Nanoscience and Nanotechnology. 2011; 1(1): 1-7
DOI: 10. 5923/j.nn.20110101.01

Serpent Implementation in Quantum Cellular Automata

M. A. Amiri[1,*], S. Mirzakuchaki[1], M. Mahdavi[2], N. K. Darav[3]

[1]E. E. Department, Iran University of Science and Technology, Tehran, Iran
[2]Shahr-e-Qods Branch, Islamic Azad University, Shahr-e-Qods, Iran
[3]Lahijan Branch, Islamic Azad University, Lahijan, Iran

Abstract Quantum Cellular Automata (QCA) is an emerging technology at the nanotechnology level. Lower power consumption, higher density and higher speed nature of QCA technology are very interesting. Nowadays, many applications of QCA technology are introduced and cryptography can be an attractive application of QCA technology. The implementation of the Serpent block cipher in Quantum Cellular Automata is the main purpose of this paper. The main modules of this cryptographic algorithm are implemented in this technology and the implementation results are discussed. The two methods of S-Box design, i.e. LUT-Based and Logic-Based methods are inspected. The Serpent's S-Boxes are designed and simulated by these two methods. The simulation results are obtained from QCADesigner software.

Keywords Quantum Cellular Automata, Serpent, Cryptography, Implementation

1. Introduction

During past several decades, the microelectronics industry has improved the integration, the power consumption, and the speed of integrated circuits by means of reducing the feature size of transistors. But it seems that even by decreasing the transistor sizes, some problems such as power consumption cannot be ignored. QCA which was first introduced by Lent et al.[1] represents an emerging technology at the nanotechnology level.

Utilizing the QCA technology for implementing logic circuits is one of the approaches which in addition to increasing the clock frequency of these circuits to tera-hertz frequencies and decreasing the size of logic circuits to nanoscale, decreases the power consumption of these circuits[1, 2]. QCA cells have quantum dots in which the position of electrons will determine the binary levels of 0 and 1. This is the most distinct feature of QCA against conventional logic in which logical states are stored by means of voltage levels.

Serpent block cipher was a finalist candidate of Advanced Encryption Standard (AES). This cryptographic algorithm has 32 rounds with an 128-bit block size and a 256-bit key size. This cryptographic algorithm consists of an initial permutation IP, 32 rounds, and a final permutation FP. Each round involves a key mixing operation, a pass through S-boxes, and a linear transformation. In the last round, the linear transformation is replaced by an additional key mixing operation[3].

* Corresponding author:
amiri@ee.iust.ac.ir (M. A. Amiri)
Published online at http://journal.sapub.org/nn

As an application of QCA technology, we have implemented the Serpent block cipher. Simulation results of this implementation are obtained from QCADesigner v2.0.3 software. QCADesigner is developed by the ATIPS lab at the University of Calgary in Canada. QCADesigner v2.0.3 has different simulation engines. Throughout this paper, the coherence vector simulation engine is used due to its accurate and detailed evaluation of QCA.

The remainder of this paper is organized as follows. In Section 2, a brief review of QCA is presented. In Section 3, Serpent block cipher and its important modules are discussed. The implementation of the Serpent block cipher by means of the implementation of its modules is presented in Section 4. Section 5 concludes this chapter.

2. Quantum Cellular Automata

A cell of Quantum Cellular Automata is constructed by means of four quantum dots as schematically shown in Figure 1 where open circles implicate the quantum dots. Two of four quantum dots in each cell are filled by two electrons which are schematically shown as solid dots. The electrons are permitted to jump between the various quantum dots in a cell by means of the mechanism called quantum mechanical tunneling but they are not allowed to tunnel between two individual cells. The barrier height between two adjacent cells is assumed so high that it completely blocks intercellular tunneling.

If the electrons are left alone, they will meet the arrangement corresponding to the physical ground state of the cell. It is clear that the coulombic force between two electrons in each cell compels them to occupy two different dots. By these concepts, the ground states of a cell will be two basic arrangements with electrons at opposite corners, as

shown in Figure 1. The positions of the electrons are also illustrated in this figure.

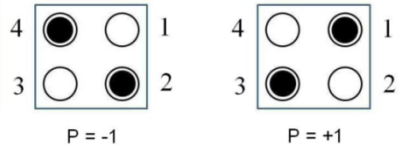

Figure 1. QCA cell and its ground states.

Coulombic interaction between two adjacent cells provides their coupling. Figure 2 illustrates how one cell is affected by the state of its neighbor cell[4]. In this figure, the polarization of cell 1 follows the polarization of cell 2. The polarization of cell 2 (P2) is assumed to be fixed at given value and this polarization affects on cell 1 and determines the polarization of this cell.

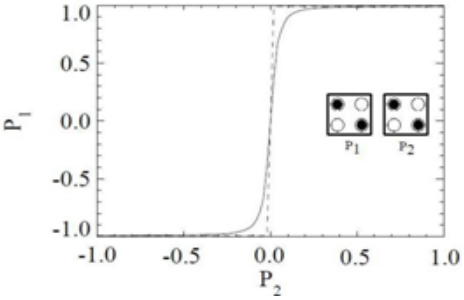

Figure 2. Coupling of QCA cells.

The non-linear nature of the cell-cell coupling is a result that can be drawn here. Cell 1 is almost completely polarized in presence of cell 2 which might be partially and not completely polarized[2,4].

Elementary Boolean logic functions can be realized by exploiting the physical interaction between cells. In Quantum Cellular Automata, the Majority gate and the Inverter gate as shown in Figure. 4(a) and Figure 3 respectively are the basic logic gates and other logic functions are implemented by means of these two gates. As shown in Figure 4, a Majority gate is composed of only 5 QCA cells[5]. An AND function can be implemented by a Majority gate that one of its inputs is stuck at logical zero whereas if we set one input of a Majority gate to logical one, we obtain a logic OR function.

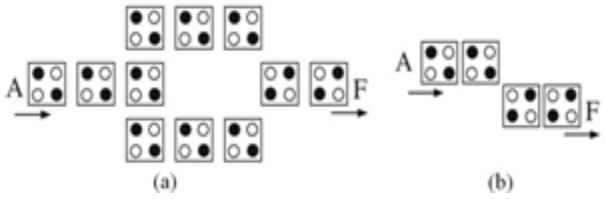

Figure 3. (a) Redundant inverter gate and (b) Inverter gate.

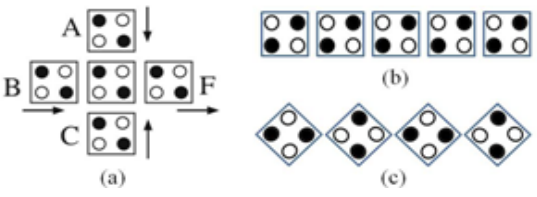

Figure 4. (a) Majority logic gate, (b) Binary wire, and (c) Inverter chain.

Synchronization of information flow through a QCA circuit and the direction of information flow in a QCA cell are accomplished by QCA clocking mechanism as well as the required power for the operation of a QCA circuit in other words. The QCA clocking mechanism is used to control the tunneling barrier height in QCA cells. The electrons are trapped in their positions, when the clock is low and they cannot tunnel into other dots, therefore latching the cell (Hold phase). This condition is caused by the intracellular barriers that are held at their maximum height. The cell goes to the null polarization state (Relax phase) when the clock signal is high because the intracellular barriers are held at their minimum height. Between these two conditions, the cells are either switching or releasing.

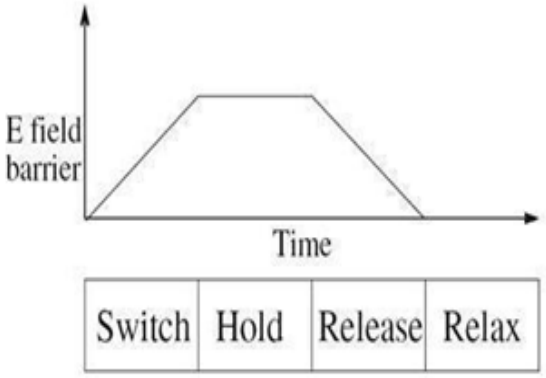

Figure 5. Barrier hight in four phases of clock.

Figure 5 illustrates the barrier height in four phases of the QCA clock. Each cell in an individual clocking zone is connected to one of the four available phases of the QCA clock demonstrated in Figure 6. Each QCA cell is synchronously latched and unlatched by changing of the clock signal and thus the information is distributed throughout the circuits[6,8,9,10].

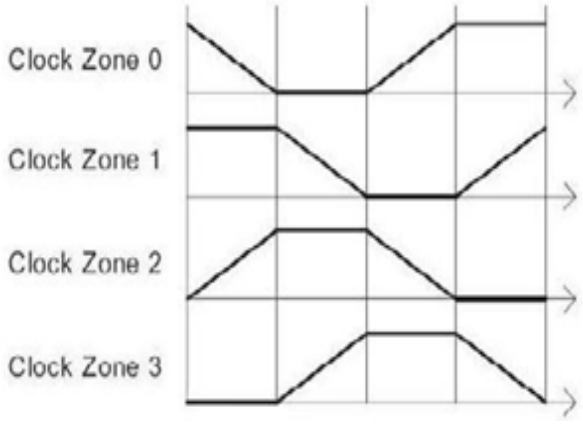

Figure 6. QCA clock zones.

3. Serpent Block Cipher

Serpent is a 32-round substitution permutation network (SPN) operating on four 32-bit words, thus having a block size of 128 bits[3,11]. Serpent encrypts a 128-bit plaintext to a 128-bit ciphertext in 32 rounds with 33 subkeys. The user

key length is assumed to be variable but in the proposal, it is fixed to be 128, 192 or 256 bits. It should be mentioned that the short keys with less than 256 bits are mapped to 256 bits keys by appending one '1' bit to the MSB end followed by as many '0' bits as required to produce 256 bits. The cipher consists of an initial permutation IP, 32 rounds, and a final permutation FP. Each round involves a key mixing operation, a pass through S-boxes, and a linear transformation. In the last round, the linear transformation is replaced by an additional key mixing operation[3].

3.1. Key Mixing

At each round, a 128-bit subkey Ki is XORed with the current intermediate data Bi.

3.2. The S-Boxes

Serpent has 8 individual 4×4 S-Boxes which repeat every 8 round. Every 128-bit input of the S-Box will be divided to 32 blocks of 4-bits and every block will be applied to a 4×4 S-Box. The outputs of these 32 S-Boxes will be concatenated again together to perform a 128-bit block.

3.3. Linear Transform

The 128-bit output of S-Box will be divided to four 32-bits and these words are linearly mixed by some shift, rotate and XOR operations. A complete round of Serpent is described as:

$X0, X1, X2, X3 := Si(Bi \oplus Ki)$
$X0 := X0 <<< 13$
$X2 := X2 <<< 3$
$X1 := X1 \oplus X0 \oplus X2$
$X3 := X3 \oplus X2 \oplus (X0 << 3)$
$X1 := X1 <<< 1$
$X3 := X3 <<< 7$
$X0 := X0 \oplus X1 \oplus X3$
$X2 := X2 \oplus X3 \oplus (X1 << 7)$
$X0 := X0 <<< 5$
$X2 := X2 <<< 22$
$Bi+1 := X0, X1, X2, X3$

Where <<< means Rotation and << means Shift. In the last round, this linear transform is replaced by an additional key mixing.

$B32 := S7(B31 \oplus K31) \oplus K32$

4. QCA Implementation

In this work, each QCA cell is assumed to have the width and length of 18 nm like previous works such as[12,13]. The neighbor cells have a center to center distance of 20 nm. Implementation of initial permutation and final permutation which are just bit reordering functions[3] is accomplished by routing of inputs to desired outputs. Key mixing or XOR function is implemented by 3 majority gates. Figure 7 illustrates the implementation of the key mixing function. This module has two inputs and one output. One of the inputs of this module comes from the round input and the other is the corresponding bit of the round key. This implementation has a latency of two clock periods, a complexity of 68 cells and an area of about 79200 nm2.

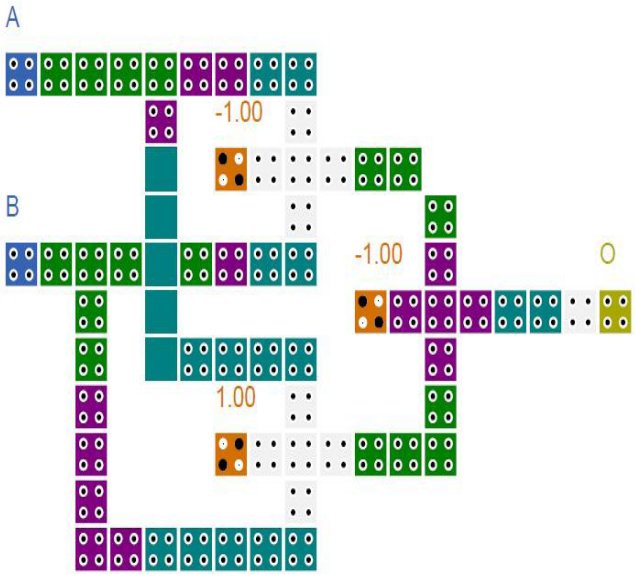

Figure 7. Key mixing function or XOR.

The linear transform function has been first simplified and then it has been implemented. The simplified function was only constructed by XOR gates. As an example, the simplified 33rd and 34th bits of linear transform stage are as follows:

$$LT(32) = LTI(14) \oplus LTI(33) \oplus LTI \quad (68)$$

$$LT(33) = LTI(15) \oplus LTI(34) \oplus LTI \quad (69)$$

Figure 8 illustrates the implementation of the 33rd output of linear transform. As illustrated in this figure, 2 XORs are applied to 3 inputs to perform the output.

This implementation has a latency of three clock periods, a complexity of 143 cells and an area of about 179200 nm2.

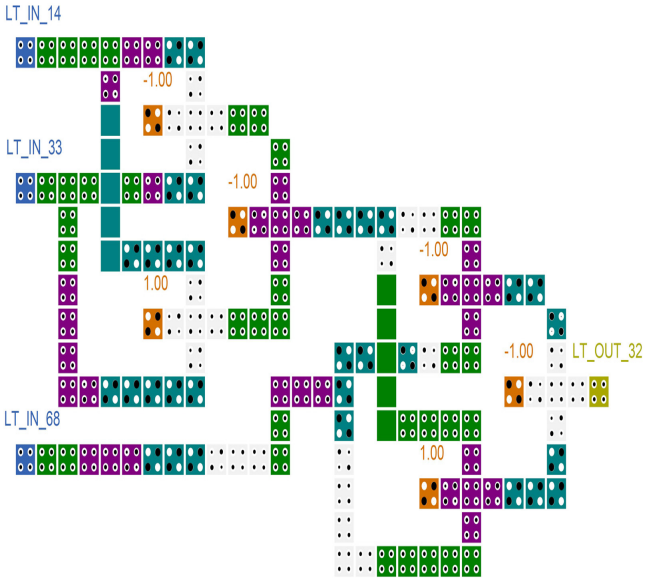

Figure 8. Linear transform for 33rd output of linear transform.

4.1. Design and Implementation of S-Boxes

Serpent block cipher has 8 individual S-Boxes named S0 to S7. Here, the S0 substitution function is selected among Serpent's S-Boxes to be implemented. Other S-Boxes can be designed and implemented in such a manner. The S0 substitution function is illustrated in Table 1. Input and output values of this S-Box are shown in binary format and they have the length of 4 bits each.

4.1.1. LUT-Based Design

In this method, a Look-Up-Table or memory is used to implement the S-Box. All the output bits of the S-Box i.e. O3, O2, O1, and O0 are implemented separately. The O0 design and implementation is discussed here. All the other output bits are implemented in such a manner. Just the memory contents are the differences between the implementation of these output bits.

It means that for implementation of each output bit, the corresponding values should be stored in the memory. A 16 to 1 multiplexer is used to design a LUT-Based S-Box. The 16 to 1 multiplexer is composed of fifteen 2 to 1 multiplexers. A 2 to 1 multiplexer is illustrated in Figure 9.

This novel multiplexer has a complexity of 31 QCA cells. When the "S" input of this multiplexer has the value of '0', the value of the "A" input will appear on the "O" output and when the "S" input has the value of '1', the value of the "B" input will appear on the "O" output.

The input bits of the S-Box are connected to the "S" input of the 16 to 1 multiplexer. The output values of the S-Box are fixed on the Data inputs of the 16 to 1 multiplexer. Using this structure, when the input value is applied to the "S" input of the multiplexer, the corresponding values will appear on the output port of the multiplexer after 10 clock periods of latency.

This latency is the result of clocking which is used for data propagation throughout the circuit. It should be mentioned that after 10 clock periods of latency, the output value of the multiplexer will be valid in each clock period.

4.1.2. Logic-Based Design

In this method, the logic function of each output is used to implement it. All the output bits of the S0 S-Box i.e. O3, O2, O1, and O0 are implemented separately. Design and Implementation of O0 output bit is discussed here.

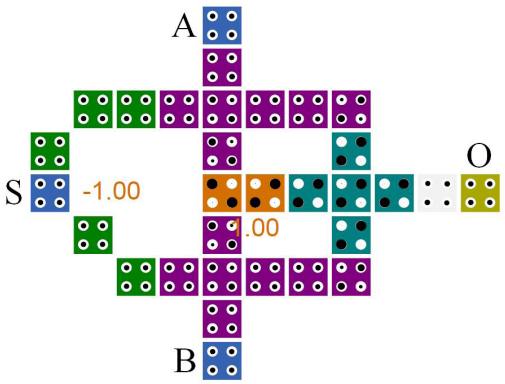

Figure 9. QCA layout of 2 to 1 multiplexer.

Table 1. The S0 S-Box of Serpent Block Cipher.

S-Box Input				S-Box Output			
S3	S2	S1	S0	O3	O2	O1	O0
0	0	0	0	0	0	1	1
0	0	0	1	1	0	0	0
0	0	1	0	1	1	1	1
0	0	1	1	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1
0	1	1	1	1	0	1	1
1	0	0	0	1	1	1	0
1	0	0	1	1	1	0	1
1	0	1	0	0	1	0	0
1	0	1	1	0	0	1	0
1	1	0	0	0	1	1	1
1	1	0	1	0	0	0	0
1	1	1	0	1	0	0	1
1	1	1	1	1	1	0	0

LUT-Based QCA implementation of the O0 bit of the S0 S-Box is illustrated in Figure 10. Each output bit of the S0 S-Box is exhaustively simulated. Simulation results of the O0 bit is illustrated in Figure 12. The results of LUT-Based QCA implementation of the S0 S-Box is discussed in Table 2. It can be seen that the Delay, Complexity and Area of all the output implementations are the same.

All the other output bits are implemented in such a manner. The following logic functions are extracted from Table 1. Considering any output bit, the implementation of each term of logic function is composed of one, two, or three majority gates which are used as logic AND functions.

$$O3 = \overline{B}\overline{C}D + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + BCD + ABC + A\overline{B}\overline{C}$$

$$O2 = \overline{A}C\overline{D} + \overline{A}B\overline{C}D + ABCD + A\overline{C}\overline{D} + A\overline{B}\overline{C} + A\overline{B}\overline{D}$$

$$O1 = \overline{C}\overline{D} + \overline{A}BD + \overline{A}\overline{B}\overline{D} + A\overline{B}CD$$

$$O0 = \overline{A}\overline{B}\overline{D} + AB\overline{D} + \overline{A}C + A\overline{B}\overline{C}D$$

One majority gate is used for the terms which contain only two inputs, two majority gates are used for the terms which contain three inputs, and three majority gates are used for the terms which contain four inputs. The output of AND functions are also logically ORed to result the desired output.

Logic-Based QCA implementation of the O0 bit of the S0 S-Box is illustrated in Figure 11. Each output bit of the S0 S-Box is exhaustively simulated.

Simulation results of the output bits are illustrated in Figure 13. The results of Logic-Based QCA implementation of the S0 S-Box is discussed in Table 3.

The O0 and O1 output values are valid after 6 clock periods of latency and the O2 and O3 output values are valid after 8 clock periods of latency. The maximum Delay among four output bits is considered to be the Delay of Logic-Based S-Box.

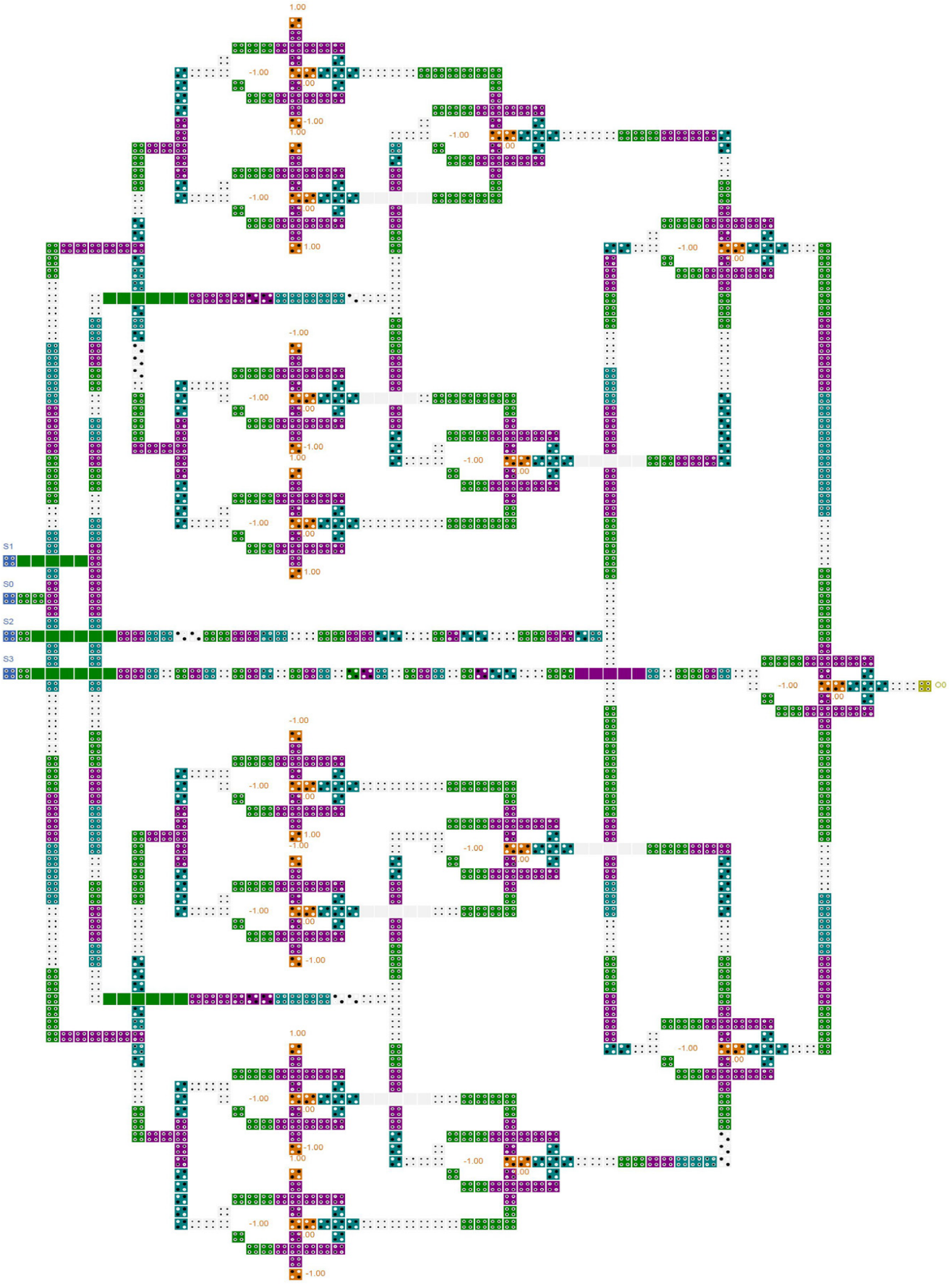

Figure 10. LUT-Based QCA Implementation of O0 bit of S0 S-Box.

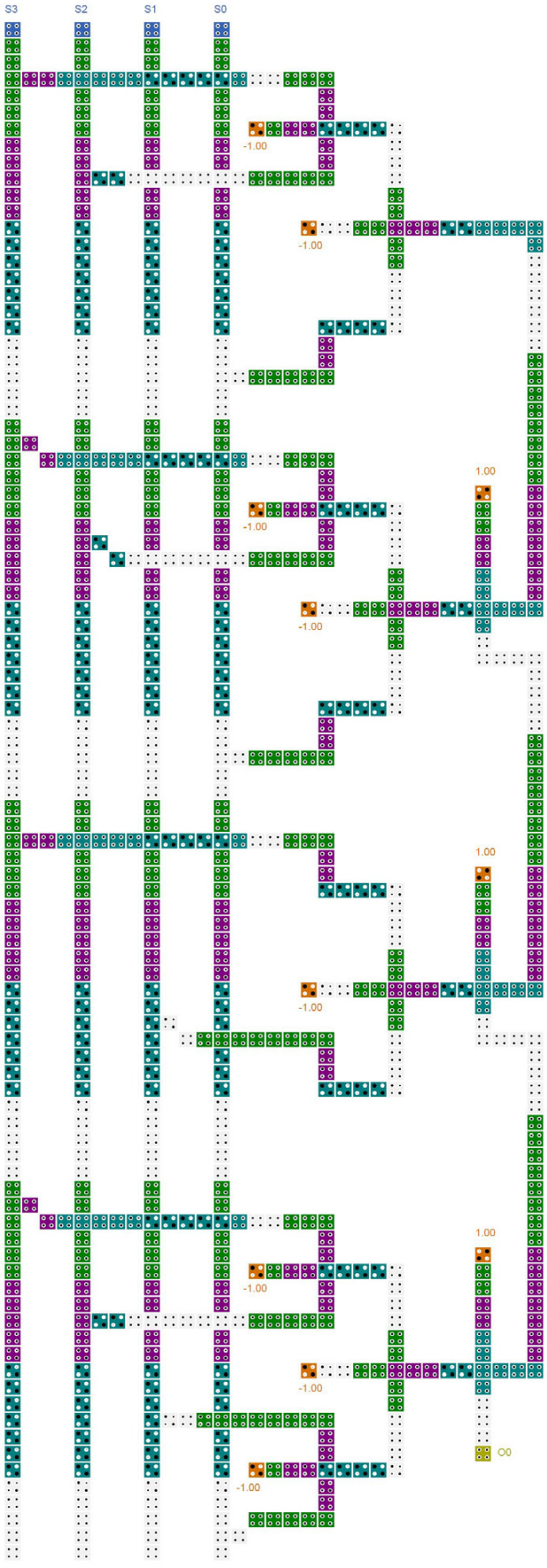

Figure 11. Logic-Based QCA Implementation of O0 bit of S0 S-Box.

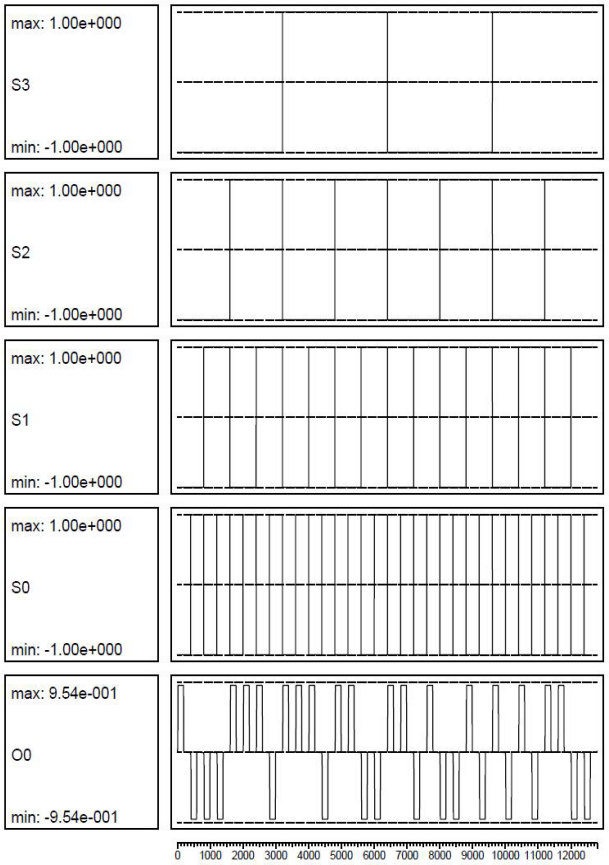

Figure 12. Simulation Result of LUT-Based S-Box.

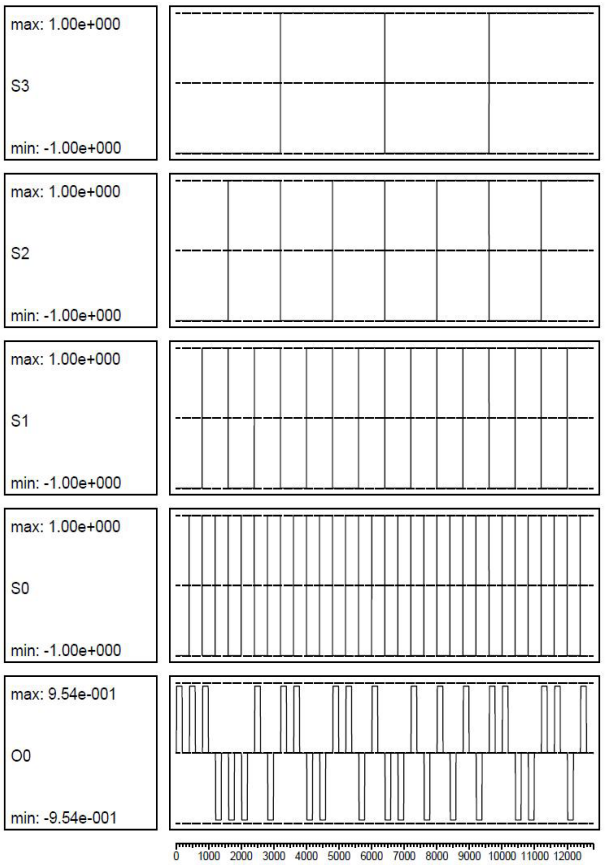

Figure 13. Simulation Result of Logic-Based S-Box.

Table 2. Implementation Results of the LUT-Based S0 S-Box.

	O3	O2	O1	O0	S0 S-Box
Complexity(Cells)	1200	1200	1200	1200	4800
Area(μm^2)	2.626	2.626	2.626	2.626	10.504
Delay(Clocks)	10	10	10	10	10

Table 3. Implementation Results of the Logic-Based S0 S-Box.

	O3	O2	O1	O0	S0 S-Box
Complexity(Cells)	1204	1205	758	798	3965
Area(μm^2)	1.7236	1.7236	1.1532	1.1532	5.7536
Delay(Clocks)	8	8	6	6	8

5. Conclusions

The implementation of the Serpent block cipher in Quantum Cellular Automata is investigated here. The main modules of this cryptographic algorithm are implemented in this technology and the implementation results are discussed. The two methods of S-Box design, i.e. LUT-Based and Logic-Based methods are inspected. The Serpent's S-Boxes are designed and simulated by these two methods. The implementation results show that the Logic-Based method has better advantages than LUT-Based method. Its Delay, Complexity and Area are less than the other method. A novel multiplexer is also introduced. This multiplexer is composed of only 31 QCA cells.

REFERENCES

[1] C. S. Lent, P. D. Tougaw, W. Porod and G. H. Bernstein, "Quantum Cellular Automata," Nanotechnology, Vol. 4, No. 1, pp. 49-57, 1993

[2] P. D. Tougaw and C. S. Lent, "Dynamic Behavior of Quantum Cellular Automata," J. Appl. Phys., Vol. 80, No. 8, pp. 4722-4735, Oct. 1996

[3] R. Anderson, E. Biham and L. Knudsen, "Serpent: A proposal for the Advanced Encryption Standard," NIST AES Proposal, 1998

[4] P. D. Tougaw, C. S. Lent, and W. Porod, "Bistable Saturation in Coupled Quantum-dot Cells," J. Appl. Phys., Vol. 74, No. 5, pp. 3558-3565, Sep. 1993

[5] P.D. Tougaw and C.S. Lent, "Logical Devices Implemented Using Quantum Cellular Automata," J. Appl. Phys., Vol. 75(3), pp. 1818-1825, 1994

[6] K. Hennessy and C. S. Lent, "Clocking of Molecular Quantum-dot Cellular Automata," J. Vac. Sci. Technol., Vol. 19, No. 5, pp. 1752-1755, Sep. 2001

[7] C. S. Lent and B. Isaksen, "Clocked Molecular Quantum-dot Cellular Automata," IEEE Transaction on Electron Devices, Vol. 50, No. 9, Sep. 2003

[8] M. A. Amiri, M. Mahdavi and S. Mirzakuchaki, "QCA Implementation of a Mux-Based FPGA CLB", Proceedings of International Conference On Nanoscience and Nanotechnology, Australia, Melbourne, pp. 141-144, Feb. 2008

[9] K. Kim, K. Wu, R. Karri, "The Robust QCA Adder Designs Using Composable QCA Building Blocks," IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, No. 1, Jan. 2007

[10] V. Vankamamidi, M. Ottavi, and F. Lombardi, "Two-Dimensional Schemes for Clocking/Timing of QCA Circuits," IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, Vol. 27, No. 1, pp. 34-44, Jan. 2008

[11] R. Anderson, E. Biham and L. Knudsen, "Smart Card. Research and Applications," Berlin/Heidelberg, Springer, 2006

[12] Heumpil Cho and Earl E. Swartzlander, "Adder Designs and Analyses for Quantum-Dot Cellular Automata," IEEE Trans. on Nanotechnology, Vol. 6, No. 3, pp. 374-383, May 2007

[13] Heumpil Cho and Earl E. Swartzlander, "Adder and Multiplier Design in Quantum-Dot Cellular Automata," IEEE Trans. on Computer, Vol. 58, No. 6, pp. 721-727, 2009