Design of a fault-tolerant reversible control unit in molecular quantum-dot cellular automata

Golnaz Bahadori*, Monireh Houshmand* ‡ and Mariam Zomorodi-Moghadam†

*Department of Electrical Engineering, Imam Reza International University, Mashhad, Iran
†Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
‡m.hooshmand@imamreza.ac.ir

Received 2 November 2017
Accepted 17 December 2017
Published 22 January 2018

Quantum-dot cellular automata (QCA) is a promising emerging nanotechnology that has been attracting considerable attention due to its small feature size, ultra-low power consuming, and high clock frequency. Therefore, there have been many efforts to design computational units based on this technology. Despite these advantages of the QCA-based nanotechnologies, their implementation is susceptible to a high error rate. On the other hand, using the reversible computing leads to zero bit erasures and no energy dissipation. As the reversible computation does not lose information, the fault detection happens with a high probability. In this paper, first we propose a fault-tolerant control unit using reversible gates which improves on the previous design. The proposed design is then synthesized to the QCA technology and is simulated by the QCADesigner tool. Evaluation results indicate the performance of the proposed approach.

Keywords: Quantum-dot cellular automata; reversible computing; fault-tolerant QCA circuit; reversible control unit; fault-tolerant reversible logic.

1. Introduction

In the recent decade, the design of integrated circuits has had significant development in critical features such as reducing transistor size, reducing power consumption, and increasing functionality. However, based on Moore’s law for CMOS technology, many fundamental physical limitations have been anticipated and hence, emerging technologies have been considered as alternative computing approaches. In consideration of quantum effects which can be appeared at nanoscale, quantum-dot cellular automata (QCA) as a promising emerging nanotechnology which works based on quantum effects can be an appropriate approach. QCA works based on coulombic interactions which are among adjacent neighboring cells. Coulombic interactions
cause a change in the state of polarization and hence, lead to binary information transfer.

Polarized states are dependent on the location of electrons in each of the four electron holes which are at the corners of the square QCA cell.

QCA technology first was proposed by Lent et al. in 1993.¹ Same as other computational nanotechnologies, defects may occur in the synthesis and deposition phases of molecular QCA production process. However, the probability of occurring defects in the deposition phase is higher.² It is necessary to make sure that system is working properly, so it would be important that a system has the ability for being tested and have a low error rate.

In this paper, we propose a design for control unit that will tolerate with any possible fault due to single missing or additional cell defect in molecular QCA. In our approach, we use a class of reversible logic gates called fault-tolerant reversible gates where the parity of the inputs is equal to the parity of the outputs. Since fault-tolerant reversible circuits have an ability to detect and correct errors at outputs, so detection and correction of errors would be simpler. If there was a single missing/additional cell defect, there would be a parity mismatch between the inputs and the outputs; otherwise, the input parity is the same as the output parity. A system would be fault-tolerant if it only includes fault-tolerant gates which can preserve parity. Also, this condition is required for the fault-tolerant reversible system.³ A reversible logic circuit does not erase (lose) information, so it prepares maximum output information and probability of fault detection.⁴ We choose Fredkin and Double Feynman gates to design a fault-tolerant reversible QCA circuit.

They are not only reversible but also parity preserve gates. In design of layout, we utilize layout of Fredkin gate which was proposed in Ref. 5.

This paper is organized as follows. Section 2 deals with preliminaries including a brief overview of QCA technology, reversible logic and fault-tolerant reversible circuits. Related work is explored in Sec. 3. The proposed design of fault-tolerant reversible control unit at logic level is presented in Sec. 4. In Sec. 5, the proposed logic design is implemented in QCA framework. QCADesigner version 2.0.3 is used to determine the functionality of QCA circuits. In Sec. 6, the comparison with previous work is presented in terms of evaluation criteria of reversible logic and QCA technology. The conclusion is given in Sec. 7.

2. Preliminaries

This section introduces the basic concepts of QCA and reversible logic and fault-tolerant reversible circuits that are relevant to the present work.

2.1. Quantum dot cellular automata

A fundamental element in QCA computation is QCA cell which is a square consisting of four quantum dots. Quantum dots are positioned at the corners of the cell which...
has two free electrons. The electrons can tunnel between the dots but not among cells, as shown in Fig. 1. They have to remain either in $P = +1$ or $P = -1$ polarization because of the coulombic repulsion. Coulombic repulsions are responsible for the transfer of information from one cell to an adjacent cell.

The basic QCA logic gates which can realize Boolean functions are majority voter (MV) and inverter (INV). The logic function of the MV is $MV(A, B, C) = AB + AC + BC$ which can be implemented by only five QCA cells, as shown in Fig. 2(a). The AND and OR gates are implemented by keeping one input of the MV in a stable state which are shown in Figs. 2(b) and 2(c), respectively. The INV is another basic gate in QCA that corresponds to NOT gate in conventional logic which is shown in Fig. 2(d). The inverter chain ($45^\circ$ wire), binary wire, and L-shaped wire are used as interconnect structures to transfer signals, which are shown in Figs. 2(e)–2(g), respectively. In general, fan-out is not allowed in reversible logic but based on a thermodynamic analysis which was presented in Ref. 8, it is possible to use it in QCA without violating reversibility. In Ref. 8, others have used a mechanical model to investigate the reversibility of QCA without a quantum-dynamical calculation. So, if zero power dissipation was considered, fan-out can be used without any limitation. Two structures of fan-outs are available, as shown in Figs. 2(h) and 2(i). Fan-out is just a connection of several QCA latches (wires). A fan-out gate has one input latch and two independent output latches. The coulombic interactions between cells cause a state transfer to outputs.

About $45^\circ$ wire is constructed using rotated cells that propagate the binary signal alternatively between the two polarizations, so the transfer of inverse state to the output is possible, as shown in Fig. 2(i). As shown in Fig. 3, in QCA-based logic implementations, there exist two types of wire-crossing; coplanar and multilayer wire-crossing. Wire-crossing achieves when two vertical and horizontal wires cross over each other.

Coplanar wire-crossing can be done in the same layer, with two QCA wires in different orientations ($90^\circ$ and $45^\circ$). The cellular nature of normal cell (or $x$-shaped) and rotated cell (or $+$-shaped) allow us to cross them in the same plane.
Fig. 2. Basic QCA devices. (a) MV, (b) INV, (c) QCA OR, (d) QCA AND, (e) INV chain, (f) Binary wire and (g) Fan-out.

Fig. 3. Wire-crossing: (a) Multilayer wire-crossing and (b) Coplanar wire-crossing.
Multilayer wire-crossing is formed by two normal wires which are connected through crossover bridge.

QCA computation uses a multi-phases clock in order to adjust timing entirely in the circuit. As shown in Fig. 4, QCA clock has four phases, i.e. switch, hold, release and relax whereas each phase is periodic. In this scheme, in every clock, a QCA array is divided into four sub-arrays called clock zones. Every phase is shifted in phase by 90° with respect to the previous one. This clocking scheme allows one of the clock zones to perform its computation in four phases and have its output as the input to the successor array. In other words, it refers to a pipeline computation where different sub-arrays can perform different parts of the computation.

2.2. Reversible logic

A computation is reversible if its outputs can always be reconstructed from its inputs. Reversible logic circuits have the same number of inputs as outputs and have the one-to-one mapping between vectors of inputs and vectors of outputs. Thus, inputs can always be recovered by outputs. The prominent feature that distinguishes reversible computing from conventional counterpart is the ability to prevent the loss of information which can lead to zero power dissipation. In 1961, Landauer proved that each bit of information loss generates kTln2 joules of heat energy, where k is Boltzmann’s...
constant and $T$ is the absolute (Kelvin) temperature.\textsuperscript{12} Also, in 1973, Bennett proved that if a system can retrieve its initial state from its final state, there would be no energy dissipation in it.\textsuperscript{13} The main component in reversible logic theory is the reversible function, which is defined as follows:

**Definition 2.1.** Suppose $f$ is a function which maps $I_f = \{I_1, I_2, I_3, \ldots, I_N\}$ to $O_f = \{O_1, O_2, O_3, \ldots, O_N\}$. $f$ is called reversible if each input pattern is related to exactly one output pattern and vice versa. Therefore, the number of inputs is equal to the number of outputs and $I_f$ can be constructed by $f^{-1}O_f$, where $f^{-1}$ is inverse function of $f$.

### 2.3. Fault-tolerant at logic-level

Fault-tolerance is a property that provides a system to operate properly even if some of its components have been defected. If a system is made up of fault-tolerant components, then the detection and correction of its faults become easier.

**Definition 2.2.** In computers, parity (from the Latin paritas, meaning equivalent) is a technique that checks whether information has been lost or preserved in the process of transformation. A gate will be parity preserving where the EXOR of the inputs matches with the EXOR of the outputs. In other words, for a parity preserving gate which has input $I = \{I_1, I_2, I_3, \ldots, I_N\}$ and outputs $O = \{O_1, O_2, O_3, \ldots, O_N\}$, it would be expected $I_1 \oplus I_2 \oplus I_3 \oplus \cdots \oplus I_N$ to be equal to $O_1 \oplus O_2 \oplus O_3 \oplus \cdots \oplus O_N$. Parity checking is one of the long-established methods for error detection mechanisms in digital logic and data communication systems.\textsuperscript{3}

If the system can preserve the parity of the input data throughout the computation, there is no need to intermediate checking.\textsuperscript{3} The necessary condition for parity preservation of a reversible circuit is just to include parity preserving gates. Due to the bijective nature of reversible logic, testing of the reversible circuit is also generally simpler than conventional irreversible circuits.\textsuperscript{14} Also, because of the information-lossless computation of reversible logic, the output information is maximized. Therefore, according to Ref. 4, the probability of fault detection can also be maximized. Based on Ref. 6, when there is a permanent fault due to manufacturing defects in molecular QCA, there would be parity mismatch between the input and the output of the parity preserving gates. Any permanent and transient fault in molecular QCA due to parity preserving property can be simultaneously detected. Fredkin and Double Feynman gates are two well-known reversible parity preserving gates, which are as follows:

Fredkin also called controlled-SWAP, is a universal reversible gate which is reversible and parity preserving. As shown in Fig. 5(a), Fredkin is $3 \times 3$ gate having the mapping $(A, B, C)$ to $(P = A, Q = A'B \oplus AC, R = AB \oplus A'C)$. While its truth table is shown in Table 1. A few number of parity preserving logic gates had been proposed in the literature. Among them, $3 \times 3$ Fredkin gate (FRG) depicted in
Fig. 5(a) and 3 × 3 Double Feynman gate (F2G) depicted in Fig. 7(a) are one-through gates as defined in Ref. 15. In other words, one input of these gates directly maps to the output, as depicted in Fig. 5(b) and Fig. 7(b). The QCA design of Fredkin gate was proposed in Ref. 16 and its cell layout is shown in Fig. 6(b). Also,

![Diagram of Fredkin gate](image)

Table 1. Truth table of Fredkin gate.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>P</th>
<th>R</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 5(a) and 3 × 3 Double Feynman gate (F2G) depicted in Fig. 7(a) are one-through gates as defined in Ref. 15. In other words, one input of these gates directly maps to the output, as depicted in Fig. 5(b) and Fig. 7(b). The QCA design of Fredkin gate was proposed in Ref. 16 and its cell layout is shown in Fig. 6(b). Also,

![Diagram of Fredkin gate with clock zones](image)

Fig. 6. Fredkin gate. (a) Logic diagram with clock zone ($D_0$–$D_3$ represent clock zones 0–3 respectively) and (b) Cell layout of Fredkin gate.
Fredkin gate logic diagram using four-phase clocking scheme is shown in Fig. 6(a). In this figure, the clocking zone is shown by the number next to $D$ ($D_0$ means clock zone 0, $D_1$ means clock zone 1, etc.). Thus, it can be observed that synthesis of Fredkin gate requires two-level MV and six MVs.

Double Feynman gate was proposed in Ref. 3. As shown in Fig. 7(a), Double Feynman is a $3 \times 3$ gate having the mapping $(A, B, C)$ to $(P = A, Q = A \oplus B, R = A \oplus C)$. Also, truth table of Double Feynman gate is shown in Table 2. The Feynman gate can be used as a signal copier to avoid the fan-out problem in the reversible logic. Its block diagram and quantum circuit implementation are shown in Fig. 7. In cell layout of Double Feynman depicted in Fig. 8(b), one clock containing four clock zones is used.

In Fig. 8(a), Double Feynman logic diagram is illustrated using four-phase clocking scheme. Thus, it can be observed that Double Feynman gate has two-level MV implementation with six MVs.

There are two types of wire-crossings: multilayer and coplanar wire-crossings. The cost of multilayer wire-crossing is three times of coplanar wire-crossing because at least three single layers are used in multilayer wire-crossing.

There are some figures of merit to evaluate the complexity of a QCA circuit design. To the best comparison and evaluation of the design, an appropriate figure of merit should be chosen. In most cases, the number of the primitive gates (MV, INV, and wire-crossings) are counted.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>P</th>
<th>R</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2. Truth table of Double Feynman.
Recently, some revolutionary algorithms, in particular, genetic algorithm, are used for reducing the number of required primitive QCA gates.\(^{17}\)

The aim of this paper is to utilize the fault-tolerance benefits of the logically reversible system and the one-to-one mapping in logic level design. Because of higher speed, Landauer clocking scheme is a better choice for our purpose. Therefore, the number of MVs could be a good metric relating to irreversible power dissipation. Because irreversible power dissipation mostly derives from MVs, so power dissipation is proportional to the number of MVs.\(^{18}\)

Delay is one of the cost metrics used for comparing the circuit speed.\(^{19}\) It is computed according to the number of clocks used for generating the outputs (every clock contains four clock zones).

An optimization approach has proposed to minimize the QCA gates, which results in delay and complexity minimization simultaneously.\(^{20}\) Area of QCA circuit is proportional to the number of cells which are used in implementing the circuit. An optimization approach based on genetic algorithms has proposed to reduce the number of cells.\(^{21}\)

On the other hand, important figures of merit in the design of a reversible logic are the number of gates, the number of constant inputs, the number of garbage outputs and quantum cost which are defined as follows:\(^{22}\):

Number of the gates: It is a figure that shows the number of the gates which are used in the circuit without considering types and sizes of the gates. So it might not be a good parameter to distinguish complexity if the circuits include gates with different types and sizes.

Number of constant inputs: There are some parts of inputs with constant value that are used for translating an irreversible circuit into a reversible one.
Garbage outputs: These outputs can be added to satisfy the condition of equality between the number of inputs and outputs of a gate.

Quantum cost: Quantum cost is a technology-independent metric which explains the effort required to transform a reversible circuit to a quantum circuit. It is recognized as the most important figure of merit that calculates the number of primitive reversible logic gates which are used for the implementation of the circuit. These are \( \text{NOT}, \text{CNOT}, \text{Controlled-V}, \text{and Controlled-}V^\dagger \), where

\[
V = \left( \frac{1 + i}{2} \right) \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}.
\]

3. Related Work

One of the earliest studies in the test and defect characterization of QCA was proposed by Tahoori et al., in an influential work in Ref. 2. The authors showed that a network of MVs and INVs have specific testing properties. They partitioned the design into INVs and MVs which resulted in a reduced test set and with 100% fault coverage. In Ref. 23, QCA defects are described at the molecular level for combinational circuits and fault characterization is analyzed for single missing/additional cell defect on different QCA devices such as MV, INV, fan-out, wire-crossing and L-shaped wire.

In Ref. 24, test generation framework for QCA is presented. In Ref. 25, fault-tolerant QCA designs are presented using triple modular redundancy with shifted operands. In Ref. 6, reversible logic is used to detect single missing/additional cell defect. This method of error detection is offline which applies on two new gates (QCA1 and QCA2). In every step, it is assumed that one cell of the gate is missed. Faulty outputs are compared to the normal outputs and test set vectors are generated. There is a minimal test set that detects a single missing/additional cell defect. Defect analysis of reversible QCA gates and corresponding testability can be extended to a 1D array whose basic module is one of the reversible gates. They proved that if the 1D array is made by \( N \) number of QCA reversible gates module, then the number of vectors for testing the 1D array is independent of \( N \) but dependent on reversible modules. Although, it seems an efficient approach but it has many limitations in design. The reason is that designing an array of a reversible logic gate is inadequate for the synthesis of efficient reversible circuits.

Among previous work, many of them utilized reversible logic and its parity preserving property. For example, in Ref. 26, Fredkin gate was used as a parity preserving gate to design testable latch and in Ref. 27, in order to design a testable FPGA. Due to its parity preserving property, if the circuit works properly, then parity of the inputs and the outputs are the same.

In Ref. 2, two new reversible gates (QCA1 and QCA2) were proposed. These gates were compared (in terms of delay, area and logic synthesis) with other reversible gates (such as Toffoli and Fredkin) for QCA implementation. Defect analysis of the reversible gates was pursued under a single missing/additional cell assumption.
A new reversible QCA gate called RQCA was proposed in Ref. 28. The functionality of it was synthesized by three parity preserving reversible gates (one Fredkin and two Double-Feynman gates), so a fault-tolerant gate (FT-RQCA) was made.

A fault-tolerant ALU was designed using FT-RQCA gate. A reversible ALU was proposed in QCA framework.29 The design is based on new fault-tolerant reversible adder (FTRA) which is a parity preserving gate implemented in QCA. The ALU was synthesized using one FTRA, two Fredkin and four Double Feynman gates.

Recent developments in QCA manufacturing focus on molecular implementation, in which each QCA cell is a molecule.1,30 Also, previous work for fault-tolerant QCA circuits have focused on single missing or additional cell defects. As shown in Fig. 9,31 in manufacturing processes of QCA circuit, a defect may occur in both chemical synthesis phase and deposition phases. However, cells are manufactured in chemical phase and they are located in a particular position on the surface in deposition phase.

Missing or extra dots and/or electrons can happen in chemical synthesis phase and may cause defects.6,32,33 However, according to Ref. 26, defects are more probably to happen in the deposition phase than in the chemical synthesis phase, which results in perfectly produced cells, imperfectly located in the substrate. This means that they make a cell misplacement defect. As shown in Fig. 10 deposition phase defects are mainly classified into four parts31:

- **Cell Displacement and Misalignment**: The direction of the defective cell is misplaced.
- **Cell Omission/Missing**: A particular cell without electron or missing electron.
- **Additional Cell Deposition**: An additional cell is deposited on the substrate.
- **Rotation Cell Defect**: A cell is displaced/rotated from its main place by an angle of $\Theta$.

In Ref. 34, to design the reversible control unit, two Fredkin gates, two $3 \times 3$ Toffoli gates, three NOT gates, and two Feynman gates have been used. Their design realizes eight arithmetic and four logical operations. In terms of criteria, the design has three constant inputs, six garbage outputs and quantum cost equal to 23. In Ref. 35, three Peres gates, three Feynman gates, and one Fredkin gate were used. It has eight garbage outputs, four constant inputs and quantum cost equal to 20. In Ref. 36, a reversible control unit was designed using three Feynman gates, one Fredkin and three R-I gates. The number of constant inputs is four and the number of

![Fig. 9. MV gate defects. (a) Cell Omission, (b) Cell displacement, (c) Cell misalignment, (d) Extra Cell and (e) Rotation Cell.](image-url)
garbage outputs is eight. In Ref. 37, reversible logic gates with parity preserving property were used to design a fault-tolerant control unit. As we mentioned before, if a circuit only contains parity preserving gates, then the whole of the circuit maintains parity. In this design, two Double Feynman gates, three NFT gates, and one PPPG gate were used. It has nine constant inputs, 12 garbage outputs and quantum cost equal to 31.

4. The Proposed Design

The reversible computing is almost focused in design and synthesis of computational units by reversible gates and less attention has been paid to take advantages of fault-tolerant characteristics of specific reversible gates and to design fault-tolerant reversible circuits. On the other hand, QCA is one of the most promising emerging nanotechnologies in which efficient implementation of reversible logic gates is possible.38,39

Realizing reversible logic gates in nanotechnology based on molecular QCA or other nanotechnology devices are susceptible to high errors rates. All these facts motivate us to design a reversible circuit in QCA framework with test capabilities.

The design of a control unit for any computing system is the most important part and involves many critical limitations. Here, we propose a fault-tolerant reversible control unit for a typical processing unit using parity preserving reversible gates. Quantum cost of the fault-tolerant control unit, which is synthesized with six parity preserving gates including four Fredkin gates and two Double Feynman gates, is 24.

4.1. Proposed fault-tolerant control unit in QCA framework

Our design for a processing unit contains two parts; one is a control circuit, and another is a full adder. The operations are executed on one-bit inputs. To have a
simple design, in the proposed design, there are not a complete ALU for all arithmetic and logic function, but it is the responsibility of control unit to change the inputs and then the only function in the ALU part is just a full adder. The proposed control unit utilizes the Fredkin and Double Feynman gates to produce 12 arithmetic-logic operations.

By cascading the proposed control unit including reversible logic gates with a full adder, a processing unit can be constructed, as shown in Fig. 11. The full adder circuit is used to generate final results, so with this one-bit processing unit, we can realize 12 one-bit operations. Also, $n$-bit processing unit can be constructed by cascading $n$ number of the proposed one-bit circuit. The general operations executed by the proposed are as follows:

$$ F_i = X_i \oplus Y_i \oplus Z_i, \quad (1) $$

$$ C_i = X_iY_i + Y_iZ_i + X_iZ_i, \quad (2) $$

where $X_i, Y_i, Z_i$ are defined by following equations:

$$ X_i = A_i + S_2S_0'(S_1 \oplus B_i), \quad (3) $$

$$ Y_i = S_0B_i + S_1B_i', \quad (4) $$

$$ Z_i = S_2'C_i, \quad (5) $$

The proposed processing unit has six inputs, which are divided into data inputs, $A_i, B_i$ and $C_i$, and control inputs, $S_0, S_1$ and $S_2$, and it has two outputs, $F_i, C_{i+1}$, as shown in Fig. 11. It performs 12 different operations (either arithmetic or logical) based on the values of control bits, $S_0, S_1, S_2$ along with the carry input, $C_i$.

$F_i$ is the main output of the processing unit, and $C_{i+1}$ will be equal to one if the arithmetic operation produces a carry otherwise is zero.

The processing unit implements the logic functions AND, OR, EXOR, and NOT, as well as increment and decrement arithmetic operations. $S_2$ is the mode input which selects between arithmetic and logical operations. When $S_2 = 0$, the circuit performs
eight arithmetic operations, as follows:

\[ X_i = A_i, \]
\[ Y_i = S_0 B_i + S_1 B'_i, \]
\[ Z_i = C_i. \]

Referring to the aforementioned equations and tables, the arithmetic operations are performed based on addition function. It uses two’s complement addition to perform subtraction operation. In other words, \( A_i + B'_i + 1 \) operation is implemented instead of \( A_i - B_i \). Two’s complement addition is same as subtraction but easier to implement. It is also equivalent to ones’ complement plus one.

As we mentioned before, performing an operation depends on control lines. Eight arithmetic operations will be provided by different values of \( C_i, S_0, S_1 \). Depending on the values of \( C_i, S_0 \) and \( S_1 \), one of the arithmetic functions will operate. As shown in Tables 3 and 4, if \( C_i = 0 \), then one of these functions; \( A_i + B_i, A_i + B'_i, A_i, A_i - 1 \), will be performed. If \( C_i = 0 \), then \( A_i + B_i + 1, A_i + B'_i + 1, A_i + 1, A_i \) will be operated. When \( S_2 = 1 \), we get four logical operations, i.e. NOT, AND, OR and EXOR as follows:

\[ X_i = A_i + S_0 (S_1 \oplus B_i), \]
\[ Y_i = S_0 B_i + S_1 B'_i, \]
\[ Z_i = 0, \]

so logical operations are generated. For \( S_2 S_1 S_0 = 100, F_i \) is OR operation and when \( S_2 S_1 S_0 = 101, \) then \( F_i \) is exclusive-OR operation. To perform AND and NOT operations, \( S_2 S_1 S_0 \) should be equal to 110 and 111, respectively.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>( A_i + B_i )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>( A_i )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>( A_i + B'_i )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>( A_i )</td>
</tr>
</tbody>
</table>
Different arithmetic (including transfer) and logical operations, that can be performed using the circuit in Fig. 12, are summarized in Table 5.

The proposed control unit not only is able to realize the arithmetic–logical operations but also has ability to detect and correct the faults that may occur during the operations. Most of the current researchers emphasize only on the reversible implementation of control unit without any consideration of testability and controllability.

Generally, parity preserving reversible logic circuits are more complicated compared to reversible logic circuits. The reason is that the designers have a limitation in choosing gates with less quantum cost. However, our proposed design is better in terms of number of constant inputs, number of garbage outputs and number of gates.

**Table 5. Complete functional table of the designed control unit.**

<table>
<thead>
<tr>
<th>(S_2)</th>
<th>(S_1)</th>
<th>(S_0)</th>
<th>(C_i)</th>
<th>Output equals</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(F = A)</td>
<td>Transfer A</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>(F = A + 1)</td>
<td>Increment A</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>(F = A + B)</td>
<td>Addition</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>(F = A + B + 1)</td>
<td>Add with carry</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>(F = A + B')</td>
<td>Subtract with borrow</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>(F = A + B' + 1)</td>
<td>Subtraction</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>(F = A - 1)</td>
<td>Decrement</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>(F = A)</td>
<td>Transfer A</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(F = A \lor B)</td>
<td>OR</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>(F = A \oplus B)</td>
<td>XOR</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>(F = A \land B)</td>
<td>AND</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>(F = A')</td>
<td>Complement</td>
</tr>
</tbody>
</table>
5. Results

The design is verified using QCADesigner tools version 2.0.3.\textsuperscript{40} Bistable approximation engine was chosen to simulate the designed circuit. The obtained outputs from bistable simulation engine, confirm the correctness of the proposed designs, as shown in Fig. 14.

Also, it is possible to calculate complexity, delay and area of QCA circuits using QCADesigner tools.\textsuperscript{41} There are some parameters in a bistable engine that the designer is able to either set them or use default values. In this paper, we change some default values which are: the number of samples, radius of effect and maximum iteration per sample, as shown in Table 6. Based on the number of samples, the simulation engine scans each cell and then estimates its polarization. If number of samples is chosen too small, it may be insufficient to get the correct results. So, the value of the number of samples must be chosen sufficiently large, especially in a layout with large number of cells.\textsuperscript{18}

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell size</td>
<td>18 nm</td>
</tr>
<tr>
<td>Number of samples</td>
<td>420000</td>
</tr>
<tr>
<td>Convergence tolerance</td>
<td>0.001000</td>
</tr>
<tr>
<td>Radius of effect</td>
<td>41 nm</td>
</tr>
<tr>
<td>Relative permittivity</td>
<td>12.900000</td>
</tr>
<tr>
<td>Clock high</td>
<td>$9.800000 \times 10^{22}$</td>
</tr>
<tr>
<td>Clock low</td>
<td>$3.800000 \times 10^{23}$</td>
</tr>
<tr>
<td>Clock amplitude factor</td>
<td>2.000000</td>
</tr>
<tr>
<td>Layer separation</td>
<td>11.500000</td>
</tr>
<tr>
<td>Maximum iterations per sample</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 7. QCA parameters of proposed design.

<table>
<thead>
<tr>
<th>Figure of merit</th>
<th>MV</th>
<th>INV</th>
<th>Clock zone</th>
<th>Coplanar wire-crossing</th>
<th>Number of cells</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed design</td>
<td>35</td>
<td>16</td>
<td>12</td>
<td>19</td>
<td>1314</td>
<td>3.27 $\mu^2$</td>
</tr>
</tbody>
</table>

Table 8. Comparison of existing reversible control unit designs and proposed design.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Number of gates</th>
<th>Number of garbage outputs</th>
<th>Number of constant inputs</th>
<th>Quantum cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>23</td>
</tr>
<tr>
<td>35</td>
<td>7</td>
<td>8</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>36</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>Proposed design</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>24</td>
</tr>
</tbody>
</table>

G. Bahadori, M. Houshmand & M. Zomorodi-Moghadam

Int. J. Quantum Inform. Downloaded from www.worldscientific.com
Table 9. Comparison of existing design of fault-tolerant reversible control unit and proposed design.

<table>
<thead>
<tr>
<th></th>
<th>Number of gates</th>
<th>Number of garbage outputs</th>
<th>Number of constant inputs</th>
<th>Quantum cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing fault-tolerant</td>
<td>7</td>
<td>12</td>
<td>9</td>
<td>31</td>
</tr>
<tr>
<td>Proposed design</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>Improvement</td>
<td>14.28%</td>
<td>66.66%</td>
<td>77.77%</td>
<td>22.58%</td>
</tr>
</tbody>
</table>

Fig. 13. Cell layout of proposed fault-tolerant control unit.
Fig. 14. Simulation result of fault-tolerant control unit achieved by QCA Designer.
As we mentioned before, there are effective interactions between neighbor cells. Radius of effect parameter determines what neighbors are included to calculate polarization of a cell. It should be chosen based on the next-to-nearest neighbors.

As there was no existing implementation of fault tolerant reversible control unit in QCA, we just compared our design with existing logic-level designs of reversible control units. But, we tried to optimize the important factors of QCA design as much as possible and the measured values of these factors are presented in Table 7.

Table 8 demonstrates the comparison of the proposed design with three existing reversible control units in the field and it can be seen that our proposed design is better than the existing counterparts in terms of the number of gates, the number of garbage outputs and the number of constant inputs.

Also Table 9 compares our proposed design with the existing fault-tolerant control unit design in Ref. 36. It can be seen that our proposed design is better in all measured factors and there is a remarkable improvement in all four factors.

The information about implementation of the proposed design in QCA are illustrated in Table 7. Tables 8 and 9 include information about figures of merit of the proposed circuit and previous works.

Using gates with larger size and higher quantum cost like NFT and PPPG gates have caused significant differences between our proposed design and the design proposed by Ref. 37.

The QCA cell layout of the proposed fault-tolerant control unit is shown in Fig. 13. The design requires 35 MVs, 16 INVs and has an input to output delay of 12 clock zones. Simulation results of the proposed design are performed by QCADesigner 2.0.3 and it is shown in Fig. 14.

6. Conclusion

In this paper, a novel architecture of reversible fault-tolerant QCA of the control unit is presented. Application of reversible computing in the detection of the faults is demonstrated and QCA fault models are represented. The proposed fault-tolerant control unit is modular and can be used as part of an n-bit processing unit. In terms of factors of reversible computation, our design has significant improvements compared to the previous fault-tolerant reversible control unit; which is 14.28% in the number of gates, 66.66% in the number of garbage outputs, 77.77% in the number of constant inputs and 22.58% in quantum cost. Since making a circuit fault-tolerant results in more complexity, it is expected that the fault-tolerant circuit would be more complicated compared to its reversible counterpart. Also as this design is optimized the results are even better than nonfault tolerant counterparts in terms of the number of gates, the number of garbage outputs and the number of constant inputs.

The final contribution of this work is that to the best of our knowledge, there is no such fault-tolerant control unit implementation in QCA framework. It is an attempt to design a fault-tolerant control unit in QCA framework which is also optimized in terms of delay, complexity and area. Synthesis and implementation of presented work
are done by well-known reversible gates to utilize advantages of reversible computation. The design of QCA layouts and the verification of the design are performed using the QCADesigner. For the future work, we can design a new parity preserving reversible logic gate with MV function to reduce the complexity of fault-tolerant reversible circuit in QCA.

References


