

# Serial Concatenated Convolutional Code Encoder in Quantum-dot Cellular Automata

Yongqiang Zhang<sup>a</sup>, Guangjun Xie<sup>a,\*</sup>, and Jie Han<sup>b</sup>

<sup>a</sup> School of Electronic Science and Applied Physics, Hefei University of Technology, Hefei, 230009, China

<sup>b</sup> Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, T6G 1H9, Canada

## ABSTRACT

Quantum-dot cellular automata (QCA) are a prospective nanotechnology with striking performance for complementing complementary metal oxide semiconductor (CMOS) based integrated circuit technique. For applications in communications, serial concatenated convolutional codes (SCCCs) have been investigated due to their negligible error floor for deep space communication. Using a combination of top-down and bottom-up approaches, we design and implement an SCCC encoder in QCA technology. In this work, a Bose-Chaudhuri-Hocquenghem (BCH) code encoder with the ability for correcting burst errors, a pseudo random interleaver for permuting signal bits, and a convolutional code encoder for correcting random errors are logically connected in series for composing the SCCC encoder. Specifically, inspired by the idea for a divider design, a layout for  $(n, k, t)$  BCH code encoders is proposed and implemented with QCA cells. For using the four-phase clock mechanism, we also devise the serial-to-parallel and parallel-to-serial converters for constructing the interleaver. In addition,  $(u, f, K)$  convolutional code encoders with various constraint lengths are designed as well. The functionalities and validity of the proposed circuit design are verified by using QCADesigner.

*Keywords:* BCH codes, Convolutional codes, Serial concatenated convolutional codes, Encoder, Interleaver, Modulo-2 adder, Quantum-dot cellular automata

## 1. Introduction

Quantum-dot cellular automata (QCA) offer a computing paradigm and information transmission pattern in the nanometer regime [1]. Binary information is represented by the positions of electrons in a cell that consists of four quantum dots and two free mobile electrons. The information is then transmitted by purely field-coupled interaction between cells [2]. By this means, there is no current flow in QCA circuits, so that it enables systems with extremely low energy dissipation [3]. Along with ultrahigh integration and processing speed, these performance merits make QCA a hopeful candidate for complementing complementary metal oxide semiconductor (CMOS) based integrated circuits [4-6]. Additionally, to correctly control signal flow in complex circuits, a quasi-adiabatic switching mechanism was introduced to QCA, resulting in four cyclic clock zones, each of which is composed of four phases (switch, hold, release and relax), respectively [7]. Therefore, a binary string will successively transmit through cells defined in these zones.

The design methodology for a QCA system was derived from the automated CMOS design process [8]. Majority-based logic synthesis techniques were put forward to reduce the number of operational components, leading to optimized and efficient systems [9, 10]. Meanwhile, basic arithmetic and memory circuits were realized as well [11-13]. However, the wiring in a system heavily relies on the adopted clock scheme. Two-dimensional and Universal-Scalable-Efficient (USE) clocks provide preferable solutions at present [14, 15]. One can use a basic clock for any circuit to ensure the synchronization of signals arriving at the same device [16]. Naturally, related evaluation criteria for variable circuits, such as the cost and power dissipation, were proposed

to select the best scheme [17, 18]. It is worth mentioning that nanomagnon logic and molecular QCA may be more promising in physical implementation because of their ambient working temperature [19, 20].

In order to achieve reliable data transmission and communication, coding circuits are crucial. Error correcting codes typically include block codes (BCs) and convolutional codes (CCs). Specifically, BCs (including Hamming codes, low density parity check (LDPC) codes, and Bose-Chaudhuri-Hocquenghem (BCH) codes) split a long message into independent fragments and separately encode them by adding specific parity bits into each fragment. Hamming code encoders and LDPC code decoders were already designed with QCA [21-23]. BCH codes are one of the most important cyclic codes for correcting multiple burst errors and widely used in fiber-optical communication and satellite communication systems. The codeword for BCH codes is usually generated by linear feedback shift registers (LFSRs) in conventional circuits. To the best of our knowledge, BCH codes have not been studied in QCA technology.

CCs differ from BCs in that the parity bits are not only related to the current bits but also involving former  $K+1$  fragments, where  $K+1$  indicates the constraint length of the operational CC encoder. Three schemes for rate-1/2 and constraint length-3 CC encoders were presented [24] to directly map conventional circuits to QCA with a high cost. In addition, Turbo codes or parallel concatenated convolutional codes (PCCCs) make it possible to approach the Shannon limit in terms of bit error rate (BER) [25]. Turbo code encoders using parallel concatenated recursive CC encoders and an interleaver were also investigated in QCA and simulated by hardware description language for QCA (HDLQ) [26]. A component in an interleaver is a serial-to-parallel converter [22, 26, 27]. Although these designs seem to be simple in QCA, an additional special clock was applied to keep some cells in the Relax

---

\* Corresponding author.  
E-mail addresses: ahzhangyq@mail.hfut.edu.cn (Y. Zhang),  
gjxie8005@hfut.edu.cn (G. Xie), jhan8@ualberta.ca (J. Han).

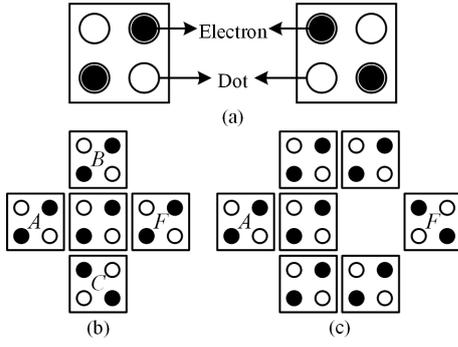


Fig. 1. QCA basics. (a) Cells. (b) Three-input majority voter. (c) Inverter.

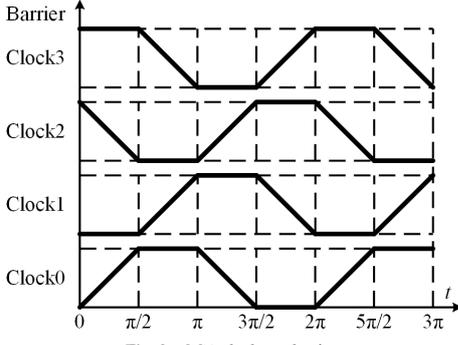


Fig. 2. QCA clock mechanism

phase to complete signal transmission, which violets the four-phase clock mechanism and results in complex clock wiring under the QCA substrate layer. Furthermore, with increasing signal-noise ratio (SNR), the BER of PCCCs may not continue to decrease, reaching the so-called error floor. Consequently, serial concatenated convolutional codes (SCCCs) were invented to provide superior error-free performance [28]. In some cases, SCCC can be considered as a credible or superior replacement for PCCCs. One can find further theoretical basics and comparisons in [28]. SCCC encoders were widely used in deep space communication due to their negligible error floor by employing serially concatenated BC and CC encoders.

In this paper, we propose designs of SCCC encoders in QCA and verify the designs by simulation. The main contributions of this work are as follows:

- A framework for designing SCCC encoders is developed by using QCA cells for the first time.
- Using the four-phase clock mechanism in QCA, we devise a new scalable serial-to-parallel converter and an expandable parallel-to-serial converter.
- A scalable interleaver is designed with the aforementioned two converters, which can also be used as a Last-In-First-Out (LIFO) converter.
- Abandoning the LFSR structure, we propose a layout for BCH code encoders by applying coplanar XOR gates, and then implement encoders with various error correcting capabilities in QCA.
- The CC encoders with various constraint lengths are designed with XOR gates.

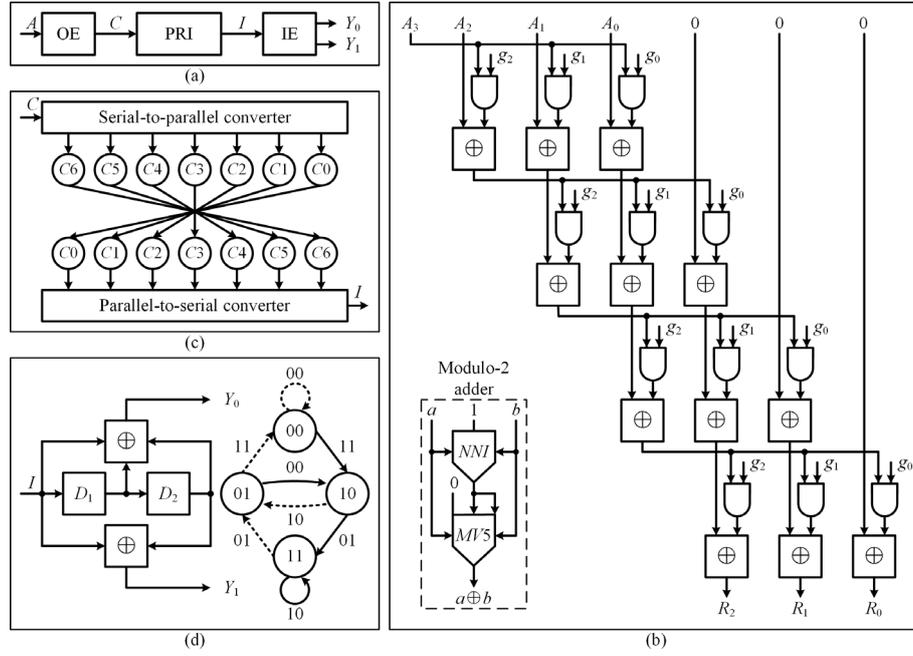
- The functionalities and validity of presented circuits are demonstrated with QCADesigner.
- The QCA costs for BCH code and CC encoders are calculated and summarized.

This paper proceeds as follows: the background for QCA, including basic devices and clock mechanism, is briefly reviewed in Section 2. Section 3 presents the rationale of SCCC encoders in detail. Each component and the full circuit of an SCCC encoder are then designed in QCA in Section 4. Simulation results are presented in Section 5. Finally, Section 6 concludes this paper.

## 2. QCA background

The primitive in QCA is a cell that can be viewed as a charge container with four quantum dots and two free mobile electrons [29]. These electrons can quantum mechanically tunnel between the dots inside one cell. The Coulomb interaction between two electrons tends to localize them in a diagonal pattern, leading to two configurations for a cell to encode binary information, either logic “1” or “0” as shown in Fig. 1(a), respectively. A typical semiconductor cell is defined with a width of 18nm in QCADesigner and employed as a prototype in circuit design and simulation [16]. As illustrated in Fig. 1(b), the three-input majority voter (*MV3*) is a distinctive logic element in QCA [30]. The Boolean function for an *MV3* with inputs *A*, *B*, and *C* and output *F* is  $F = MV3(A, B, C) = AB + AC + BC$ , which tends to express the majority of three inputs. By fixing one input in binary 0 or 1, a two-input AND or OR gate will be realized. With a diagonal configuration of two cells, an inverter for producing the inverse of an input signal is then realized, as shown in Fig. 1(c).

The clock in QCA is performed by an electric field, which plays two crucial roles including supplying energy for cells and controlling data flow in circuits [3, 7]. The quasi-adiabatic clock mechanism with four zones was introduced to ensure a circuit remaining in instantaneous ground state. As shown in Fig. 2, each clock is composed of four cyclical phases: switch, hold, release, and relax. Taking the clock0 as an example, the inter-dot barrier will gradually increase during the switch phase from  $t=0$  to  $t=\pi/2$  and then peaks in the hold phase, resulting in confining the electrons in quantum dots. Thus, this configuration can encode binary 0 or 1 and polarize neighboring cells in clock1. After the hold phase of clock0, the barrier continues to decrease during the release phase, so that the tunneling probability of electrons accordingly increases. Meanwhile, the cells in clock1 are in the hold phase. When the inter-dot barrier reaches the minimum value, the cells in clock0 completely lose their polarizations and do not have any influence on neighboring cells while getting ready for next cycle at  $t=2\pi$ . The signals propagated from clock0 to clock3 experience one clock cycle. Besides, the phase difference between two adjacent input bits is exactly one clock cycle as well in QCADesigner [16].



**Fig. 3.** An SCCC encoder. (a) Overall diagram. (b) Parity bit generator diagram of outer (7, 4, 1) BCH code encoder and modulo-2 adder diagram. (c) Pseudo random interleaver model. (d) Inner (2, 1, 2) CC encoder diagram and state transition model.

### 3. Design principles of SCCCs

#### 3.1. Overall circuit structure

An SCCC encoder serially consists of an outer encoder (OE), a pseudo random interleaver (PRI) and an inner encoder (IE) whose input is a permuted codeword of the OE by the PRI [28]. The overall diagram for an SCCC encoder is shown in Fig. 3(a), of which the OE can be a CC or BC encoder, while the IE may as well use a CC encoder. The coding process is as follows: an original message string  $A$  is sent to an OE and coded as a systematic codeword  $C$  by adding parity bits into the message  $A$ ; the codeword  $C$  is then fed into and permuted by a PRI to a bit string  $I$ ; finally, the string  $I$  is coded by a CC encoder as codewords  $Y_0$  and  $Y_1$ . We employ a BCH code encoder for the OE and a CC encoder for the IE in this paper, respectively.

#### 3.2. Outer encoder

BCH codes are one type of the cyclic codes for correcting multiple burst errors, which are described by the Galois fields ( $GF$ s) and generator polynomials [31, 32]. Taking an example in  $GF(2^3)$ , the primitive polynomial is  $p(x) = x^3 + x + 1$ ; the primitive element is  $\alpha$ ; we have  $\alpha^3 + \alpha + 1 = 0$ , that is  $\alpha^3 = \alpha + 1$ . One will access  $\alpha^4 = \alpha \cdot \alpha^3 = \alpha^2 + \alpha$ ,  $\alpha^5 = \alpha^2 + \alpha + 1$ ,  $\alpha^6 = \alpha^2 + 1$ , and  $\alpha^7 = \alpha \cdot \alpha^6 = 1$ . As a result, there are eight elements  $\{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$  in  $GF(2^3)$ . For a (7, 4, 1) BCH code encoder with ECC  $t=1$ , the generator polynomial is  $g(x) = m_1(x)$ , and the conjugate root class of  $m_1(x)$  is  $\{\alpha, \alpha^2, \alpha^4\}$ . Therefore, the code generator polynomial is  $g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^4) = x^3 + x + 1$ . The message polynomial and parity polynomial for a message string  $(A_3, A_2, A_1, A_0)$  are  $A(x) =$

$$A_3x^3 + A_2x^2 + A_1x + A_0 \quad \text{and} \quad R(x) = R_2x^2 + R_1x + R_0 = [A(x)x^3] \bmod [g(x)],$$

respectively. By adding this parity polynomial into the message polynomial  $A(x)$ , the codeword  $C(x) = A(x)x^3 + [A(x)x^3] \bmod [x^3 + x + 1]$  is achieved.

Fig. 3(b) depicts the proposed layout for the parity bit generator of a (7, 4, 1) BCH code encoder, in which each square with a symbol  $\oplus$  is a modulo-2 adder (realized by an XOR gate) as shown in the inset surrounded by a rectangle. This generator utilizes the idea for a divider design, where three 0s in the dividend ( $A_3, A_2, A_1, A_0, 0, 0, 0$ ) are generated by shifting  $(A_3, A_2, A_1, A_0)$  to left by three bits and the divisor is the coefficients  $(g_3, g_2, g_1, g_0)$  of the code generator polynomial. In each computing stage, no matter what the most significant bit of the dividend is, latter three bits are of interest. The calculated remainder is then shifted by one bit to right. Further, because the coefficients of the code generator polynomial are constant, all AND gates can be deleted. For example, in the first stage,  $g_2 = 0$ , the output of a AND gate is certainly 0, so the output of a modulo-2 adder is equal to  $A_2$ ; similarly, for  $g_1 = 1$ , the output of a AND gate becomes  $A_3$ , so the output of an adder depends on the XOR operation on  $A_3$  and  $A_1$ . The full circuit for a (7, 4, 1) BCH code encoder needs a serial-to-parallel converter for generating parallel bits from a message string and a parallel-to-serial converter for connecting parity bits and the message. Regarding the modulo-2 adder or XOR gate, although it was comprehensively studied, complex crossovers or other logic gates are used in previous designs [33-36]. With a NAND-NOR-Inverter (NNI) gate and a five-input majority voter (MV5) in [37, 38], a coplanar layout for the XOR gate is proposed. This layout with fewer logic gates not only removes crossovers but also reduces power dissipation compared with its counterparts.

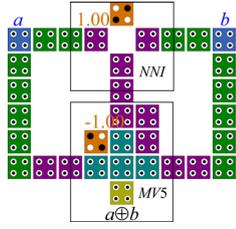


Fig. 4. A modulo-2 adder in QCA.

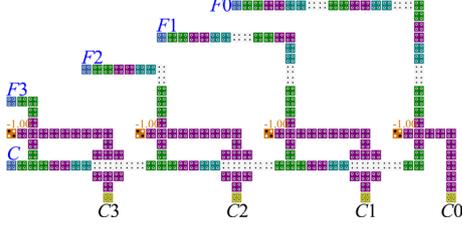


Fig. 5. A 4-bit serial-to-parallel converter in QCA.

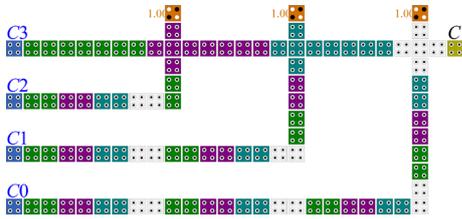


Fig. 6. A 4-bit parallel-to-serial converter in QCA.

### 3.3. Interleaver

An interleaver is a data mixer to disorganize input signal bits [28, 39]. Fig. 3(c) illustrates a model for a 7-bit PRI. A bit string  $C = (C_6, C_5, C_4, C_3, C_2, C_1, C_0)$  should be parallelized into separate bits  $C_6 C_5 \dots C_0$ , where  $C_6$  and  $C_0$  are the most and least significant bits, respectively. The parallelized bits are then swapped by using this pattern that turns the order of these bits upside down in this paper. A parallel-to-serial converter serializes the permuted bits into a bit string  $I$  in the end. Hence, the design for serial-to-parallel converter and parallel-to-serial converter (abbreviated as SPC and PSC) is the core in obtaining the PRI. A side note is that this interleaver can be extended and used as a Last-In-First-Out (LIFO) converter.

### 3.4. Inner encoder

A  $(u, f, K)$  CC encoder is utilized as the IE, where  $u, f$  and  $K+1$  signify the output count, input count and constraint length of the encoder, respectively. The code rate is  $f/u$ , which means that an  $f$ -bit message string is encoded as a  $u$ -bit codeword. The description methods that correspond to different decoding algorithms include a generator polynomial matrix, the state diagram and trellis diagram for CC encoders [40]. Fig. 3(d) displays the coding circuit and state transfer diagram for a  $(2, 1, 2)$  CC encoder, which frequently appears in many CC designs. In this circuit,  $D_1$  and  $D_2$  are the states of two 1-bit shift registers. These registers can be implemented by a wire with four serial clock

zones in QCA. Each square with a symbol  $\oplus$  is a modulo-2 adder proposed in the last subsection. The generator polynomial matrix of this encoder is  $Y = [Y_0, Y_1] = [I + D_1 + D_2, I + D_2]$ , where the symbol  $+$  denotes modulo-2 addition [41]. In this matrix, the first element indicates that the codeword  $Y_0$  relates to input bits  $I, I_{t-1}$  and  $I_{t-2}$  and the second means that the codeword  $Y_1$  involves input bits  $I$  and  $I_{t-2}$ . With regard to the state diagram, two binary bits in circles are the states of two shift registers; dashed or solid arrow lines indicate input bit 0 or 1; the values near arrow lines are two codewords  $Y_0$  and  $Y_1$ , respectively. The diagram provides the state transition process, codewords and states of shift registers for each input bit.

## 4. Design and implementation of an SCCC encoder in QCA

In this section, we will present the proposed circuit design in QCA technology. The components are illustrated and logically connected for producing an SCCC encoder.

### 4.1. Modulo-2 adder

To get a coplanar structure of modulo-2 adders, the *NNI* and *MV5* gates are used as shown in Fig. 4. The logic function for a *NNI* gate with inputs  $a, b$  and  $c$  is  $NNI(a, b, c) = \bar{a}\bar{b} + \bar{a}c + \bar{b}c$ , which realizes NOT operation in pins  $a$  and  $b$ . We obtain the logic function of XOR  $a \oplus b = MV5(a, b, NNI(a, b, 1), NNI(a, b, 1), 0)$ . This implementation has 34 cells, occupies  $0.0288\mu\text{m}^2$  and 0.75 clock cycles. The principles that two-cell spacing and at least two cells in each clock zone are established to avoid sneak noise between wires and keep cells stable. Note that the input and output pins are not surrounded by other cells, leading to being non-essential to consider crossovers. Additionally, various-bit parity checkers can easily be constructed by hierarchically connecting XOR gates.

### 4.2. Serial-to-parallel and parallel-to-serial converters

Previous works on SPC and PSC are based on an additional special clock to control data flow. Four pins of the SPC designed in [22] will generate the same bit strings and two adjacent strings are shifted by one clock cycle if the special clock is missing. Moreover, the phase difference between two neighboring input bits is one clock cycle, which provides clues to construct an SPC and PSC. We define a filtration control matrix (*FCM*) as a unitary matrix whose dimension is equal to the length of the input bit string to be parallelized. Fig. 5 shows a 4-bit SPC in QCA, of which the *FCM* is a four-dimensional unitary matrix as equation(1).

$$FCM = \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

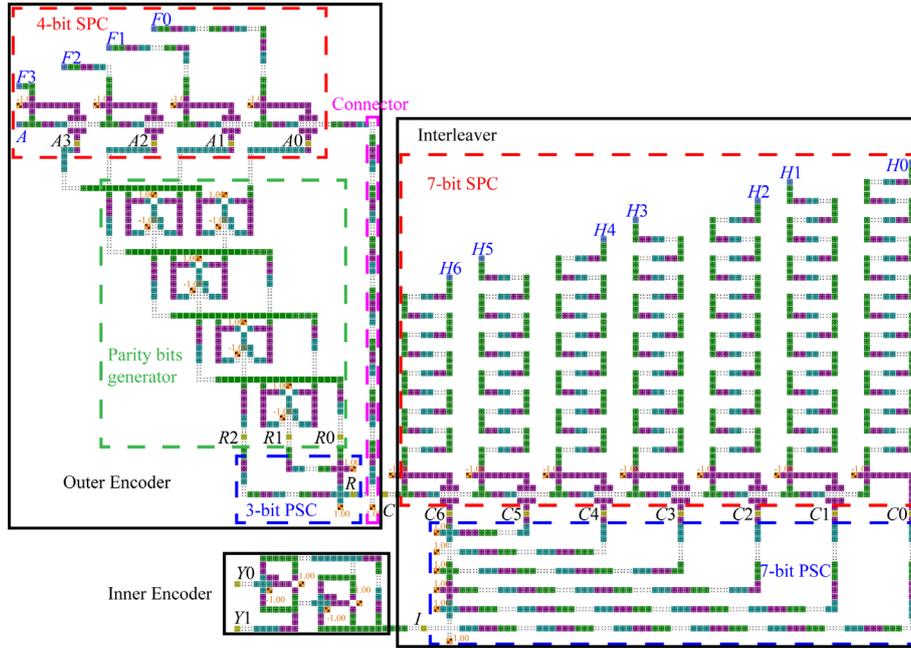


Fig. 7. Full circuit of an SCCC encoder in QCA.

The string  $C = (C_3, C_2, C_1, C_0)$  accompanies the  $FCM$  as inputs, where  $C_3$  and  $C_0$  are the most and least significant bits, respectively. We first input  $C_0$ , and then  $C_1$ ,  $C_2$ , and  $C_3$  in turn in QCADesigner. Input bits  $C_3$  and  $F_3$  arrive at the leftmost AND gate when  $C_0$  and  $F_0$  reach the rightmost AND gate, so that the parallelized outcomes keep synchronous. With the 0s in each row of the  $FCM$  and AND gates, the undesired output bits will be masked. In addition, the crossovers using clock1 and clock3 are implemented for a coplanar layout, where six cells in clock1 are used for avoiding interference while strengthening the signals.

Fig. 6 displays a 4-bit PSC for serializing four separate bits  $C_0, C_1, C_2$  and  $C_3$  into a bit string  $C = (C_3, C_2, C_1, C_0)$ , where  $C_3$  and  $C_0$  are the most and least significant bits, respectively. Although four bits are simultaneously fed into this converter, bit  $C_3$  gets to output pin  $C$  first because the delay of two neighboring transmission lines is one clock cycle. OR gates play the role in making a selection between input bits and undesired signals. This layout can readily be changed for firstly generating bit  $C_0$  by accordingly adjusting the clock delay of wires.

#### 4.3. Full circuit of an SCCC encoder

In this subsection, an SCCC encoder consisting of an OE, a PRI and an IE will be implemented with QCA cells. Fig. 7 shows the full circuit of an SCCC encoder, in which three main components are surrounded by black rectangles and marked with text. A  $(7, 4, 1)$  BCH code encoder is applied to the OE. There are four steps for depicting its work process. First, a 4-bit message string  $A = (A_3, A_2, A_1, A_0)$  to be encoded is parallelized into four isolated bits by using a 4-bit SPC, where input  $[F_0; F_1; F_2; F_3]$  is a four-dimensional unitary matrix. Next, four parallelized bits are fed into a parity bit generator. This generator has four computing stages, of which each stage consumes one clock delay.

In the parity bit generator, each computing stage uses 3 modulo-2 adders, while its implementation utilizes five adders in total because of the materialized coefficients of generator polynomial and three 0s generated by shifting input string to left by three bits. Then, a 3-bit PSC is connected to the endpoint of this generator for serializing the computed remainder  $R_2, R_1, R_0$  into a bit string  $R$ . Finally, an OR gate, which we called a connector and marked with pink, is employed to connect the string  $R$  and the message  $A$  for making up a systematic codeword  $C$  that will be put into the PRI next.

For a PRI, a 7-bit SPC and a 7-bit PSC are serially connected, where  $[H_0; H_1; H_2; H_3; H_4; H_5; H_6]$  is a seven-dimensional  $FCM$  for filtering unfavorable data. The serpentine lines are just used to make signals synchronous. The bit string  $C$  is parallelized, then permuted, and serialized in the end, generating a disorganized bit string  $I$ . The data exchange model is already mentioned and completed by employing the 7-bit PSC. Hence, the PSC serves for two purposes including serializing the parallelized bits and turning these bits upside down. One can conveniently realize other models by interchanging the clock delay of output wires of the PSC. Taking the swapping of  $C_3$  and  $C_2$  in string  $I$  as an example, extending four clock cycles to five for  $C_3$  and shortening five clock cycles to four for  $C_2$  will effectively achieve this function.

We employ a  $(2, 1, 2)$  CC encoder in QCA to explain the principle of IEs. The input string of this encoder is the string  $I$  and two codewords are  $Y_0$  and  $Y_1$ . As we can see from its generator polynomial matrix  $Y = [Y_0, Y_1] = [I + D_1 + D_2, I + D_2]$ , two elements have a factor  $I + D_2$ , which is also used to create the codeword  $Y_1$ . That is to say, one can realize  $Y_1$  first and  $Y_0$  will be produced next. It is worth noting that only in this way can this encoder be constructed with two modulo-2 adders. This encoder holds two clock cycles at most and gets a superior optimization compared with counterparts in [24].

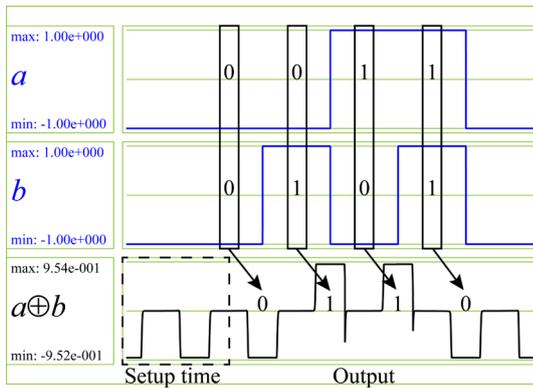


Fig. 8. Functional simulation results for a modulo-2 adder.

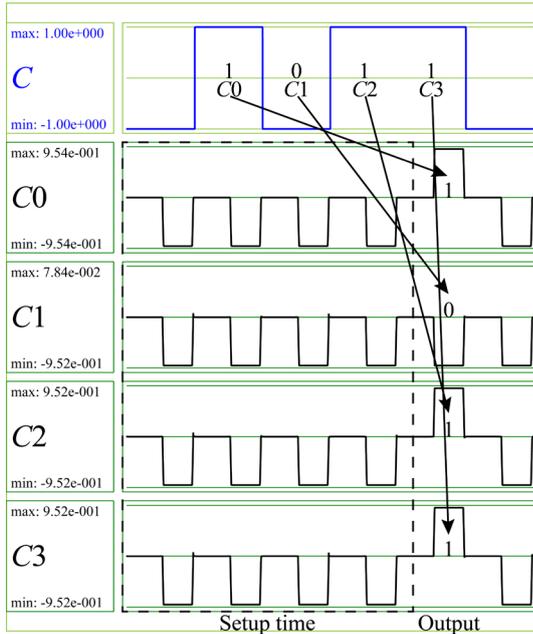


Fig. 9. Functional simulation results for a serial-to-parallel converter.

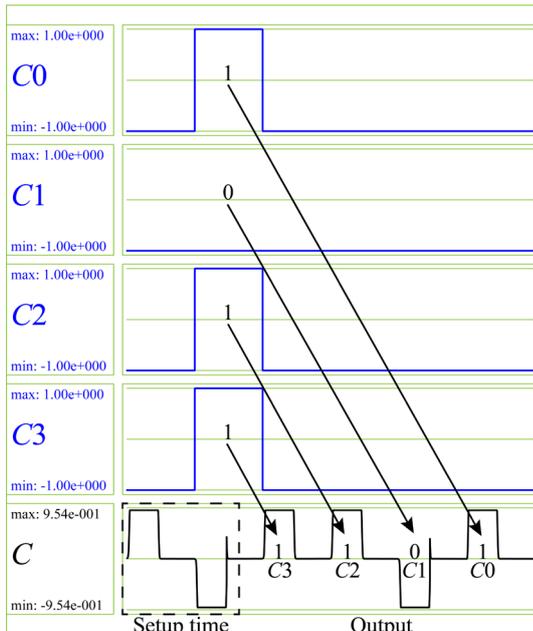


Fig. 10. Functional simulation results for a parallel-to-serial converter.

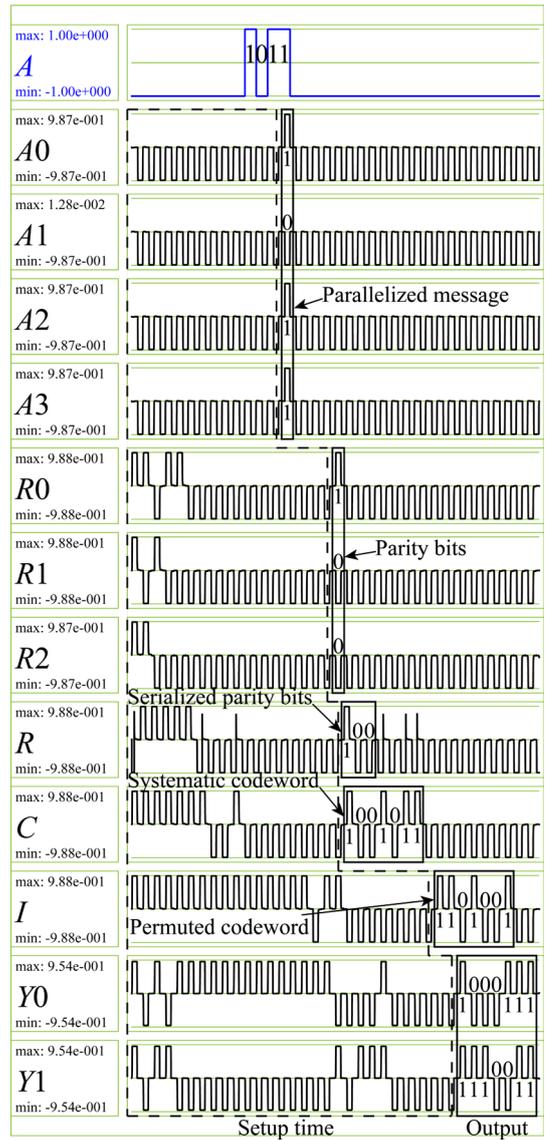


Fig. 11. Functional simulation results for an SCCC encoder.

## 5. Results

This section presents the simulation and analysis results for the circuit design proposed in the last section. The waveforms produced by utilizing the bistable approximation simulation engine in QCA Designer verify the validity of these circuits, while performance figures account for their physical properties. The testing samples and their results are presented in detail. The QCA costs for various BCH code and CC encoders are also derived.

### 5.1. Modulo-2 adder

We first present the simulation results for the modulo-2 adder or XOR gate, as shown in Fig. 8. Two input vectors and output vector are labelled and marked with the text. It is easy to test its correctness with this figure. Here, setup time means that adequate 0s are put into the circuit to set all cells in state 0 for the test examples.

**Table 1**

Performance figures of the designed circuits.

Circuits	Area ( $\mu\text{m}^2$ )	Delay (clocks)	Cell	Crossover
Modulo-2 adder	0.0288	0.75	34	0
4-bit SPC	0.3192	3.50	184	3
4-bit PSC	0.1248	1.00	98	0
SCCC encoder	5.4320	19.00	2300	18 (10/8)

**Table 2**

Performance figures of various modulo-2 adders.

Modulo-2 adders	Area ( $\mu\text{m}^2$ )	Delay (clocks)	Cell	Crossover
[33]	0.0528	1.00	45	0
[34]	0.0396	0.75	48	0
[35]	0.0216	0.75	28	0
[36]	0.0252	1.00	32	0
Proposed	0.0196	0.75	27	0

**Table 3**

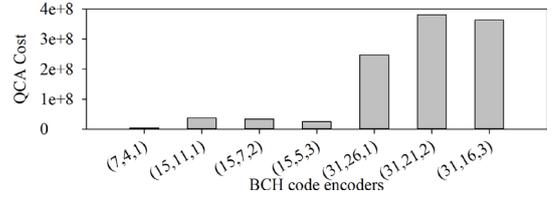
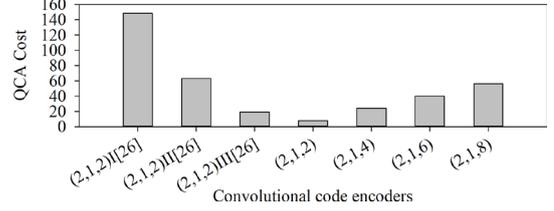
Performance figures of various BCH coder encoders.

Encoders	Area ( $\mu\text{m}^2$ )	Delay (clocks)	Cell	XOR	M/3	Crossover
(7,4,1)	1.7936	9.00	747	5	7	12 (4/8)
(15,11,1)	9.2192	23.25	2922	18	15	44 (11/33)
(15,7,2)	5.8604	16.25	2790	20	15	56 (7/49)
(15,5,3)	6.1308	12.75	2742	20	15	50 (5/45)
(31,26,1)	46.8488	53.50	11730	47	31	130 (26/104)
(31,21,2)	35.0520	44.75	12449	116	31	210 (21/189)
(31,16,3)	31.5732	36.00	13604	145	31	240 (16/224)

**Table 4**

Performance figures of various CC encoders

Encoders	Area ( $\mu\text{m}^2$ )	Delay (clocks)	Cell	XOR	M/3	Crossover
(2,1,2) I in [24]	0.3400	6.50	228	0	12	0
(2,1,2) II in [24]	0.7100	4.00	378	0	8	0
(2,1,2) III in [24]	0.3400	3.50	239	0	6	0
(2,1,2)	0.1104	2.00	114	2	0	0
(2,1,4)	0.4800	3.00	273	4	0	0
(2,1,6)	0.5376	4.00	373	5	0	0
(2,1,8)	0.9648	4.00	552	7	0	0

**Fig. 12.** QCA cost for BCH code encoders.**Fig. 13.** QCA cost for convolutional code encoders.

## 5.2. Serial-to-parallel and parallel-to-serial converters

We then show the results (Fig. 9 and Fig. 10) for the 4-bit SPC and 4-bit PSC, respectively. A serial input  $C = (C_3, C_2, C_1, C_0) = (1, 1, 0, 1)$  is used to test the function of the SPC to parallelize the input into four parallel bits surrounded by a red rectangle. The order of four bits in the input string in simulation waveform is reversed from its written form because we first feed  $C_0$  and then  $C_1, C_2$  and  $C_3$  into the circuit in QCADesigner. The parallel four bits 1, 1, 0, 1 are also serialized for testing the PSC. The serialized output vector is marked as well and the signal bit  $C_3$  is exported first. The delay counted from the first input bit to output for the SPC is 3.5 clock cycles, while it is 1.0 cycle for the PSC. Setup time in these figures has same meaning as mentioned before.

## 5.3. Full circuit of an SCCC encoder

For the SCCC encoder, several test points in the full circuit are chosen to verify its functionalities by successive steps as follows (Fig. 11). By using a 4-bit SPC, a message string  $A = (A_3, A_2, A_1, A_0) = (1, 1, 0, 1)$  to be encoded in the OE is first parallelized into four separate bits, which are then passed onto a parity bit generator. The order of input bits is opposite to its expression as described before. A (7, 4, 1) BCH code encoder generates three parity bits 0, 0, 1, which ought to follow the original message string for making up a systematic codeword. Therefore, a 3-bit PSC is utilized for serializing the parity bits into a string  $R = (R_2, R_1, R_0) = (0, 0, 1)$  (in a reversed order as in the figure). After that, a connector plays the role in linking the message string and string  $R$  into  $C = (A_3, A_2, A_1, A_0, R_2, R_1, R_0) = (1, 1, 0, 1, 0, 0, 1)$  (in a reversed order as in the figure) that is the codeword of the outer (7, 4, 1) BCH code encoder. The second stage is that the codeword of the OE is poured into a PRI for generating a permuted codeword  $I = (A_3, A_2, A_1, A_0, R_2, R_1, R_0) = (1, 1, 0, 1, 0, 0, 1)$  (in the same order as in the figure). Again, the function of the PRI is turning the order of the codeword of the OE upside down. Finally, the inner codeword  $(Y_0 Y_1) = (11, 01, 01, 00, 10, 11, 11)$  is produced by the (2, 1, 2) CC encoder, as labelled with the text.

We begin to quantify the performance figures of the designed circuits to evaluate their physical properties. Table 1 lists several figures including the area, delay, cell count and crossover count of the circuits presented in section IV. For the modulo-2 adder, aforementioned principles are carried out to avoid interference between wires, whereas previous designs do not comply with those rules. It enables us to simplify the adder design.

Table 2 displays the physical features of the renewed XOR gate and its counterparts. The results show that the proposed design is more efficient than others in the considered figures. For the two converters, these values will gradually increase with the bit number. More specifically, the area, delay and number of cells for an  $n$ -bit SPC are  $(144n^2 + 264n - 168) \times 10^{-4} \mu\text{m}^2$ ,  $(2n - 1)/2$  clock cycles and  $4n^2 + 34n - 16$ , while they are  $(84n^2 - 24n) \times 10^{-4} \mu\text{m}^2$ ,  $n/4$  clock cycles and  $5n^2 + 6n - 6$  for an  $n$ -bit PSC, respectively. In addition, the number of coplanar crossovers for an  $n$ -bit SPC is  $n - 1$ . The SCCC encoder gets large values in these terms because it is composed of the above components. The crossover count for the SCCC encoder is 18 including 10 coplanar crossings and 8 multi-layer structures.

Table 3 summarizes the performance figures of different BCH code encoders. The area and cell count for BCH code encoders with various ECCs roughly show irregular fluctuation. The delay of  $(n, k, t)$  encoders decreases with increasing ECC. To be exact, the delay of an  $(n, k, t)$  BCH code encoder is  $k + k + (n - k + 1)/4$ , where the first  $k$  is the delay of a  $k$ -bit SPC; the second indicates the delay of a parity bit generator and the third term means the total delay of a  $(n - k)$ -bit PSC and a connector. Similarly, the number of XOR gates relates to the structure of an encoder and the supplementary 0s generated by shifting the message string. The XOR count in a  $(n, k, t)$  BCH code encoder is  $gk - n$ , where  $g$  is the number of nonzero coefficients of its generator polynomial. Here, both SPC and PSC consist of  $MV3$ s, whose count is equal to  $n$  in total. The last column represents the number of utilized crossovers, where the values in parentheses are the number of coplanar and multi-layer crossovers, whose counts are  $k$  and  $k(n - k - 1)$ , respectively.

The case for the IEs is simpler because they consist of modulo-2 adders without crossovers in Table 4. Their performance figures gradually increase with their constraint length. The expressions for these figures of a  $(u, f, K)$  CC encoder depend on its concrete conditions, such as the generator polynomial matrix. The last three rows show the performance of three  $(2, 1, 2)$  CC encoders in [24]. These designs mainly depend on OR and AND gates, which obviously reduced their performance. With the proposed method, the designed CC encoders do not only get superior optimization in circuit area, delay and cell count, but also consume fewer modulo-2 adders that consist of  $NNI$  and  $MV5$  gates.

Besides the above considered criteria, the concept of QCA cost was used to evaluate the quality of a system [17]. The cell count of a

QCA circuit is generally proportional to its area, while the area mainly relates to the utilized crossover types. Moreover, both logic gates and wires are implemented by using QCA cells. The cost takes these aspects into consideration and is given by  $\text{Cost} = (M^x + I + C^y) \times L^z$ , where  $M, I, C$  and  $L$  are respectively the number of majority gates, inverters, crossovers and latency;  $x, y$ , and  $z$  are the exponential weights for these parameters. We extend this equation by taking it into account the  $NNI$  and  $MV5$  gates because both of them realize simple logical functions as the  $MV3$ . Additionally, a multilayer crossing consumes more cost than a coplanar crossover; for instance,  $C_m = 3C_c$ , where  $C_m$  and  $C_c$  are the costs of a three-layer crossover and a coplanar crossover, respectively. Fig. 12 and Fig. 13 show the calculated QCA costs for two types of encoders. We can see that the cost of BCH code encoders shows great diversity in  $GF(2^m)$  with increasing ECC, which is jointly resulted from the decreasing delay and increasing number of crossovers. Varying changes in their values induce the irregular fluctuation. With the increase of constraint length of  $(2, 1, K)$  encoders without crossovers, it is evident that the number of logic components and delay progressively accrues. Therefore, the gradually increasing cost of CC encoders appears with incremental constraint length. In addition, the proposed  $(2,1,2)$  CC encoder gets a significant optimization compared with the three schemes in [24] in respect of QCA cost.

## 6. Conclusions

This paper proposes a layout for serial concatenated convolutional code (SCCC) encoders in quantum-dot cellular automata (QCA) technology. An SCCC encoder is made up of a Bose-Chaudhuri-Hocquenghem (BCH) code encoder, a pseudo random interleaver and a convolutional code encoder. We employ a  $(7, 4, 1)$  BCH code encoder as the outer encoder and a  $(2, 1, 2)$  convolutional code encoder as inner encoder to illustrate the design method of SCCC encoders. To be specific, a scheme for BCH code encoders is proposed and the encoders with various error correction capacities are implemented in QCA. The convolutional code encoders with various constraint lengths are investigated as well. For data swapping, we design a pseudo-random interleaver, which consists of a serial-to-parallel converter and a parallel-to-serial converter. It is worth noticing that these converters are implemented by using the four-phase clock mechanism in QCA. The simulation waveforms and performance figures verified their functionalities, credibility and physical properties. The QCA cost of BCH code encoders presents great diversity with increasing error correcting capability, while the cost of convolutional code encoders will increase with incremental constraint length. The circuit design of three components of SCCC encoder in this paper are rudimentary. Future work will be considered to design more efficient practical components for communication applications, i.e. universal or serial BCH encoders and block interleaver.

## Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 61271122).

## References

- [1] P.D. Tougaw, C.S. Lent, Logical devices implemented using quantum cellular automata, *J. Appl. Phys.* 75(3) (1994) 1818-1825.
- [2] F. Karim, K. Walus, Efficient simulation of correlated dynamics in quantum-dot cellular automata (QCA), *IEEE Trans. Nanotechnol.* 13(2) (2014) 294-307.
- [3] J. Timler, C.S. Lent, Power gain and dissipation in quantum-dot cellular automata, *J. Appl. Phys.* 91(2) (2002) 823-831.
- [4] L. Livadaru, P. Xue, Z. Shaterzadeh-Yazdi, G.A. DiLabio, J. Mutus, J.L. Pitters, B.C. Sanders, R.A. Wolkow, Dangling-bond charge qubit on a silicon surface, *New J. Phys.* 12 (2010) 1-15.
- [5] H. Kawai, F. Ample, Q. Wang, Y.K. Yeo, M. Saeys, C. Joachim, Dangling-bond logic gates on a Si(100)-(2 x 1)-H surface, *J. Phys.-Condes. Matter* 24(9) (2012) 1-13.
- [6] R. Wang, A. Pulimeno, M.R. Roch, G. Turvani, G. Piccinini, M. Graziano, Effect of a clock system on bis-ferrocene molecular QCA, *IEEE Trans. Nanotechnol.* 15(4) (2016) 574-582.
- [7] C.S. Lent, P.D. Tougaw, A device architecture for computing with quantum dots, *Proc. IEEE* 85(4) (1997) 541-557.
- [8] S.C. Henderson, E.W. Johnson, J.R. Janulis, P.D. Tougaw, Incorporating standard CMOS design process methodologies into the QCA logic design process, *IEEE Trans. Nanotechnol.* 3(1) (2004) 2-9.
- [9] P. Wang, M.Y. Niamat, S.R. Vemuru, M. Alam, T. Killian, Synthesis of majority/minority logic networks, *IEEE Trans. Nanotechnol.* 14(3) (2015) 473-483.
- [10] K. Kong, Y. Shang, R. Lu, An optimized majority logic synthesis methodology for quantum-dot cellular automata, *IEEE Trans. Nanotechnol.* 9(2) (2010) 170-183.
- [11] D. Abedi, G. Jaberipur, Decimal full adders specially designed for quantum-dot cellular automata, *IEEE Trans. Circuits Syst. II Express Briefs* 65(1) (2018) 106-110.
- [12] G. Turvani, M. Bollo, M. Vacca, F. Cairo, M. Zamboni, M. Graziano, Design of mram-based magnetic logic circuits, *IEEE Trans. Nanotechnol.* 16(5) (2017) 851-859.
- [13] M. Kianpour, R. Sabbaghi-Nadooshan, A novel quantum-dot cellular automata X-bit  $\times$  32-bit SRAM, *IEEE Trans. Very Large Scale Integr. VLSI Syst.* 24(3) (2016) 827-836.
- [14] V. Vankamamidi, M. Ottavi, F. Lombardi, Two-dimensional schemes for clocking/timing of QCA circuits, *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 27(1) (2008) 34-44.
- [15] C.A.T. Campos, A.L. Marciano, O.P.V. Neto, F.S. Torres, USE: A universal, scalable, and efficient clocking scheme for QCA, *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 35(3) (2016) 513-517.
- [16] K. Walus, T.J. Dysart, G.A. Jullien, R.A. Budiman, QCADesigner: A rapid design and simulation tool for quantum-dot cellular automata, *IEEE Trans. Nanotechnol.* 3(1) (2004) 26-31.
- [17] W. Liu, L. Lu, M. Oneill, E.E. Swartzlander, A first step toward cost functions for quantum-dot cellular automata designs, *IEEE Trans. Nanotechnol.* 13(3) (2014) 476-487.
- [18] S. Srivastava, S. Sarkar, S. Bhanja, Estimation of upper bound of power dissipation in QCA circuits, *IEEE Trans. Nanotechnol.* 8(1) (2009) 116-127.
- [19] F. Riente, G. Turvani, M. Vacca, M.R. Roch, M. Zamboni, M. Graziano, ToPoliNano: A CAD tool for nano magnetic logic, *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 36(7) (2017) 1061-1074.
- [20] A. Pulimeno, M. Graziano, A. Sanginario, V. Cauda, D. Demarchi, G. Piccinini, Bis-ferrocene molecular QCA wire: Ab initio simulations of fabrication driven fault tolerance, *IEEE Trans. Nanotechnol.* 12(4) (2013) 498-507.
- [21] G. Xie, Y. Xiang, Y. Zhang, S. Liu, H. Lv, Design and implementation of encoding and check code circuit with hamming code on QCA, *Int. J. Unconv. Comput.* 10(5-6) (2014) 391-404.
- [22] D.S. Silva, L.H.B. Sardinha, M.A.M. Vieira, L.F.M. Vieira, O.P.V. Neto, Robust serial nanocommunication with QCA, *IEEE Trans. Nanotechnol.* 14(3) (2015) 464-472.
- [23] M. Awais, M. Vacca, M. Graziano, M.R. Roch, G. Masera, Quantum dot cellular automata check node implementation for LDPC decoders, *IEEE Trans. Nanotechnol.* 12(3) (2013) 368-377.
- [24] M. Zhang, L. Cai, X. Yang, H. Cui, Z. Wang, Implementation of convolutional encoder in quantum-dot cellular automata, *Key Eng. Mater.* 645-646 (2015) 1078-1084.
- [25] C. Berrou, A. Glavieux, P. Thitimajshima, Near shannon limit error-correcting coding and decoding: Turbo-codes (1), *Proceedings of ICC '93 - IEEE International Conference on Communications*, 1993, pp. 1064-1070.
- [26] M. Zhang, L. Cai, X. Yang, H. Cui, C. Feng, Design and simulation of turbo encoder in quantum-dot cellular automata, *IEEE Trans. Nanotechnol.* 14(5) (2015) 820-828.
- [27] C.R. Graunke, D.I. Wheeler, D. Tougaw, J.D. Will, Implementation of a crossbar network using quantum-dot cellular automata, *IEEE Trans. Nanotechnol.* 4(4) (2005) 435-440.
- [28] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding, *IEEE Trans. Inf. Theory* 44(3) (1998) 909-926.
- [29] M. Taucer, F. Karim, K. Walus, R.A. Wolkow, Consequences of many-cell correlations in clocked quantum-dot cellular automata, *IEEE Trans. Nanotechnol.* 14(4) (2015) 638-647.
- [30] M. Momenzadeh, J. Huang, M.B. Tahoori, F. Lombardi, Characterization, test, and logic synthesis of And-Or-Inverter (AOI) gate design for QCA implementation, *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 24(12) (2005) 1881-1892.
- [31] J. Cho, W. Sung, Efficient software-based encoding and decoding of BCH codes, *IEEE Trans. Comput.* 58(7) (2009) 878-889.
- [32] R.F.T. El-Din, R.M. El-Hassani, S.H. El-Ramly, A novel high-speed systematic encoder for long binary cyclic codes, *IEEE Commun. Lett.* 17(5) (2013) 984-987.
- [33] M. Poorhosseini, A.R. Hejazi, A fault-tolerant and efficient XOR structure for modular design of complex QCA circuits, *J. Circuits Syst. Comput.* 27(7) (2017) 1-24.
- [34] M. Rahimpour Gadim, N. Jafari Navimipour, A new three-level fault tolerance arithmetic and logic unit based on quantum dot cellular automata, *Microsyst. Technol.* 24(2) (2017) 1-11.
- [35] G. Singh, R.K. Sarin, B. Raj, A novel robust Exclusive-OR function implementation in QCA nanotechnology with energy dissipation analysis, *J. Comput. Electron.* 15(2) (2016) 455-465.

- [36] S. Sheikhaal, S. Angizi, S. Sarmadi, M.H. Moaiyeri, S. Sayedsalehi, Designing efficient QCA logical circuits with power dissipation analysis, *Microelectron. J.* 46(6) (2015) 462-471.
- [37] K. Navi, R. Farazkish, S. Sayedsalehi, M. Rahimi Azghadi, A new quantum-dot cellular automata full-adder, *Microelectron. J.* 41(12) (2010) 820-826.
- [38] B. Sen, B. Sikdar, Characterization of universal NAND-NOR-inverter QCA gate, *Proceedings of 11th IEEE VLSI Design and Test Symposium, Kolkata, 2007*, pp. 433-442.
- [39] Y. Jungpil, M.L. Boucheret, R. Vallet, A. Duverdier, G. Mesnager, Interleaver design for serial concatenated convolutional codes, *IEEE Commun. Lett.* 8(8) (2004) 523-525.
- [40] Q.T.Z. Keith, Channel coding, in: Q.T.Z. Keith (Ed.), *Wireless communications: Principles, theory and methodology*, John Wiley & Sons, England, 2015, pp. 121-170.
- [41] S. Moriya, K. Kikuchi, H. Sasano, Efficient search for high-rate punctured convolutional codes using dual codes, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* E99-A(12) (2016) 2162-2169.