2}(N-1)} + Z^{-\frac{1}{2}}\Big(b_{N-2}Z^{-\frac{3}{2}(N-2)} + \\ & \cdots + Z^{-\frac{1}{2}}\left(b_0 Z^0\right)\Big)\Bigg)x_i \\ = \; & Z^{-\frac{N}{2}}b_{N-1}Z^{-(N-1)}x_i + Z^{-\frac{N}{2}}b_{N-2}Z^{-(N-2)}x_i + \\ & \cdots + Z^{-\frac{N}{2}}b_0 x_i \\ = \; & Z^{-\frac{N}{2}}\left(\sum_{k=0}^{N-1} b_k Z^{-k}\right)x_i \\ = \; & Z^{-\frac{N}{2}}y_i \end{aligned} \tag{4}$$

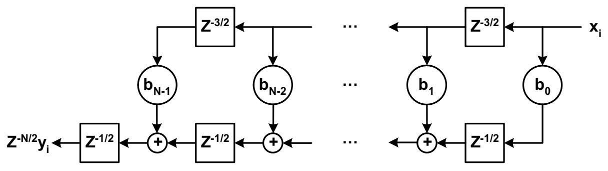


Figure 7. Pipelined FIR filter network

3.2. Multiplication networks

These relations can be applied to multiplication. Assume an unsigned number system. Fig. 8 illustrates the bit product matrix for an unsigned multiplication. In this case, only one digit calculations are used in the network, so all additions and multiplications use binary computations. A one digit multiplication is represented by a logical AND and a one digit addition corresponds to a full adder. The main difference between the FIR filter and the multiplication network is the handling of the carry-out of the adder. The filter networks internally use carry flow, but the multiplication network needs distinct signal flows, so the network for multiplication should be changed accordingly from the filter network.

$$
\begin{array}{ccccccccccc}
 & & & & & & a_{N-1} & a_{N-2} & \cdots & a_1 & a_0 \\
 & & & & & \times & b_{N-1} & b_{N-2} & \cdots & b_1 & b_0 \\
\hline
 & & & & & & a_{N-1}b_0 & a_{N-2}b_0 & \cdots & a_1b_0 & a_0b_0 \\
 & & & & & a_{N-1}b_1 & a_{N-2}b_1 & \cdots & a_1b_1 & a_0b_1 & \\
 & & & & \vdots & \vdots & \cdots & \vdots & \vdots & & \\
 & & a_{N-1}b_{N-2} & a_{N-2}b_{N-2} & \cdots & a_1b_{N-2} & a_0b_{N-2} & & & & \\
 & a_{N-1}b_{N-1} & a_{N-2}b_{N-1} & \cdots & a_1b_{N-1} & a_0b_{N-1} & & & & & \\
\hline
p_{2N-1} & p_{2N-2} & p_{2N-3} & \cdots & p_N & p_{N-1} & p_{N-2} & \cdots & p_2 & p_1 & p_0
\end{array}
$$

Figure 8. Bit product matrix for unsigned multiplication

Let (a_i, b_i) be the multiplicand and multiplier pair and p_i be the product sum of the bit position i. Bits a_i and p_i correspond to words x_i and y_i of the filter example. The position i is considered as the input applied at time i. Define the sum and carry-out of full adder at i^{th} time and j^{th} location as (s_{ij}, c_{ij}) when $0 \leq i \leq 2N-1$ and $0 \leq j \leq N-1$ where j is numbered from right to left. Assume that the sum generation takes at least $Z^{-\frac{1}{2}}$ and the carry generation takes at least $Z^{-\frac{1}{4}}$. Even though Figs. 6 and 7 ignored the zeroth full adder, the derivation includes that adder. The implementation can be done in two ways. Equations (5) and (6) represent the two alternatives.

$$
\begin{aligned}
(s_{ij}, c_{ij}) &= \text{Add}\left(b_j Z^{-\frac{7}{4}j} a_i, Z^{-\frac{3}{4}} s_{i(j-1)}, Z^{-\frac{1}{4}} c_{i(j+1)}\right) \\
&= \text{Add}\left(b_j a_{i-\frac{7}{4}j}, s_{(i-\frac{3}{4})(j-1)}, c_{(i-\frac{1}{4})(j+1)}\right) \quad (5) \\
(s_{ij}, c_{ij}) &= \text{Add}\left(b_j Z^{-\frac{3}{2}j} a_i, Z^{-\frac{1}{2}} s_{i(j-1)}, Z^{-1} c_{ij}\right) \\
&= \text{Add}\left(b_j a_{i-\frac{3}{2}j}, s_{(i-\frac{1}{2})(j-1)}, c_{(i-1)j}\right) \qquad (6)
\end{aligned}
$$

Equation (5) sends the carry-out in a backward direction with minimum delay. Instead, Equation (6) uses a feedback loop to the adder itself using a one clock delay unit. Both handle the carry-out correctly carrying it to the higher bit calculation. Call them a carry shift multiplication (CSM) and a carry delay multiplication (CDM), respectively. Figs. 9 and 10 show the network diagrams based on these equations. In view of the flow direction between input and output, they are named right-to-left (RtL) networks. The angled output line of a full adder is the carry-out. The CSM design is optimized for minimum delay of the carry shift while the CDM design is optimized to minimize the latency of the output.

In this realization, the minimum latency from the first input to first output is either $3N/4$ or $N/2$ cycles. Going to Fig. 6 and redirecting the output to the right side, which

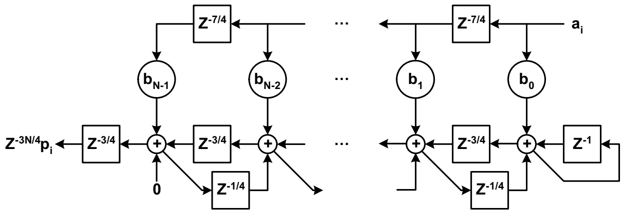


Figure 9. Right-to-left (RtL) carry shift multiplication network

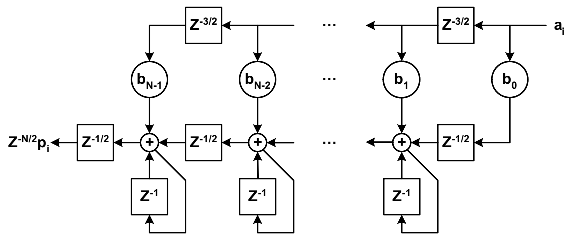


Figure 10. Right-to-left (RtL) carry delay multiplication network

is the same side to the input, Fig. 11 shows the redirection graph and for a pipeline design, it can be redrawn as shown in Fig. 12 by use of Equation (7).

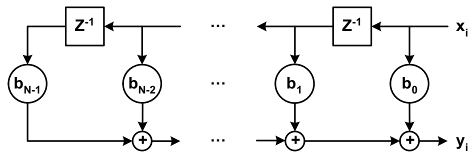


Figure 11. Redirected FIR filter network

$$
\begin{aligned}
Z^{-\frac{1}{2}} y_i &= Z^{-\frac{1}{2}} \left(\sum_{k=0}^{N-1} b_k Z^{-k} \right) x_i \\
&= Z^{-\frac{1}{2}} \left(\sum_{k=0}^{N-1} b_k Z^{-\frac{k}{2}} Z^{-\frac{k}{2}} \right) x_i \\
&= Z^{-\frac{1}{2}} \left(\sum_{k=0}^{N-1} Z^{-\frac{k}{2}} b_k Z^{-\frac{k}{2}} \right) x_i \qquad (7)
\end{aligned}
$$

Finally, Fig. 12 is a network design comparable to Fig. 7. The main difference is that there is a much smaller latency from the first input to the first output. Based on Fig. 12, the multiplication networks are represented by Equations (8) and (9) according to the carry out handling. Figs. 13 and 14 show respective network implementations. Likewise, the networks are called a right-to-right (RtR) networks and by

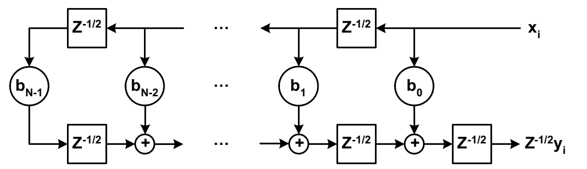


Figure 12. Redirected pipelined FIR filter network

the carry flow, they are distinguished as either carry shift multiplication (CSM) or carry delay multiplication (CDM) networks. The CSM design has the minimum delay of the carry shift and the CDM design has the minimum latency to the output.

$$\begin{aligned}(s_{ij}, c_{ij}) &= \text{Add}\left(b_j Z^{-\frac{1}{4}j} a_i, Z^{-\frac{3}{4}} s_{i(j+1)}, Z^{-\frac{1}{4}} c_{i(j-1)}\right) \\ &= \text{Add}\left(b_j a_{i-\frac{1}{4}j}, s_{(i-\frac{3}{4})(j+1)}, c_{(i-\frac{1}{4})(j-1)}\right) \quad (8)\end{aligned}$$

$$\begin{aligned}(s_{ij}, c_{ij}) &= \text{Add}\left(b_j Z^{-\frac{1}{2}j} a_i, Z^{-\frac{1}{2}} s_{i(j+1)}, Z^{-1} c_{ij}\right) \\ &= \text{Add}\left(b_j a_{i-\frac{1}{2}j}, s_{(i-\frac{1}{2})(j+1)}, c_{(i-1)j}\right) \quad (9)\end{aligned}$$

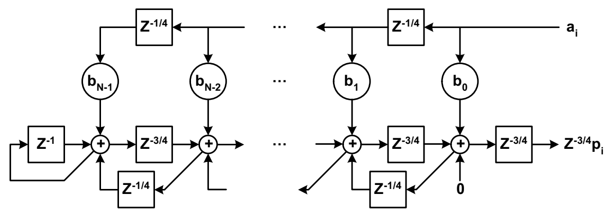


Figure 13. Right-to-right (RtR) carry shift multiplication network

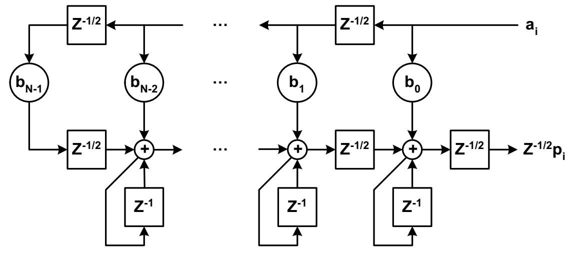


Figure 14. Right-to-right (RtR) carry delay multiplication network

4. Multiplier implementation

4.1. Multiplication networks for QCA

The QCA circuit has a four phase clock and the circuit areas are divided into four clock zones. One clock zone delay is denoted by the D^{-1} operator, which corresponds to a quarter cycle. That is, $D^{-4} = Z^{-1}$. Based on the QCA circuit characteristics, one clock zone delays a quarter clock, so that delay matches to one D^{-1} operation. Assume a logical AND operation delays by one D^{-1} operation and one full adder sum and carry-out computation from the input by D^{-2} and D^{-1} operations, respectively. Wires also take some delay based on the wire length. After incorporating these characteristics, the filter networks are redrawn as Figs. 15 and 16. Delay amounts in upper and lower signal flows in either case are chosen to make one clock cycle differences between the adjacent paths.

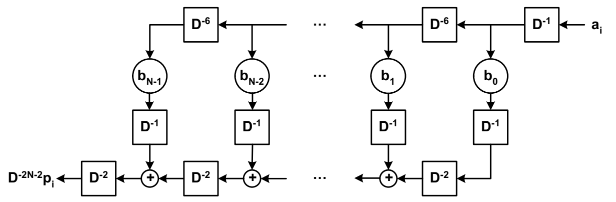


Figure 15. FIR filter network for QCA

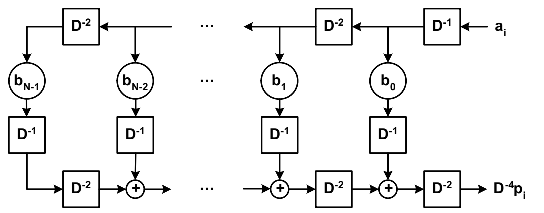


Figure 16. Redirected FIR filter network for QCA

From the filter network examples, the multiplier networks for QCA are drawn. Based on Equations (5-6) and (8-9), Equations (10-11) and (12-13) are rewritten for QCA multiplication. The previous figures are modified by the carry flow options. The serial multiplier can be implemented with four different options as in Figs. 17, 18, 19, and 20.

$$\begin{aligned}(s_{ij}, c_{ij}) &= \text{Add}\left(b_j D^{-7j-2} a_i, D^{-3} s_{i(j-1)}, D^{-1} c_{i(j+1)}\right) \\ &= \text{Add}\left(b_j a_{i-7j-2}, s_{(i-3)(j-1)}, c_{(i-1)(j+1)}\right) \quad (10)\end{aligned}$$

$$\begin{aligned}(s_{ij}, c_{ij}) &= \text{Add}\left(b_j D^{-6j-2} a_i, D^{-2} s_{i(j-1)}, D^{-4} c_{ij}\right) \\ &= \text{Add}\left(b_j a_{i-6j-2}, s_{(i-2)(j-1)}, c_{(i-4)j}\right) \quad (11)\end{aligned}$$

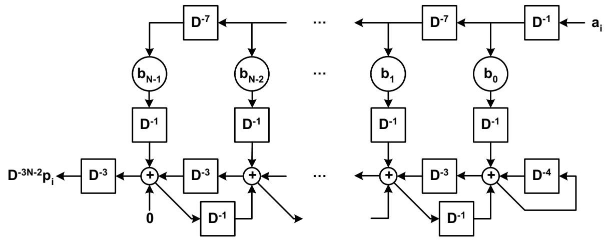


Figure 17. RtL CSM network for QCA

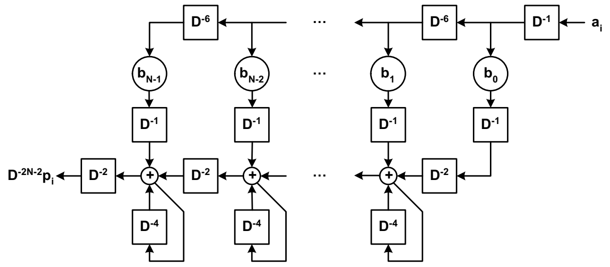


Figure 18. RtL CDM network for QCA

$$
\begin{aligned}
(s_{ij}, c_{ij}) &= \text{Add}\left(b_j D^{-j-2} a_i, D^{-3} s_{i(j+1)}, D^{-1} c_{i(j-1)}\right) \\
&= \text{Add}\left(b_j a_{i-j-2}, s_{(i-3)(j+1)}, c_{(i-1)(j-1)}\right) \quad (12) \\
(s_{ij}, c_{ij}) &= \text{Add}\left(b_j D^{-2j-2} a_i, D^{-2} s_{i(j+1)}, D^{-4} c_{ij}\right) \\
&= \text{Add}\left(b_j a_{i-2j-2}, s_{(i-2)(j+1)}, c_{(i-4)j}\right) \quad (13)
\end{aligned}
$$

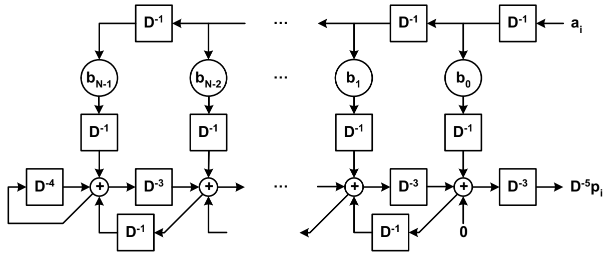


Figure 19. RtR CSM network for QCA

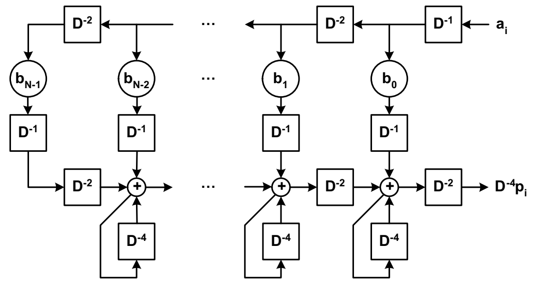
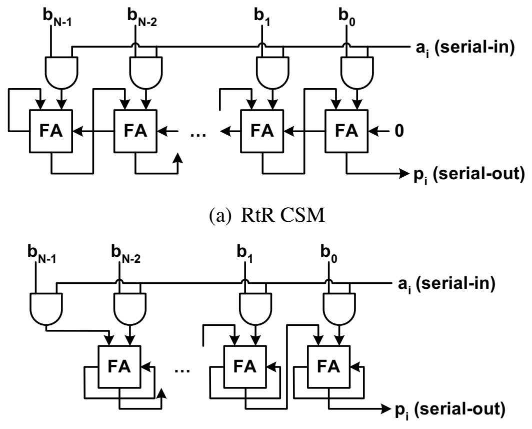


Figure 20. RtR CDM network for QCA

4.2. Multiplier design

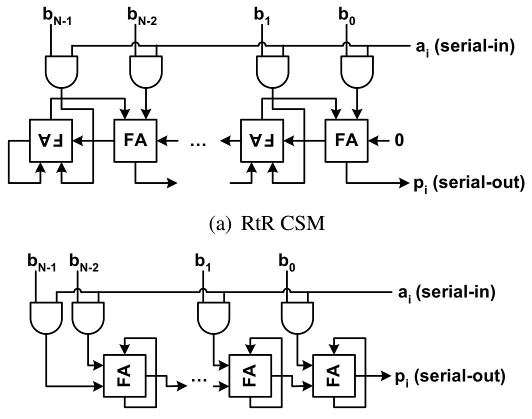
Previous literature [14, 15] presented a design for a multiplier which uses a structure like Fig. 10 except multiplying $Z^{-\frac{1}{2}}$ to all the horizontal flow delay units. That design inherently undergoes a long delay and has an unnecessary right-most full adder. In this paper, a RtR structure is chosen for the QCA circuit implementation because it reduces the first input to first output latency. The final design will have two options as in Figs. 19 and 20. Fig. 21 represents the general block diagrams of QCA multipliers and Fig. 22 shows the block diagrams of the optimized designs for QCA layouts.



(a) RtR CSM

(b) RtR CDM

Figure 21. Multiplier block diagrams



(a) RtR CSM

(b) RtR CDM

Figure 22. Modified multiplier block diagrams

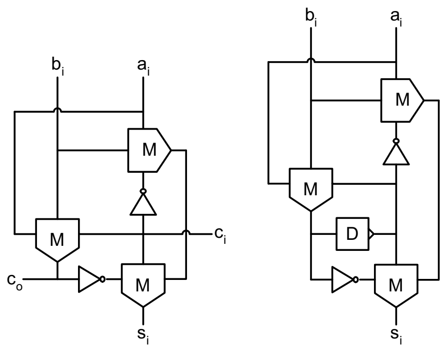
4.3. QCA Implementation

For easy comparisons, 4 bit multiplier examples are shown. Fig. 23 shows the bit product matrix for 4 bit multiplication.

$$
\begin{array}{cccccccc}
 & & & & a_3 & a_2 & a_1 & a_0 \\
\mathrm{x} & & & & b_3 & b_2 & b_1 & b_0 \\
\hline
 & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & \\
 & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & \\
\hline
p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

Figure 23. Bit product matrix for 4 bit multiplication

Full adders are used for the carry shift multiplier and bit-serial adders are used for the carry delay multiplier. Those adders are made using the "carry flow adder (CFA)" which is a layout optimized ripple carry adder. The bit-serial adder is similar to the full adder except that the carry-in and carry-out are connected internally with a one clock delay. Figs. 24 and 25 represent the schematics and layouts of both adders.



(a) Full adder (b) Bit-serial adder

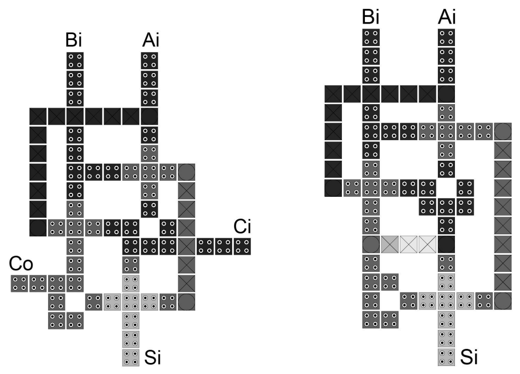
Figure 24. 1 bit adder schematics

Using these adders, 4 bit RtR CSM and CDM designs according to Fig. 22 are implemented as shown in Figs. 26 and 27. Multipliers for larger word sizes can be implemented easily by adding additional bit slices.

5. Design analyses

5.1. Simulation results

With QCADesigner ver. 2.0.3, the circuit functionality is verified. The following parameters are used for a bistable approximation: cell size = $20nm$, number of samples = 102400, convergence tolerance = 0.00001, radius of effect



(a) Full adder (b) Bit-serial adder

Figure 25. 1 bit adder layouts

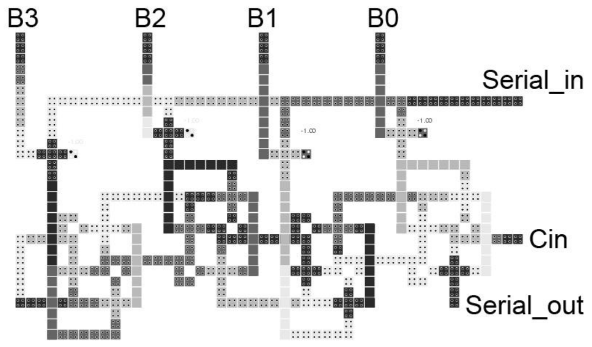


Figure 26. Layout of 4 bit carry shift multiplier

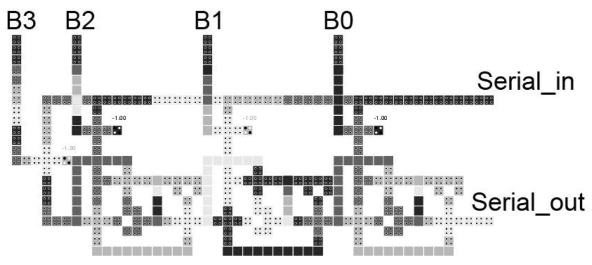


Figure 27. Layout of 4 bit carry delay multiplier

= $41nm$, relative permittivity = 12.9, clock high = 9.8e-22 J, clock low = 3.8e-23 J, clock amplitude factor = 2, layer separation = $11.5nm$, maximum iterations per sample = 10000 [18].

4 bit multiplier simulation results are provided with the input and output waveforms as shown in Fig. 28 for CSM and Fig. 29 for CDM. First and last input/output pairs are highlighted. The highlighted protrusion of "0" is only shown for understandability.

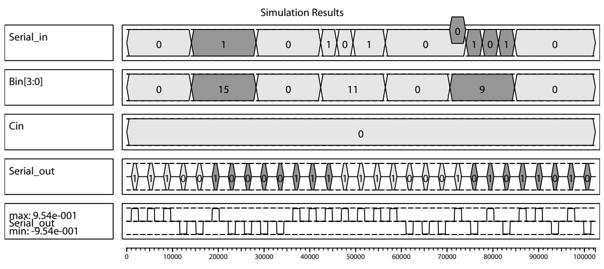


Figure 28. Simulation result of 4 bit carry shift multiplier

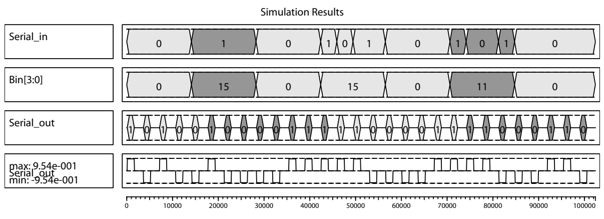


Figure 29. Simulation result of 4 bit carry delay multiplier

For an N bit case, multiplier inputs are an $N + 1$ bit number (1 bit serial input and N bit parallel inputs) and output is a 1 bit number (serial output) ignoring a constant carry-in. The serial input and output use the order from LSB to MSB and parallel inputs are repeated whenever a new serial input is available (N cycles). For the initialization of the multiplier, $N+1$ zero bits at each cycle are provided for the period of N clock cycles and padding $N+1$ zero bits at each cycle are also provided between the input sets for the period of N clock cycles. Completion time for one N bit multiplication takes $2N$ cycles.

5.2. Comparisons

Various word size carry shift multipliers and carry delay multipliers are implemented and compared in Table 1. Smaller word size multiplier trends don't match with the larger word size multiplier trends. The MSB of input B needs to be delayed more than the LSB due to the input synchronization. Because the CSM has one clock zone difference between the logical AND gate instead of two clock zones in CDM, the CSM only needs half as many clock zones for input synchronization. Synchronizing the inputs in large word size CDMs requires large areas and more cells even though the base design of CDM is simpler. Figs. 30 and 31 reflect those relations. As a design choice, the CDMs can be modified to have a smaller synchronizing section by increasing the latency. Those designs will have the same latency as the CSMs.

Table 1. Multiplier comparisons

	Complexity	Area	Latency
CSM4	507 cells	$1.04 \times 0.61\mu m^2$	$1\frac{1}{4}$ clocks
CSM8	1011 cells	$1.93 \times 0.61\mu m^2$	$1\frac{1}{4}$ clocks
CSM16	2043 cells	$3.67 \times 0.61\mu m^2$	$1\frac{1}{4}$ clocks
CSM32	4299 cells	$7.24 \times 0.67\mu m^2$	$1\frac{1}{4}$ clocks
CSM64	9579 cells	$14.30 \times 0.85\mu m^2$	$1\frac{1}{4}$ clocks
CDM4	406 cells	$1.05 \times 0.47\mu m^2$	1 clock
CDM8	903 cells	$2.12 \times 0.47\mu m^2$	1 clock
CDM16	1999 cells	$4.19 \times 0.47\mu m^2$	1 clock
CDM32	4575 cells	$8.47 \times 0.65\mu m^2$	1 clock
CDM64	11264 cells	$16.84 \times 0.95\mu m^2$	1 clock

6. Conclusions

Based on the QCA characteristics, a simple structure using a pipeline design has an advantage of reduced wire delays. The serial parallel multiplier is a possible solution with this characteristic. This paper presents serial parallel multiplication networks based on filter networks. The networks are derived from multiplication equations and implemented by network graphs. The design uses systolic array structures to pump out an output on every clock cycle and it is optimized for low latency to the first output. It also has a regular design for easy word size extension as well as small areas and a small number of cells used.

This research extends the QCA circuit designs to multiplication. For the fast multiplication of large word sizes, more complex multiplication algorithms should be investigated. Unlike the transistor circuits, added logic units do not guarantee fast operation. Future research should look for the optimal point along the delay and complexity tradeoff.

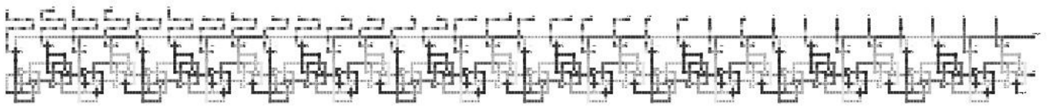


Figure 30. Layout of 32 bit carry shift multiplier

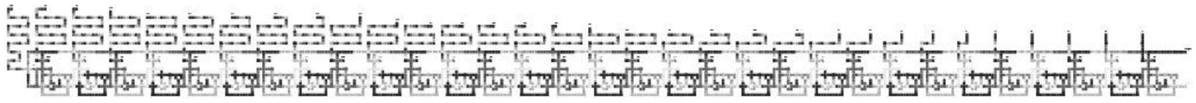


Figure 31. Layout of 32 bit carry delay multiplier

References

[1] International Technology Roadmap for Semiconductors. (ITRS) [Online]. Available: http://www.itrs.net, 2005.

[2] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, "Quantum cellular automata," *Nanotechnology*, pp. 49-57, January 1993.

[3] A. Orlov, *et al.*, "Experimental demonstration of a binary wire for quantum-dot cellular automata," *Appl. Phys. Lett.*, vol. 74, no. 19, pp. 2875-2877, 1999.

[4] I. Amlani, *et al.*, "Experimental demonstration of a leadless quantum-dot cellular automata cell," *Appl. Phys. Lett.*, vol. 77, no. 5, pp. 738-740, 2000.

[5] R. P. Cowburn, and M. E. Welland, "Room temperature magnetic quantum cellular automata," *Science*, vol. 287, no. 5457, pp. 1466-1468, 2000.

[6] H. Qi, *et al.*, "Molecular quantum cellular automata cells. Electric field driven switching of a silicon surface bound array of vertically oriented two-dot molecular quantum cellular automata," *Journal of the American Chemical Society*, vol. 125, pp. 15250-15259, 2003.

[7] R. K. Kummamuru, *et al.*, "Operation of a quantum-dot cellular automata (QCA) shift register and analysis of errors," *IEEE Trans. Electron Devices*, vol. 50, pp. 1906-1913, 2003.

[8] A. DeHon, and M. J. Wilson, "Nanowire-based sublithographic programmable logic arrays," *Proc. Intern. Sym. on Field-Program. Gate Arrays*, pp. 123-132, 2004.

[9] J. M. Seminario, *et al.*, "A molecular device operating at terahertz frequencies: Theoretical simulations," *IEEE Trans. Nano.*, vol. 3, no. 1, pp. 215-218, 2004.

[10] Y. Wang and M. Lieberman, "Thermodynamic behavior of molecular-scale quantum-dot cellular automata (QCA) wires and logic devices," *IEEE Trans. on Nano.*, vol. 3, no. 3, pp. 368-376, 2004.

[11] W. Wang, K. Walus, and G. A. Jullien, "Quantum-dot cellular automata adders," *Third IEEE Conference on Nanotechnology*, pp. 461-464, 2003.

[12] H. Cho and E. E. Swartzlander, Jr., "Pipelined carry lookahead adder design in quantum-dot cellular automata," *Conference Record of the Thirty Ninth Asilomar Conference on Signals, Systems and Computers*, pp. 1191-1195, 2005.

[13] H. Cho and E. E. Swartzlander, Jr., "Modular design of conditional sum adders using quantum-dot cellular automata," *Sixth IEEE Conference on Nanotechnology*, July 2006.

[14] K. Walus, G. A. Jullien, V. S. Dimitrov, "Computer arithmetic structures for quantum cellular automata," *Conference Record of the Thirty Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1435-1439, 2003.

[15] K. Walus and G. A. Jullien, "Design tools for an emerging SoC technology: quantum-dot cellular automata," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1225-1244, 2006.

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

[17] Danny Cohen, "A Mathematical Approach to Computational Network Design," in Earl E. Swartzlander, Jr., Ed., *Systolic Signal Processing Systems*, New York: Marcel Dekker, Inc., 1987, pp. 1-29.

[18] QCADesigner documentation. [Online] http://www.qcadesigner.ca