A High-Performance Design of Generalized Pipeline Cellular Array

Journal:	*IEEE Computer Architecture Letters*
Manuscript ID	CAL-2019-10-0108
Manuscript Type:	Letters
Keyword:	B.2 Arithmetic and Logic Structures < B Hardware, B.2.4 High-Speed Arithmetic < B.2 Arithmetic and Logic Structures < B Hardware, B.6.1.a Cellular arrays and automata < B.6.1 Design Styles < B.6 Logic Design < B Hardware

A High-Performance Design of Generalized Pipeline Cellular Array

Zhufei Chu, *Member, IEEE,* Huiming Tian, Zeqiang Li, Yinshui Xia, and Lunyao Wang

Abstract—In this letter, we proposed a high-performance quantum-dot cellular automata (QCA) design of generalized pipeline cellular array (GPCA). The GPCA can perform all the basic arithmetic operations using only one arithmetic cell. Due to its flexibility, the high-performance GPCA design is of high interest for large-scale QCA designs. We proposed both the arithmetic unit and control unit designs of GPCA using a hybrid logic network that represented by majority-of-three (MAJ), inverter (INV), and three-input exclusive-OR (XOR_3). The introduction of XOR_3 operation can significantly reduce the number of gates, levels, and inverters, which in turn results in less area and fewer clock cycles of QCA layout. On average, the proposed design can reduce the area and latency by 39.27% and 35.89% compared with the-state-of-art, respectively.

Index Terms—majority, three-input XOR, pipeline cellular array, quantum-dot cellular automata (QCA)

1 Introduction

Complementary metal-oxide-semiconductor (CMOS) technology is gradually reaching its physical limitation in recent years. Quantum-Dot Cellular Automata (QCA) is considered as one of the promising emerging nanotechnologies for its high speed, ultra-low power consumption and high density [1]. The principle for binary computing of QCA is based on Coulomb interactions. The polarization of electrons in a QCA cell can represent logic '0' and '1'.

The primitives used in QCA are inverter (INV) and majority-of three (MAJ) gate which can be set to perform like AND/OR gates. The MAJ gate over three Boolean variables a, b, and c can be represented as $\mathbb{M}(a,b,c) = ab + ac + bc$. For the three-input XOR gate (XOR_3), it can be also represented as the MAJ forms, i.e., $\mathbb{X}(a,b,c) = a \oplus b \oplus c = \mathbb{M}(\bar{a}, \mathbb{M}(a,b,\bar{c}), \mathbb{M}(a,\bar{b},c))$. By setting any one of the three inputs to constant 0/1, we can obtain AND/OR/XOR/XNOR operations over the remaining non-constant inputs. As an example, $ab = \mathbb{M}(a,b,0)$ and $a \oplus b = \mathbb{X}(a,b,0)$. Compared with the direct realization by the MAJ forms, an independent XOR_3 QCA gate with less area is proposed in [2] recently, which provides a fundamental block for MAJ/INV/XOR_3 based QCA designs.

Computer arithmetic plays a key role in modern computing. The implementations of various arithmetic circuits in QCA have been investigated in the literature, such as divider [3], multiplier [4], [5], squarer [6], and square rooting circuit [7], etc. Instead of specific arithmetic circuit design, a universal arithmetic unit named generalized pipeline cellular array (**GPCA**) which can perform all the basic arithmetic operations in pipeline manner was proposed in [8] and implemented in QCA by [9]. The GPCA consists of an adder-subtractor arithmetic unit (**AU**) and a control logic unit (**CU**). Unlike the work in [9] which focuses on realizing the GPCA only by MAJ and INV primitives, in this letter, we redesigned the AU and CU in GPCA using the combinations of MAJ, INV, and XOR_3 operations for high-performance design. The experimental results show that our design can averagely reduce the area and latency by 39.27% and 35.89%, respectively.

(Corresponding author: Zhufei Chu). *Z. Chu, H. Tian, Z. Li, Y. Xia, and L. Wang are with the Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, China. (email:chuzhufei@nbu.edu.cn)*

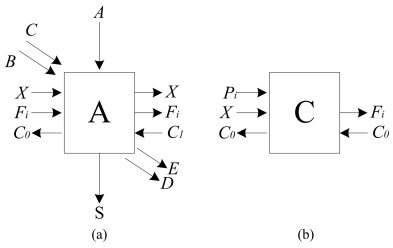

Fig. 1. Structure of the GPCA, (a) AU, (b) CU.

2 Proposed Designs

In this section, we first review the Boolean logic networks by related work, then the proposed designs of the GPCA by using MAJ, INV, and XOR_3 primitives are presented.

2.1 Boolean Expressions of the GPCA

The GPCA contains AU and CU, the structures of which are shown in Fig. 1 (a) and (b), respectively. The AU has six inputs, i.e., A, B, C, F_i, X, and C_1, and four outputs, i.e., S, D, E and C_0. The CU has three inputs, i.e., C_0, P_i, and X, and one output F_i. The Boolean expressions of AU used in the work [8] are given as in (1).

$$\left.\begin{aligned} S &= (A \oplus (B \oplus X) \oplus C_1)F_i + A\overline{F_i} \\ C_0 &= (B \oplus X)(A + C_1) + AC_1 \\ D &= BC + CF_i = C(B + F_i) \\ E &= B + CF_i = (B + C)(B + F_i) \end{aligned}\right\} \quad (1)$$

For the CU part, the Boolean expression is defined by

$$F_i = C_0 X + P_i \bar{X}. \quad (2)$$

Since QCA can natively implement MAJ and INV operations as the fundamental logic devices, the authors in [9] proposed a functional equivalent MAJ/INV based logic network.

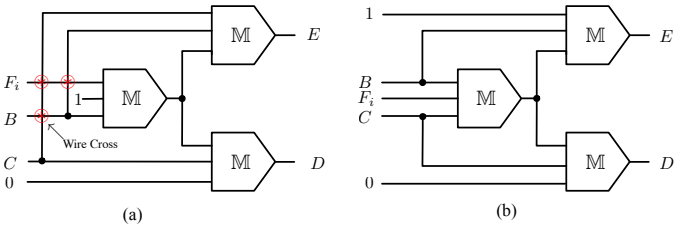

Fig. 2. Two logic networks for the outputs D and E, (a) The expressions proposed by [9] may face wire-crossing problem, (b) Our design.

The MAJ/INV based expressions of AU are shown in (3), where n_0, n_1, n_2, and n_3 are internal nodes.

$$\left.\begin{aligned} S &= \mathbb{M}(\mathbb{M}(n_0, F_i, 0), \mathbb{M}(A, \bar{F}_i, 0), 1) \\ C_0 &= n_1 \\ D &= \mathbb{M}(C, n_3, 0) \\ E &= \mathbb{M}(B, C, n_3) \\ n_0 &= \mathbb{M}(\bar{n}_1, \mathbb{M}(n_2, \bar{A}, C_1), A) \\ n_1 &= \mathbb{M}(n_2, A, C_1) \\ n_2 &= \mathbb{M}(\mathbb{M}(\bar{X}, B, 0), \mathbb{M}(X, \bar{B}, 0), 1) \\ n_3 &= \mathbb{M}(B, F_i, 1) \end{aligned}\right\} \tag{3}$$

The MAJ/INV based expression for the CU is defined by

$$F_i = \mathbb{M}(\mathbb{M}(X, C_0, 0), \mathbb{M}(\bar{X}, P_i, 0), 1). \tag{4}$$

In total, the logic network of AU proposed by [9] requires tweleve MAJ gates, six levels, and five INVs, while the CU needs three MAJ gates, two levels, and one inverter.

2.2 MAJ/INV/XOR$_3$ based Designs

We will explain the proposed Boolean expressions for all the output signals of AU first, then the design of CU will be presented.

In terms of the outputs S and C_0, from (1) we can see the subexpression $(B \oplus X)$ is appeared in both expressions. Moreover, the internal node n_2 in (3) actually preforms the same XOR operation using MAJ/INV primitives, i.e., $n_2 = B \oplus X$. The recent exact synthesis approach [10] indicates the two- or three-input XOR functions can be realized by at least three MAJ gates. Hence, the expression of n_2 in (3) is already optimal in size and depth. Since the XOR$_3$ gate was proposed in the literature, we make use of MAJ/INV/XOR$_3$ based expressions for the design. For the output C_0, we replace n_2 as $\mathbb{X}(B, X, 0)$, then

$$C_0 = \mathbb{M}(\mathbb{X}(B, X, 0), A, C_1) \tag{5}$$

For the output S, since $(a \oplus b)c + a\bar{c} = a \oplus bc$ holds, we can assign $a = A$, $b = (B \oplus X) \oplus C_1$, and $c = F_i$. Then the expression of S in (1) is rewriten as

$$\begin{aligned} S &= (((B \oplus X) \oplus C_1)F_i) \oplus A \\ &= \mathbb{X}(\mathbb{M}(\mathbb{X}(B, X, C_1), F_i, 0), A, 0) && \text{(6a)} \\ &= \mathbb{X}(\mathbb{M}(\mathbb{X}(\mathbb{X}(B, X, 0), C_1, 0), F_i, 0), A, 0). && \text{(6b)} \end{aligned}$$

Equation (6a) requires only three MAJ/XOR$_3$ operations, in contrast, equation (6b) needs one additional XOR$_3$ operation but it can share the node $\mathbb{X}(B, X, 0)$ appeared in (5). We reserve both the designs for further evaluations in QCA technology.

The outputs of D and E is relatively simple. The logic network proposed by [9] is shown in Fig. 2(a), in which signals may face the wire-crossing problem. Minimizing the number of wire-crossing in the logic network is of high interest for the QCA technology [11]. For the single-layer restriction, the wire-crossing cannot be implemented. Moreover, too many wire-crossings increase the design complexities of a QCA layout since more area and timing are required.

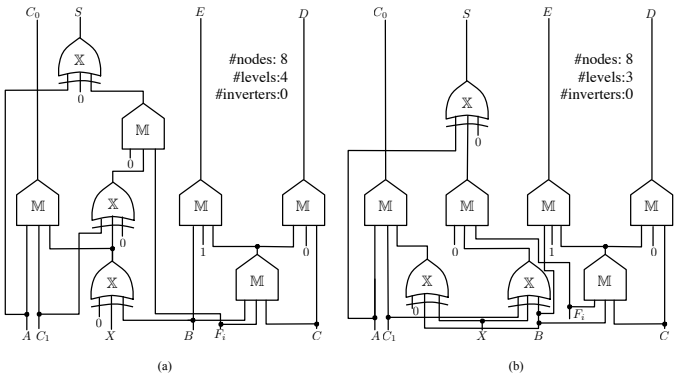

Fig. 3. The logic networks of the proposed design of AU, we make use of (a) (6b), (b) (6a) as the expression for the output S.

The proposed logic network for outputs D and E is shown in Fig. 2(b), in which the wire-crossing problem is solved. For the output D, the expression is

$$D = \mathbb{M}(\mathbb{M}(F_i, B, C), C, 0). \tag{7}$$

By expansion and simplification, one can verify the function equivalence of (7) and the D expression in (3).

$$\begin{aligned} D &= (F_iB + F_iC + BC)C = F_iBC + F_iC + BC \\ &= F_iC + BC = (F_i + B)C \\ &= \mathbb{M}(\mathbb{M}(F_i, B, 1), C, 0) \end{aligned} \tag{8}$$

Similarly, the output E is represented as

$$E = \mathbb{M}(\mathbb{M}(F_i, B, C), B, 1) \tag{9}$$

such that the subexpression $\mathbb{M}(F_i, B, C)$ is shared with (7).

The expressions of the AU are the collections of equations (5), (6), (7), and (9). Since (6) has two versions of expression, i.e., (6a) and (6b), we present the logic network of both of the two versions in Fig. 3. Both of the designs require eight gates, including five MAJ gates and three XOR$_3$ gates. The number of logic levels of Fig. 3(b) is three while the one in Fig. 3(a) is four. Besides, no inverters are required for both of the designs. The main difference is the usage of the XOR$_3$ operation. Fig. 3(b) contains one fully-utilized XOR$_3$ node by rewriting $(B \oplus X) \oplus C_1$ into $\mathbb{X}(B, X, C_1)$. Since the node $\mathbb{X}(B, X, 0)$ has two fanouts in Fig. 3(a), we still need this node in Fig. 3(b) for functional equivalence. In comparison, the design proposed in [9] requires twelve MAJ gates, six

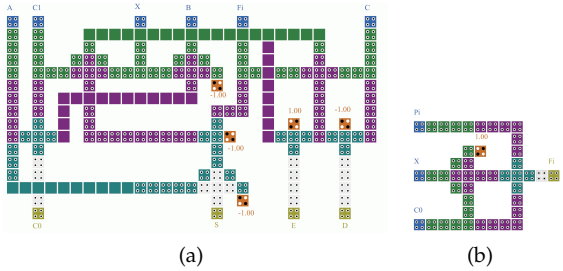

Fig. 4. The QCA layout of the proposed designs (a) AU (b) CU

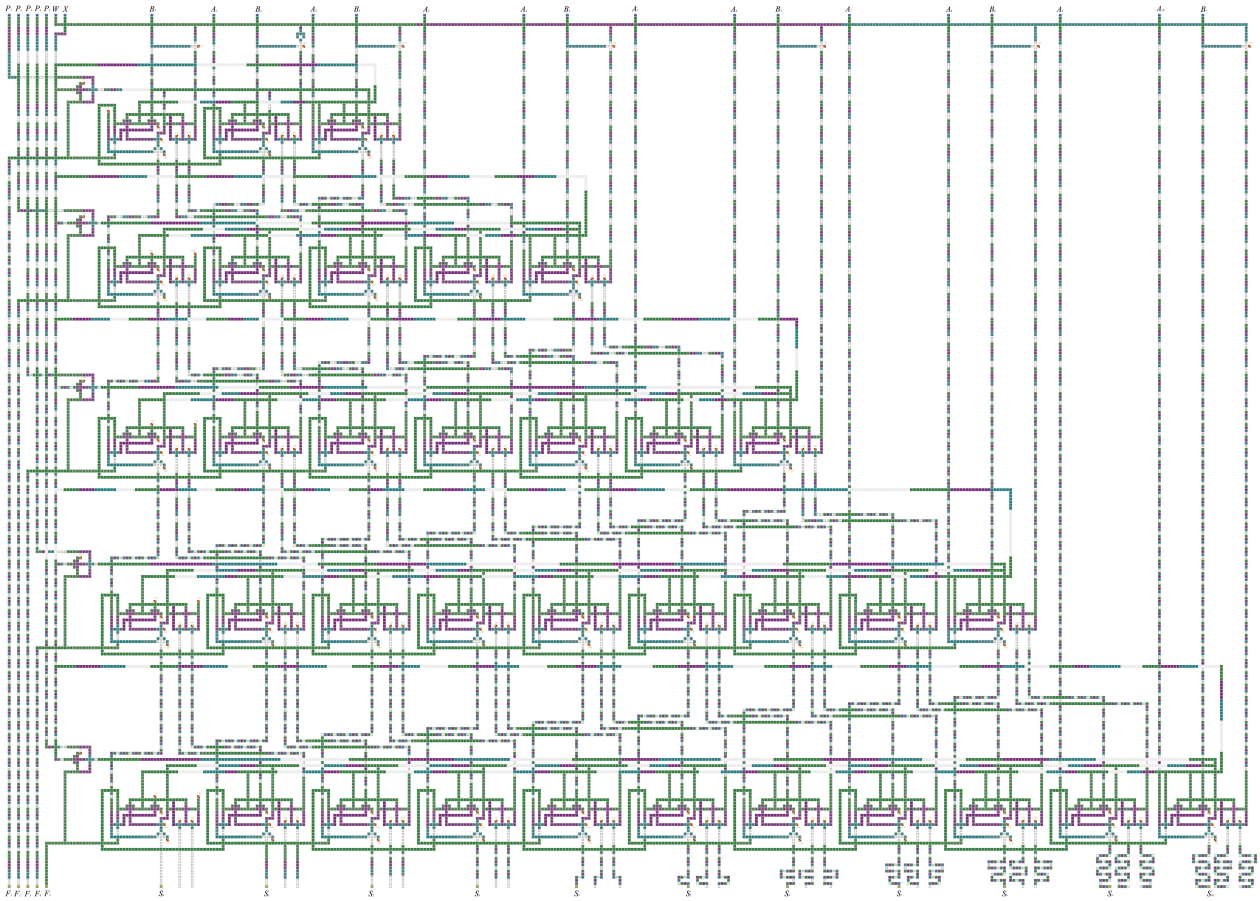

Fig. 5. The QCA layout of our proposed array

TABLE 1
Comparisons of the proposed GPCA and the designs in [9]

Designs	[9]			Proposed					
	Area (μm^2)	Latency	Cells count	Area (μm^2)	%	Latency	%	Cells count	%
Arithmetic cell	0.68	4	682	0.21	69.12	1	75	217	68.18
Control cell	0.11	1.5	77	0.05	54.55	1	33.33	44	42.86
Pipeline Array(n=3)	23.35	34	15098	17.06	26.94	25	26.47	10312	31.70
Pipeline Array(n=4)	39.44	51	26698	29.80	24.44	39	23.53	17258	35.36
Pipeline Array(n=5)	59.47	71	42174	46.80	21.30	56	21.13	26094	38.13
Average improvement					39.27%		35.89%		43.25%

levels, and five inverters. Thus, our design has more compact representations.

In terms of CU part, the design proposed in [9] needs three MAJ gates and one inverter. Due to $ab + \bar{b}c = \mathbb{M}(a, c, \overline{a \oplus b}) = \mathbb{M}(a, c, \mathbb{X}(a, b, 1))$, we can assign $a = C_0$, $b = X$, and $c = P_i$. Then the expression of F_i can be rewritten as

$$F_i = \mathbb{M}(\mathbb{X}(X, C_0, 1), P_i, C_0). \quad (10)$$

Note that a constant '1' input in XOR_3 gate leads to the inversion of the XOR operation over the other two inputs. Hence, the inverter is also eliminated thanks to the usage of XOR_3 operation.

3 Simulation Results and Comparison

In this section, the simulation results of our designs were presented. All the designs and simulations are done by QCADesigner 2.0.3 [12]. We have set the same parameters with the ones used in [9], i.e., the number of samples is 12800, convergence tolerance is 0.001, radius of 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 2, layer separation is 11.5, and maximum iterations per sample is 100.

TABLE 2
Comparisons of the proposed two AU designs

Designs	Area (μm^2)	Latency	Cells count	Logic levels
Fig. 3(a)	0.25	1.5	238	4
Fig. 3(b)	0.21	1	217	3
Improvement	16%	33.33%	8.82%	25%

For the proposed two designs of AU shown in Fig. 3, we realized the designs by the QCADesigner and the comparison

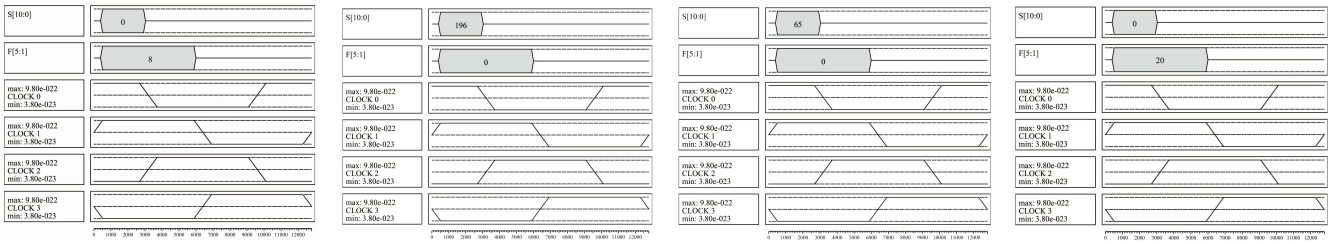

Fig. 6. Simulation results for (a) Division of 10000 by 10 (b)Squaring of 1110 (c) Multiplication of 1101 and 101 (d)Square rooting of 110010000

results are shown in Table 2, which indicate the logic network of Fig. 3(b) has a better performance in both the area and delay. Therefore, we adopt this AU design as the building block for the subsequent n-bit GPCA designs. Fig. 4 shows the QCA layout of our proposed AU and CU, from which we can see that both of the AU and CU have the same delay of 1 clock cycle.

The proposed AU and CU are the building blocks to construct the GPCA for performing arithmetic operations of any number of bits. Comparison results of the proposed designs with the state-of-the-art approach [9] is given in Table 1. It can be seen from the data in the table that both the AU and CU designs achieve better performance in the area, latency, and cells count. The proposed AU design has reduced the area by 69.12% from 0.68 μm^2 to 0.21 μm^2 and the latency is reduced by 75% from 4 clock cycles to 1 clock cycle. Moreover, the cells count is reduced from 682 to 217, which is 68.18% improvement. The better performance can be also seen for the CU design. This advantage also carries over into the n-bit GPCA designs. As an example, the proposed 5-bit pipeline array design has a total number of 26,094 QCA cells and an area of 46.80 μm^2 with a latency of 56 clock cycles, while the one in [9] has a total number of 42,174 QCA cells and an area of 59.47 μm^2 with a latency of 71 clock cycles. On average, the proposed design has reduced the area, latency, and cells count by 39.27%, 35.89%, and 43.25%, respectively.

The proposed AU and CU designs reduced the clock cycles by 3 and 0.5 compared with the ones in [9], respectively. Thus, theoretically speaking, the delay of each level in GPCA can be reduced by 3.5 clock cycles. However, in favor of the clock design of the QCA layout, we added a 0.5 clock cycle per level in our design. As a result, the delay of the first level has 6 clock cycles and the calculation formula of the delay is as given in (11). The reader can refer to [9] for details of delay calculation.

$$Delay = 1 + \sum_{i=1}^{n} (5 + 3(i-1)) \tag{11}$$

Generally, for the n-bit GPCA design, the proposed design can reduce $3n$ clock cycles compared with [9]. For example, the latency of the 5-bit GPCA is $Delay_{n=5} = 56$, in which $3n = 3 \times 5 = 15$ clock cycles are saved.

Fig. 6 shows the simulation of our pipeline array. For division, we set A's as 10000 and B's as 10 we got the result of 1000 (8 in decimal) in F's and the remainder of 0 in S's as show in Fig. 6(a). In Fig. 6(b), we set P's as 1110 and we got the result of 11000100 (196) in S's for squaring operation. For multiplication, we set A's 1101 and B's as 101 and we got the result of 1000001 (65) in S's. For square rooting, we put A's as 110010000 and we got the result of 10100 (20) in F's and the remainder of 0 in S's.

4 Conclusions

In this letter, we proposed a QCA design for generalized pipeline cellular array (GPCA). Unlike existing GPCA design using only the majority logic network, the proposed array is constructed by additionally introducing three-input XOR (XOR_3) gates. Thus, the GPCA is represented by a hybrid logic network by using majority, inverter, and XOR_3 primitives. The proposed arithmetic and control unit have a significant reduction of area, latency, and cells count. On average, we can achieve 39.27%, 35.89%, 43.25% reduction of area, delay and cells count compared with the state-of-the-art.

Acknowledgments

This work was supported in part by NSFC under Grant No 61871242 and in part by K.C.Wong Magna Fund in Ningbo University.

References

[1] Craig S Lent, P Douglas Tougaw, Wolfgang Porod, and Gary H Bernstein. Quantum cellular automata. *Nanotechnology*, 4(1):49, 1993.

[2] Ali Newaz Bahar, Sajjad Waheed, Nazir Hossain, and Md Asaduzzaman. A novel 3-input xor function implementation in quantum dot-cellular automata with energy dissipation analysis. *Alexandria Engineering Journal*, 57(2):729–738, 2018.

[3] TN Sasamal, AK Singh, and U Ghanekar. Design of non-restoring binary array divider in majority logic-based qca. *Electronics Letters*, 52(24):2001–2003, 2016.

[4] Liang Lu, Weiqiang Liu, Máire O'Neill, and Earl E Swartzlander. Qca systolic array design. *IEEE Transactions on computers*, 62(3):548–560, 2011.

[5] Weiqiang Liu, Earl E Swartzlander Jr, and Máire ONeill. *Design of semiconductor QCA systems*. Artech House, 2013.

[6] O Giannou, HT Vergos, and D Bakalis. Squarers in qca nanotechnology. In *2012 12th IEEE International Conference on Nanotechnology (IEEE-NANO)*, pages 1–6. IEEE, 2012.

[7] Mohammad Reza Jahangir, Shadi Sheikhfaal, Shaahin Angizi, Keivan Navi, and Firdous Ahmad. Designing nanoelectronic-compatible 8-bit square root circuit by quantum-dot cellular automata. In *2015 IEEE International Symposium on Nanoelectronic and Information Systems*, pages 23–28. IEEE, 2015.

[8] AK Kamal, Harpreet Singh, and DP Agrawal. A generalized pipeline array. *IEEE Transactions on Computers*, 100(5):533–536, 1974.

[9] Amjad F Almatrood and Harpreet Singh. Design of generalized pipeline cellular array in quantum-dot cellular automata. *IEEE Computer Architecture Letters*, 17(1):29–32, 2018.

[10] Mathias Soeken, Luca Gaetano Amarù, Pierre-Emmanuel Gaillardon, and Giovanni De Micheli. Exact synthesis of majority-inverter graphs and its applications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(11):1842–1855, 2017.

[11] Rajdeep Kumar Nath, Bibhash Sen, and Biplab K Sikdar. Optimal synthesis of qca logic circuit eliminating wire-crossings. *IET Circuits, Devices & Systems*, 11(3):201–208, 2017.

[12] K. Walus, T. J. Dysart, G. A. Jullien, and R. A. Budiman. Qcadesigner: a rapid design and simulation tool for quantum-dot cellular automata. *IEEE Transactions on Nanotechnology*, 3(1):26–31, 2004.