Design of non-restoring binary array divider in quantum-dot cellular automata

Huanqing Cui, Li Cai, Xiaokuo Yang, Chaowen Feng, Tao Qin

Science College, Air Force Engineering University, Xi'an 710051, People's Republic of China
E-mail: 13678146019@163.com

Published in Micro & Nano Letters; Received on 14th March 2014; Revised on 18th May 2014; Accepted on 10th June 2014

Quantum-dot cellular automata (QCA), a promising candidate nanotechnology for the next generation computers, has attracted the interest of researchers all over the world since 1993. The divider is a major component of arithmetic logic unit, which has a remarkable impact on the performance of the central processing unit. The widely used algorithm in the divider is the non-restoring division, but there is no work which has reported the implementation of non-restoring dividers based on QCA. Presented is the design of a non-restoring binary array divider in QCA and its validity is verified using QCADesigner software. The proposed non-restoring divider has the advantage of time-saving and is easy to control when compared with the existing restoring dividers.

1. Introduction: Quantum-dot cellular automata (QCA) [1, 2] is an alternative nanotechnology that may solve the problems encountered in complementary metal-oxide semiconductor (CMOS) technology because of the smaller and smaller feature size of the transistor. QCA relies on electron confinement in quantum dots. The fundamental element in QCA, namely the QCA cell, represents a bit by the configuration of electrons in the cell [3]. The most frequently used components in QCA [4] such as the wire and logic gates appear in Fig. 1. Actually, a QCA circuit is partitioned into four sequential clock zones 0, 1, 2 and 3 [5], and the computations are done under the control of the clock signals. As it is difficult to design sequential circuits in QCA [6], little research has concerned realising QCA iterative computational units, such as dividers.

Among all the arithmetic units, the divider is the most complicated and time-consuming one. Therefore engineers and algorithm designers always try their best to circumvent using the division operation. However, with the development of precise instrument, spaceflight and radar technology, the application of the divider is unavoidable. Till now, many algorithms can be used for the implementation of the divider, such as the restoring, non-restoring, Sweeney, Robertson and Tocher methods and the Newton iterative algorithm [7]. Restoring and non-restoring algorithms are based on addition and subtraction, which are suitable for integrated circuit implementation. Work by Kim and Swartzlander [8] has led to a restoring divider (RD) design for QCA. To the best of our knowledge, there has been no prior work on the implementation of a QCA non-restoring divider (NRD). In this Letter, we report for the first time a non-restoring binary array divider using the emerging QCA nanotechnology.

The reminder of this Letter is organised as follows. In Section 2, the principles of the non-restoring binary division algorithm are explained. In Section 3, a QCA implementation of the non-restoring array divider is presented. In Section 4, both the simulation results and analysis of the design are presented. Finally, we draw conclusions in Section 5.

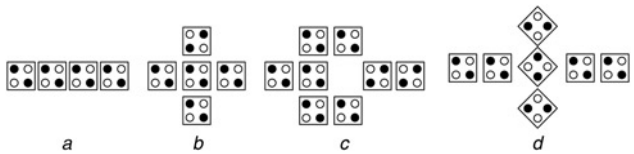

Figure 1 *Basic QCA components*
a QCA wire
b Majority gate
c Inverter
d Coplanar wire crossing

2. Non-restoring binary divider: First, we explain what is the restoring division algorithm [8], since it will be helpful to understanding the non-restoring division later. Throughout this Letter, the following notations are used for easy understanding:

N numerator or dividend
Y denominator or divisor
R_i partial remainder after ith iterations
i number of iterations
n number of bits
q quotient set for the algorithm
Q quotient for the division

The restoring algorithm is the most basic method of division, and the details of it can be presented by the following set of equations:

$$q_{i+1} = \begin{cases} 1, & \text{if } 2R_i > Y \\ 0, & \text{if } 2R_i < Y \end{cases} \tag{1}$$

$$R_{i+1} = 2R_i - q_{i+1} \cdot Y \tag{2}$$

We can get the partial remainder through a left shift of R_i and a subtraction: $R_{i+1} = 2R_i - Y$. If R_{i+1} is positive, then $q_{i+1} = 1$, else $q_{i+1} = 0$ and a restoring addition is needed. The addition is used to restore the proper partial remainder, $R_{i+1} = R_{i+1} + Y = 2R_i$.

Although the restoring division algorithm is really simple, it incurs excess delay introduced during the restoration process [7], which also leads to unnecessary power dissipation because the partial remainder remains the same at the end of the two cycles. Another problem is that we do not know how many times the restoring addition is needed and we also do not know when to do the restoring addition before a particular operation of the divider, which makes the control logic difficult to realise.

To overcome the problems presented by the restoring division, we can undertake an alternative method of binary division, the non-restoring division [9], also known as add–subtraction alternate algorithm, and the relevant divider is called the NRD. In this algorithm, if a negative partial remainder is produced at the end of a certain cycle, no restoration is executed.

doi: 10.1049/mnl.2014.0148

Non-restoring division is performed in a procedure by the following formulas:

$$q_{i+1} = \begin{cases} 1, & \text{if } R_i > 0 \\ 0, & \text{if } R_i < 0 \end{cases} \tag{3}$$

$$R_{i+1} = \begin{cases} 2R_i - Y, & \text{if } R_i > 0 \\ 2R_i + Y, & \text{if } R_i < 0 \end{cases} \tag{4}$$

$$r = \begin{cases} 2^{-n} \cdot R_n, & \text{if } R_n > 0 \\ 2^{-n} \cdot (R_n + Y), & \text{if } R_n < 0 \end{cases} \tag{5}$$

where $i = 0, 1, \ldots, n-1$ is the recursion index. The initial partial remainder R_0 equals the dividend, and r is the final remainder.

Thus, from (1) to (5) above, it is easy to see that every time the negative partial remainder appears, the non-restoring algorithm takes one addition step less than the restoring method. Hence, there will be a quite significant saving in the number of clock cycles over the entire division process.

In binary non-restoring division, partial remainders are obtained by a subtraction or an addition between the dividend and the successively right-shifted versions of the divisor. The quotient bit is determined by the sign of the partial remainder which also decides whether to add or subtract the shifted divisor in the next cycle.

A two-dimensional array of pipelined 2′s complement adder/subtractor cells (CAS cells) can be used to implement the non-restoring algorithm [10]. Fundamentally, the array consists of columns of carry propagate adders with one controlled input bit P. Figs. 2*a* and *b* illustrate a array divider with a 2-bit divisor ($0.y_1\ y_2$) and an 4-bit dividend ($0.x_1\ x_2\ x_3\ x_4$). The first bit '0' of both the divisor and dividend are their signs, it is used to represent a positive number. Each logic cell is made up of a full adder and an EXCLUSIVE-OR gate. The EXCLUSIVE-OR gate provides the divisor input to the full adder. The control signal P determines the function of the CAS cell, namely whether an addition or a subtraction is to be executed. The 2-bit divisor and the double-length

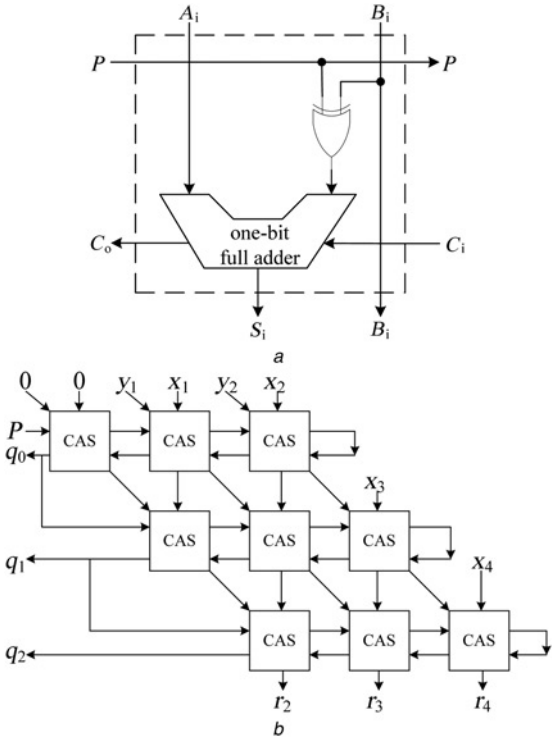

Figure 2 *Binary array divider*
a CAS cell
b 4-bit dividend, 2-bit divisor

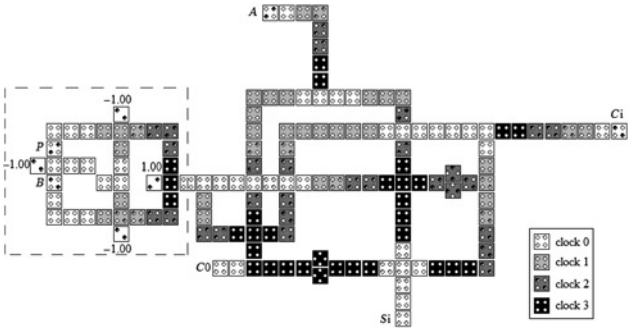

Figure 3 *QCA layout of the CAS cell*

4-bit dividend are imported at the top and right edges of the array, respectively. A 3-bit quotient is produced at the left side of the array, then every quotient bit is transported to the next row as the control signal P and the low carry of the rightmost cell. The 3-bit final reminder appears at the bottom of the array.

3. Divider implementation: The non-restoring array divider is composed of CAS cells that have one full adder and one EXCLUSIVE-OR gate. Fig. 3 shows the CAS cell layout. The EXCLUSIVE-OR gate [11] shown in the dashed box can be designed using only four majority gates and one inverter, its delay is a full clock cycle. The rest of the cell is a 1-bit full adder; it can be constructed using only three majority gates and two inverters [12]. The full adder takes three inputs, (XOR_OUT, A and C_i) and gives two outputs [$S_i = A \oplus B \oplus C_i$ and $C_o = \text{MAJ}(A, B, C_i)$]. Both the outputs have a latency of $1\frac{1}{4}$ clock cycles. The different colours in the layout are used to distinguish different clock zones.

An n-bit divider is formed by combining n^2 CAS cells to the regular array shown in Fig. 4, only $n = 3$ is presented here for simplicity. All the wire crossings in this layout are processed by the

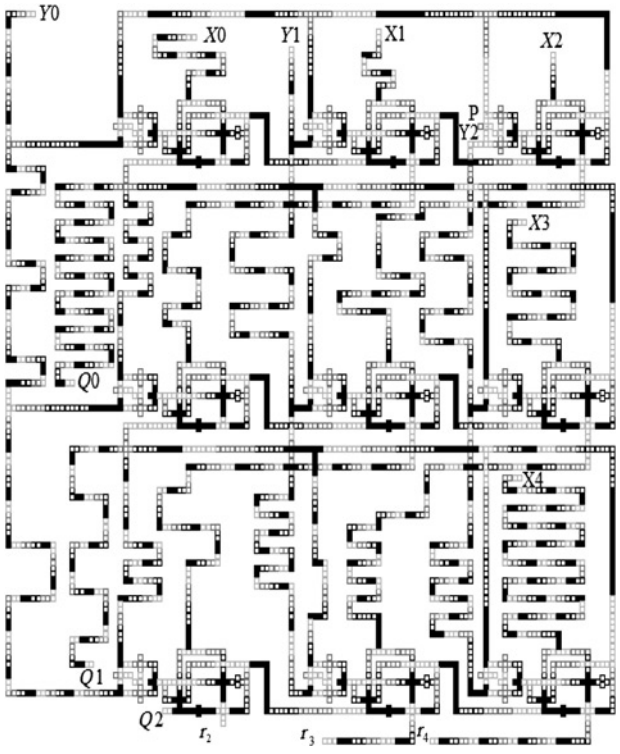

Figure 4 *QCA layout of the 3 × 3 non-restoring array divider*

technique proposed in [13], because there is no interference between the data transmitted in wires which are clocked in two non-adjacent clock regions. The total number of QCA cells is 3742 and the area of the divider layout is 6.22 μm^2.

Furthermore, we can find that the 8-bit NRD implemented by the 45 nm CMOS technology has an area of 2015.64 μm^2 [9]. It is very easy to see that the QCA NRD is much more area efficient than the normal divider because the area of the 8-bit QCA NRD is 56.98 μm^2, which also shows the advantage of the QCA device over traditional transistors.

4. Simulation and comparison: Simulation results are presented in this Section. The simulations were done using the bistable approximation engine of QCADesigner v2.0.3 [14] on a computer with a 2.8 GHz central processing unit and 1 GB RAM. The parameters of the simulation were as follows: cells in our design are set at 18 nm × 18 nm whereas the quantum dots in it have a diameter of 5 nm. The adjacent cells are placed with a centre-to-centre distance of 20 nm. In the bistable approximation options, the number of samples is 12 800, convergence tolerance is 0.001, radius effect is 65 nm, relative permittivity is 12.9, clock high is 9.8×10^{-22}, clock low is 3.8×10^{-23}, clock amplitude factor is 1, layer separation is 11.5 and maximum iterations per sample is 100. To verify the performance of the 3 × 3 NRD, we did the test using all possible input vectors and various sequences of the vectors. However, those vectors that can lead to overflow must be taken out. According to the non-restoring algorithm, if the first two bits of the dividend are larger than the divisor or equal to the divisor, an overflow will happen. Totally, 24 groups of input vectors have been tested. For a clear view, only half of the simulation results are provided in Fig. 5, each Figure contains four groups of input vectors.

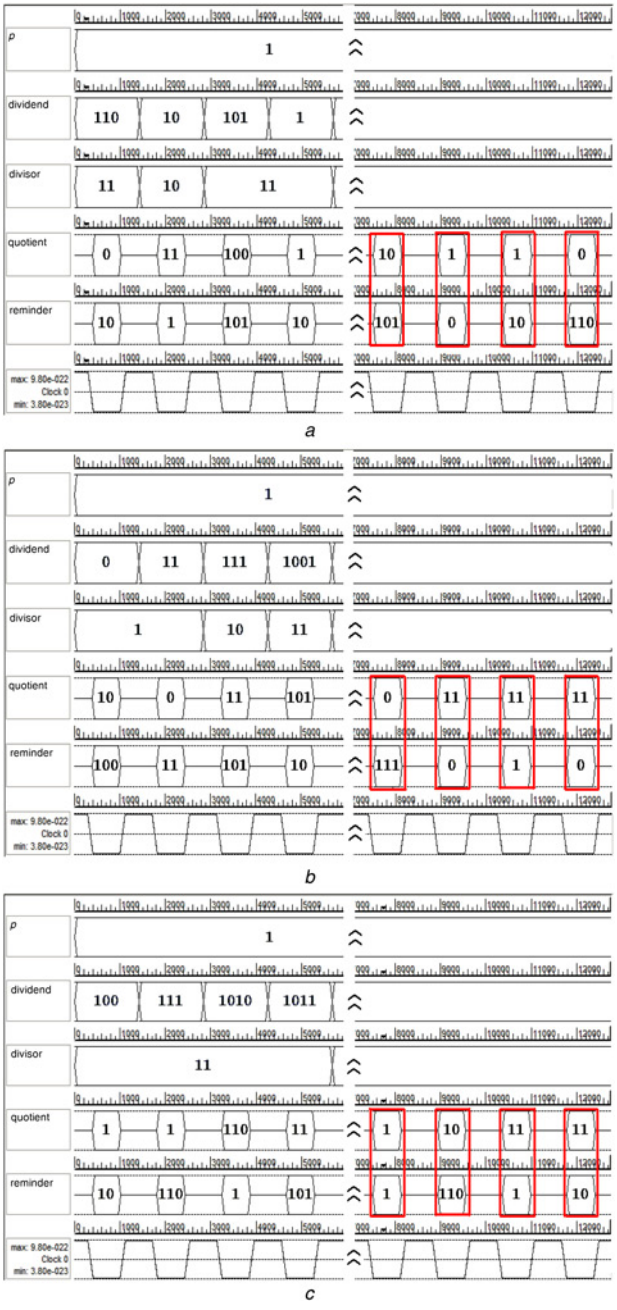

Figure 5 *Simulation results of the QCA 3 × 3 NRD*
a Dividend (0.0110, 0.0010, 0.0101, 0.0001), divisor (0.11, 0.10, 0.11, 0.11)
b Dividend (0.0000, 0.0011, 0.0111, 0.1001), divisor (0.01, 0.01, 0.10, 0.11)
c Dividend (0.0100, 0.0111, 0.1010, 0.1011), divisor (0.11, 0.11, 0.11, 0.11)

Correct output data, namely quotient vector and reminder vector, come out after $26\frac{1}{4}$ clock cycles as shown in the red boxes of Fig. 5. In Fig. 5*a*, the quotient vector is (0.10, 0.01, 0.01, 0.00) and the reminder vector is (0.0101, 0.0000, 0.0010, 0.0110). In Fig. 5*b*, the quotient vector is (0.00, 0.11, 0.11, 0.11) and the reminder vector is (0.0111, 0.0000, 0.0001, 0.0000). In Fig. 5*c*, the quotient vector is (0.01, 0.10, 0.11, 0.11) and the reminder vector is (0.0001, 0.0110, 0.0001, 0.0010). The first three rows of Fig. 5 are *P*, dividend vector and divisor vector, respectively. The last row is the clock signal of the output cells. It takes 406 s to run each simulation.

Table 1 shows the comparison of the proposed QCA NRD and the RD proposed in [8].

In general, the NRD has a latency of $3n^2 - 0.75$, where n is the operand size of the divider, and the counterpart of the RD in [8] is $4n^2 + 1$. Fig. 6 shows the latency comparison of them with the

Table 1 Comparison of dividers

Dividers	Latency	Cell amount	Area, μm^2
3 × 3 NRD	$26\frac{1}{4}$	3742	6.22
4 × 4 NRD	$47\frac{1}{4}$	6865	10.95
3 × 3 RD in [8]	37	6451	15.05
6 × 6 RD in [8]	145	42 236	86.22

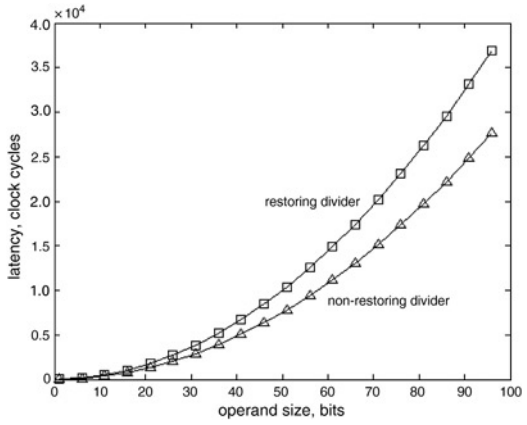

Figure 6 *Latency comparison of restoring and non-restoring dividers for various operand sizes*

doi: 10.1049/mnl.2014.0148

increasing operand size. As operand size becomes larger, the superiority of the NRD over RD is more obvious.

5. Conclusion: In this Letter, a non-restoring binary array divider has been implemented in QCA technology for the first time. It is constructed by CAS cell blocks. In the parallel array, data are processed in a pipelined style. When the operand size is fixed, the number of computing steps of the NRD is a constant, no matter what the dividend and divisor are. However, the RD does not have this characteristic. Hence, the control logic is easier to realise for the NRD. Furthermore, compared with the RD, the NRD demonstrates more advantages, such as fewer cell amounts, smaller layout area and lower latency. Thus, the NRD is more suitable for large operand size computation.

6. Acknowledgments: This work was supported by the National Natural Science Foundation of China (Grant No. 61172043), the Key Program of Shaanxi Provincial Natural Science for Basic Research (Grant No. 2011JZ015) and the Graduate Innovation Fund of Science College (Grant No. 2014LXYCXJJ002).

7 References

[1] Lent C.S., Tougaw P.D., Porod W., Bernstein G.H.: 'Quantum cellular automata', *Nanotechnology*, 1993, **4**, pp. 49–57

[2] Lent C.S., Tougaw P.D.: 'A device architecture for computing with quantum dots', *Proc. IEEE*, 1997, **85**, (4), pp. 541–557

[3] Kunal D., Debashis D., Mallika D.: 'Realisation of semiconductor ternary quantum dot cellular automata', *Micro Nano Lett.*, 2013, **8**, (5), pp. 258–263

[4] Tougaw P.D., Lent C.S.: 'Logical devices implemented using quantum cellular automata', *J. Appl. Phys.*, 1994, **75**, (3), pp. 1818–1825

[5] Yang X., Cai L., Zhao X.: 'Low power dual-edge triggered flip-flop structure in quantum dot cellular automata', *Electron. Lett.*, 2010, **46**, (12), pp. 825–826

[6] Huang J., Momenzadeh M., Lombardi F.: 'Design of sequential circuits by quantum-dot cellular automata', *Microelectron. J.*, 2007, **38**, pp. 525–537

[7] Stuart F.O., Michael J.F.: 'Division algorithms and implementations', *IEEE Trans. Comput.*, 1997, **46**, (8), pp. 833–854

[8] Kim S.-W., Swartzlander E.E.: 'Restoring divider design for quantum-dot cellular automata'. 11th IEEE Int. Conf. Nanotechnology, Portland Marriott, USA, 2011, pp. 1295–1300

[9] Kihwan J.: 'Modified non-restoring division algorithm with improved delay profile'. Thesis for the degree of Master, University of Texas at Austin, 2011

[10] Maurus C., Hamacher V.C.: 'An augmented iterative array for high-speed binary division', *IEEE Trans. Comput.*, 1973, **c-22**, (2), pp. 172–175

[11] Mohammad R.B., Mohammad M., Firdous A.: 'Performance evaluation of efficient XOR structure in quantum-dot cellular automata (QCA)', *Circuits Syst.*, 2013, **4**, pp. 147–156

[12] Heumpil C., Swartzlander E.E.: 'Adder and multiplier design in quantum-dot cellular automata', *IEEE Trans. Comput.*, 2009, **58**, (6), pp. 721–727

[13] Shin S.-H., Jeon J.-C., Yoo K.-Y.: 'Wire-crossing technique on quantum-dot cellular automata'. Proc. 2nd Int. Conf. Next Generation Computer and Information Technology, 2013, **27**, pp. 52–57

[14] Walus K., Dysart T.J., Jullien G.A., Budiman R.A.: 'QCADesigner: a rapid design and simulation tool for quantum dot cellular automata', *IEEE Trans. Nanotechnol.*, 2004, **3**, (1), pp. 26–31