Design of Logical Structures and Characteristics analysis of AOI for Quantum Dot Cellular Automata

[1]S.KARTHIGAI LAKSHMI
Electronics and Communication Engg
Anna University, Coimbatore
Assistant Professor,
Sona College of Technology, Salem
INDIA
karthi_lax@yahoo.co.in

[2]G.ATHISHA
Electronics and Communication Engg
Anna University, Madurai
Professor and Head,
PSNA College of Engg and Technology, Dindigul.
INDIA
gathisha@yahoo.com

Abstract: Quantum dot Cellular Automata (QCA) offers a new transistorless computing paradigm in nanotechnology. It has the potential for attractive features such as faster speed , smaller size and low power consumption than transistor based technology .By taking the advantages of QCA we are able to design interesting computational architecture. The basic logic elements used in this technology are the inverter and the majority gate (MG). The majority gate is not a universal gate .Hence another important logic gate introduced in this technology is And-Or-Inverter (AOI).In this paper, we propose design of different logical structures using AOI. The rules are introduced in this paper for easy implementation of AOI circuits for different functions and characteristics of AOI is also defined and analyzed. Design implementations using the AOI gate are compared with the conventional CMOS and majority gate based QCA methodology.

Key-words: QCA, QCADesigner, MG, AOI, CMOS, Computing and Nanotechnology

1. Introduction

Current silicon transistor technology faces challenging problems, such as high power consumption and difficulties in feature size reduction [4][5]. Nanotechnology is an alternative to these problems. QCA is one of the promising new technologies that not only give a solution at nanoscale, but also it offers a new method of computation and information transformation [16]. Since QCAs were introduced in 1993 by lent et al, and experimentally verified in 1997. It has been predicted as one of the future nanotechnologies in semiconductor Industries Association’s International Roadmap for Semiconductors (ITRS).

QCA structures are constructed as an array of quantum cells with in which every cell has an electrostatic interaction with its neighboring cells. QCA applies a new form of computation, where polarization rather than the traditional current, contains the digital information. In this trend, instead of interconnecting wires, the cells transfer the information throughout the circuit. [2]

This paper describes the design of different logical structures in QCA such as AND, OR, NOT, NAND, NOR, EX-OR and EX-NOR. These structures are designed using NNI. The design follows the conventional design approaches but due to the technology differences, they are modified for the best performances in QCA. We develop some standard rules in this paper for design of functions using NNI. Designs of some standard functions are also proposed in this paper and characteristics of AOI are defined and analyzed.

The paper is organized as follows, in section 2, describes the QCADesigner tool. The background of QCA technology is explained in section 3. Section 4, provides the QCA clocking and in section 5, explains the QCA wire crossings. Section 6 shows the design rules for AOI and characteristics of AOI is presented in section 7. Design of some standard functions and corresponding AOI implementations are explained in section 8. Conclusions follow in section 9

2. QCADesigner

QCA logic and circuit designers require a rapid and accurate simulation and design layout tool to determine the functionality of QCA circuits. QCADesigner gives the designer the ability to quickly layout a QCA design by providing an extensive set of CAD tools. As well, several simulation engines facilitate rapid and accurate simulation. It is the first publicly available design and simulation tool for QCA. Developed at the ATIPS Laboratory, at the University of Calgary, QCADesigner currently supports three different simulation engines, and many of the CAD features required for complex circuit design. [5], [11]

Included in the current version of QCADesigner are three different simulation engines. The first is a digital logic simulator, which considers cells to be either null or fully polarized. The second is a nonlinear approximation engine, which uses the nonlinear cell-to-cell response function to iteratively determine the stable state of the cells within a design. The third uses a two-state Hamiltonian to form an approximation of the full quantum mechanical model of such a system. Each of the three engines has a different and important set of benefits and drawbacks. Additionally, each simulation engine can perform an exhaustive verification of the system or a set of user-selected vectors. [10], [11].

3. Background Material

3.1.QCA Basics

QCA computing is based on the electrostatic interaction between QCA cells. The basic QCA cell can be viewed as a square charge container with four quantum dots at the corners with two extra electrons. The coulombic interaction between the two mobile electrons tends to localize the particles in a diagonal pattern. Hence, the QCA cell can be in two stable polarization states as shown in Fig.1. These states can be used to encode binary information. These two states are denoted as cell polarization P= +1and P= -1.By using cell polarization P =+1 to represent logic "1" and P =-1 to represent logic "0," binary information is encoded in the charge configuration of the QCA cell. [4-6].

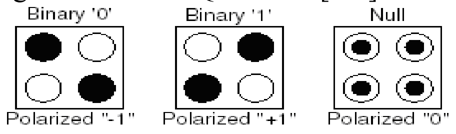


Fig.1.QCA cell polarization.

3.2.QCA Logic Devices

Logic gates are required to build arithmetic circuits. In QCA, inverters and three input majority gates serve as the fundamental gates. The QCA logic primitives include a QCA wire, QCA inverter, and QCA majority gate [4]–[6], as described below.

QCA Wire: The wire is a horizontal row of QCA cells and a binary signal propagates from input to output because of the electrostatic interactions between adjacent cells. The wire shown in Fig.2.1 is known as 90˚ QCA wire. A QCA wire can also be comprised of cells rotated to 45˚ as shown in fig.2.2. In this case, the propagation of the binary signal alternates between a binary 1 and a binary 0 polarization.

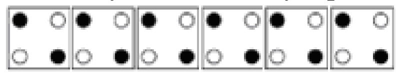


Fig. 2.1 .QCA wire (90˚).

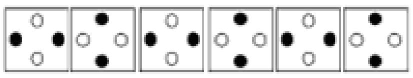


Fig. 2.2 QCA wire (45˚).

QCA Inverter: The QCA cells can be used to form the primitive logic gates. The simplest structure is the inverter shown Fig.3.which is usually formed by placing the cells with only their corners touching. The electrostatic interaction is inverted, because the quantum-dots corresponding to different polarizations are misaligned between the cells.

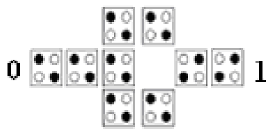


Fig.3.QCA Inverter

QCA Majority Gate: The QCA majority gate computes the function

M (A, B, C) = AB+BC+CA. (1)
and outputs the majority value of its three inputs A, B and C .By fixing the polarization of one input cell to +1 or -1 respectively, AND and OR gates can be computed.

M(A,B,0) = AB (2a)
M(A,B,1) = A+B (2b)

With ANDs, ORs and inverters any logic function can be realized. The majority gate is not a universal gate. It can not realize the logical NOT operation. The functionally complete set is {MV, NOT}.Therefore, the designers have to use separate QCA cell arrangement for realization of the logical NOT. Thus to implement MV with NOT functions, a new gate called AND-OR-Inverter (AOI) is introduced [12].

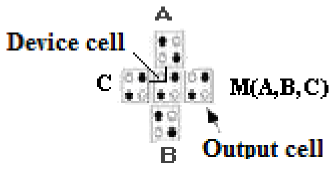


Fig. 4.1 QCA majority gate

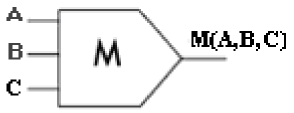


Fig. 4.2 Majority gate Symbol

A layout of a QCA majority gate is shown in Fig. 4.1 along with its symbol. It consists of 4 QCA cells around a center QCA cell. The tendency of the majority device cell to move to a ground state ensures that it takes on the polarization of the majority of its neighbors. The device cell will tend to follow the majority polarization because it represents the lowest energy state.

Nand-Nor-Inverter (NNI)
The NNI gate is a majority gate with two inverted inputs. It gate computes the function

NNI (A, B, C) = M
(A’, B, C’) = A’B+BC’+C’A’. (3)

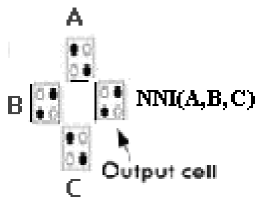


Fig. 5.1 QCA NNI gate

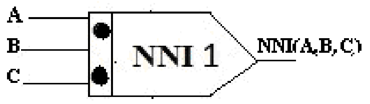


Fig. 5.2 NNI gate Symbol

The NNI gate is a universal gate as shown in Fig.5.1.It can be employed for realizing logical functions. [14], [16] and requires less overhead, for setting the variables.

AND-OR-Inverter (AOI)

This is a 7 cell gate with 5 input cells, one device cell and one output cell. We build the gate from the original 5-cell MV by adding two extra inputs (cells A and C); these two inputs have an inverting effect on the centre cell as it can be seen from the layout of the inverter in Figure 1 that cells in a diagonal orientation tend to exhibit an inverting function [12]. The logic function realized by the AOI gate is:

F=DE+(D+E)(A’C’+A’B+BC’) (4)
= Maj (D, E, Maj (A’, B, C’))

Where Maj () is the 3-input majority function.

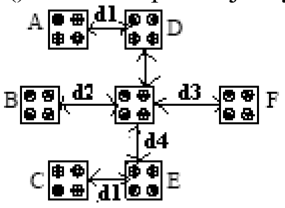


Fig. 6.1 QCA AOI gate

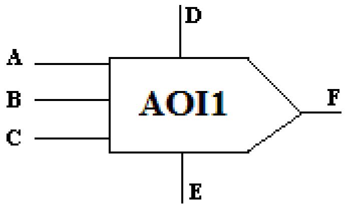


Fig. 6.2 AOI gate Symbol

Although MV can be easily adapted to realize AND or OR, it suffers from the disadvantage that it’s not an universal gate and can not offer the inverting function. Since inverting is expensive in QCA (unlike conventional CMOS), built-in inversion is desirable.

Moreover, in our experiments we found that the MV is not favorable in terms of synthesis. This motivates us to build a complex QCA gate with embedded AND, OR and INV functions, and with better usability for synthesis.

4. QCA Clocking

Clocking of QCA circuits requires a completely different approach than CMOS. A clock provides both synchronization and power gain to QCA circuits .The cells are not powered from any other external source apart from the clock. These clocks have been proposed to control the potential barrier between the dots. The change in potential barrier allows to control the rate at which the electrons quantum mechanically tunnel between the dots in the QCA cell and therefore, the switching of its polarization.

When the clock signal is high the potential barriers between the dots are low and electrons effectively spread out in the cell and no net polarization exists (P = 0).As the clock signal is switched low, the potential barriers between the dots are raised high and the electrons are localized such that a polarization is developed based on the interaction of their neighbours [7], [15].In short when clock is high cell is unlatched and when clock is low cell is latched.

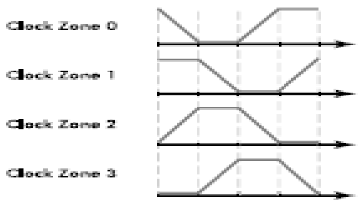


Figure 6.The four phases of the QCA clock

In order to pump information down a circuit in a controllable manner four clocking zones are available as shown in Figure 5. Each of clocking signal is phase shifted by 90 degrees with respect to one before. In this way, the cells are latched in series and propagate information in the same direction. So clocking is essential for QCA circuits.

5. QCA Interconnect

Unlike CMOS technologies, where as metallic interconnects are used to connect transistors together, QCA cells act as both the switching device as well as interconnects. This difference has a significant impact on optimizing QCA computing architectures and the latency of the circuits.

In QCA, there are two interconnect options. [3], [5]. There are coplanar crossings and multilayer crossovers. The coplanar method, crossing of two wires (horizontal and vertical) on the same plane. The vertical wire consists of cells rotated by 45 degrees while the horizontal wire has the same configuration as in Fig.2.1. The regular cell and the rotated cell do not interact with each other when they are properly aligned. By choosing the connection point from rotated cells, either an original or an inverse of the input is available. In a coplanar crossing, there is a possibility of a loose binding of the signal which causes a discontinuity of the signal propagation and there is the possibility of back-propagation from the far side constant input. So putting enough clock zones between the regular cells across the rotated cells is required.

The multilayer crossover is quite straightforward from the design perspective and the signal connection is steadier. The implementation process is less well understood than that for coplanar crossings. The multilayer crossovers use more than one layer of cells like multiple metal layers in a conventional IC. An example layout of coplanar and multilayer wire crossings are shown in Figs. 7 and 8.

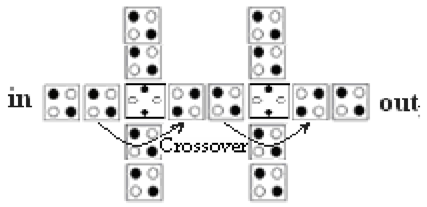


Fig. 7 .Layout of coplanar crossings.

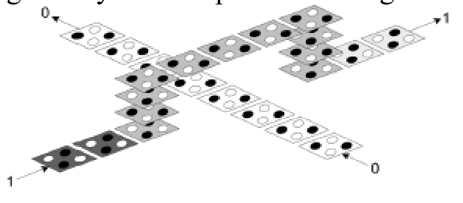


Fig. 8 .Layout of coplanar crossings.

6. Design Rules

The AOI gate rules are proposed here for the fast and easy implementation of complex circuits. The rules are defined as follows. If we set inputs DE =01 or 10 then,

Rule1: If the first two inputs A and B are set to zero then the AOI gate acts as an inverter.
Rule2: Also if the last two inputs B and C are set to zero then the AOI gate act as an inverter.
Rule3: If A=C=0 and B=1or 0 then output of AOI gate is always 1.
Rule4: If B=1 and there is no change in other two inputs then AOI gate act as a NAND gate.
Rule5: If B=0 and there is no change in other two inputs then AOI gate act as a NOR gate. The above said inverter rules are defined by the following table.

Table.1.Rules table for implementation of inverter

RULES	***INPUT***			**OUTPUT**
	A	**B**	**C**	**F**
R1	A	0	0	A’
R2	0	0	C	C’
R3	0	B	0	1

Rules R1 and R2 indicates how inverter is obtained and R3 indicates irrespective of B, if A and C are set to zero, output is always 1.The operations in table.3 is achieved by only one NNI gate. The table also defines the function of AOI gate, when any of two inputs are set to zero. Hence the gate is now said to be a single input AOI gate.

7. Characteristics of AOI

The logic function of AOI gate is realized by the ( ).Based on the equation we have to define the logical characteristics of AOI.
1. If inputs D and E are equal to zero, the corresponding output is zero.
2. If inputs DE=11 then the corresponding output is one.
3. If DE=01 or 10 then the corresponding output is M (A’, B, C’). Therefore, the logical expressions are available only in these combinations of D and E.
The above characteristics are defined in the following table.

Table.2.Characteristics table (1)

INPUT					**OUTPUT**
A	**B**	**C**	**D**	**E**	**F**
A	B	C	0	0	0
A	B	C	0	1	M (A’,B,C’) or NNI(A,B,C)
A	B	C	1	0	M(A’,B,C’) or NNI(A,B,C)
A	B	C	1	1	1

With DE=01 or 10, the output of AOI gate becomes M (A’, B, C’) or NNI (A, B, C). Hence, the AOI gate not only offers the inverter function. It can be also used to build Majority gate as well as NNI gate. Thus, keeping DE=01 or 10, all following operations are applicable for both NNI and AOI.

The logic function of AOI gate is realized by the (3).Based on the equation we have to define the logical characteristics of NNI. The standard logical AND and OR operation can be obtained by two AOI gates, is define by the following table.

Table.3.Characteristics table (2)

INPUT			**OUTPUT 1**	**APPLYING RULES**	**FINAL OUTPUT 2**
A	1	C	(AC)’	R1	AC
A	*0*	*C*	*(A+C)’*	*R1*	*A+C*

The first AOI gate output is obtained as below
1) If we set B=1, then output 1 is OR of two inverted inputs or inversion of AND of two inputs.
2) If we set B=0, then output 1 is AND of two separate inverted inputs or inversion of OR of two inputs.
Then applying inverter rule R1, the AND and OR operation can be obtained at the output of second AOI gate.
With any input A and BC =00 or 11 then the output of AOI gate is A’. Similarly, if B=01 or 10 then the output of AOI gate is equivalent to SR latch. The characteristic is defined by the following table. The same is applicable for input C also.

Table.4.Characteristics table (3)

INPUT			**OUTPUT**
A	**B**	**C**	**F**
A	0	0	A’
	0	1	0
	1	0	1
	1	1	A’

The rules R1 and R2 have already defined how inverter function is obtained from AOI gate. From the above table we know that there is another possibility of getting inverter function using AOI. Here the selection of inputs A, B and C are important. Based on these inputs we have to obtain different functions from single AOI gate.

With any input B and AC =00 then the output of AOI gate is 1 and if AC =11 then the output of AOI gate is 0. Similarly, when AC=01 or 10 then the output of AOI gate is B. The characteristic is defined by the following table.

Table.5.Characteristics table (4)

INPUT			OUTPUT
A	**B**	**C**	**F**
0	B	0	1
0		1	B
1		0	B
1		1	0

Based on the above mentioned rules and characteristics of AOI gate we have to obtain the efficient AOI expression for any given three-variable Boolean function amenable to QCA implementation. The simplified AOI expressions for some functions are given in the table.

Table.6.AOI gate functions

INPUT			**TERMS IN AOI EQUATION**			**OUTPUT**
A	**B**	**C**	**A'B**	**BC'**	**A'C'**	**Y**
1	B	C	0	BC'	0	BC'
0	B	C	B	BC'	C'	B+C'
A	B	1	A'B	0	0	A'B
A	B	0	A'B	B	A'	A'+B
A	1	C	A'	C'	A'C'	(AC)'
A	0	C	0	0	A'	(A+C)'

6. Design of Logical Structures Using NNI

Logical gates are the basic elements that built the digital system. The gate is a circuit that is able to operate on a number of binary inputs in order to perform a particular logical function. Here different logical structures such as AND, OR, NOT, NAND, NOR, EX-OR and EX-NOR are designed using AOI.

Inverter Design:

The majority gate suffers from the disadvantage that it cannot offer the inverting function. This motivated to build a QCA gate AOI with embedded AND, OR and INV functions. The design of inverter function using AOI is define below.

We know that,
F = AOI (A,B,C,D)=DE+(D+E)(A'C'+A'B+BC')= DE +(D+E)(NNI(A, B, C)) (5)
Where NNI (A,B,C)=M(A',B,C')=A'B+BC'+C'A'
If we set DE=01 or 10 in equation (4) then the resulting equation is
F=AOI (0, 1, A, B, C) = NNI (A, B, C) = M (A', B, C') = A'B+BC'+C'A' (6)
If we set any B=C=0 to above equation then,
F=AOI (0, 1, A, 0, 0) = A'.
The corresponding AOI gate implementation is show in Fig.9.

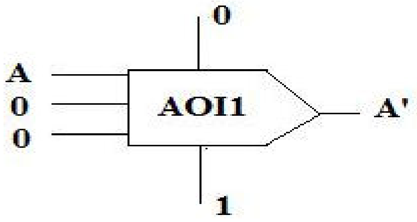


Fig.9. Inverter using AOI

AND and OR implementation
In AND gate design, If we set B=1 then equation (6) becomes,
F= AOI1 (A, 1, C) = A'+C'+C'A'
= A'+C' (1+A')
= A'+C'
= (AC)' (7)
Then the output of AOI1 is given as one of input to AOI 2.Next we set the other two inputs of AOI2 to zero, which acts as an inverter. Hence it gives the function of AND gate. The AOI - AND gate implementation is shown in Fig.10.
Therefore, AOI2 (AOI1, 0, 0) = AOI2 ((AC)', 0, 0) = AC. (8)

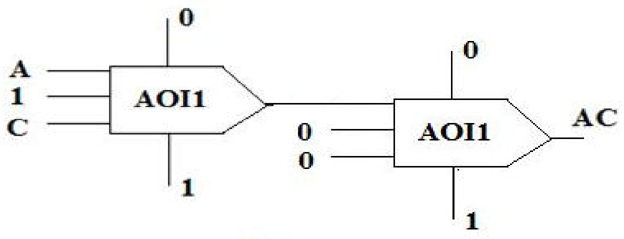


Fig.10. AND gate using AOI

In the case of OR gate design, If we set B=0 then equation (3) becomes,
AOI1 (A,0,C)=C'A'=(A+C)'. (9)
Then the output of AOI1 is given as one of input to AOI2.Next we set the other two inputs of AOI2 to zero, which acts as an inverter. Hence it gives the function of OR gate. The AOII - OR gate implementation is shown in Fig.11.
Therefore,
AOI2 (AOI1,0,0)=AOI2((A+C)',0,0)=A+C. (10)

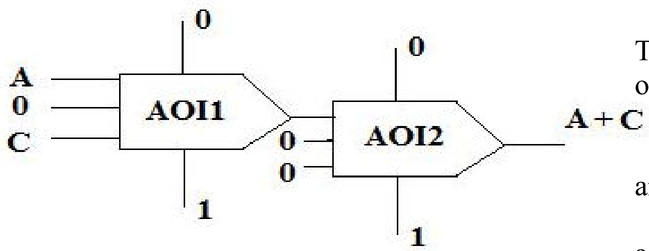


Fig.11. OR gate using AOI

NAND and NOR implementation:

The NAND function is the complement of AND functions. It is realized by connecting AND gate followed by an inverter. Similarly the NOR gate is realized by connecting OR gate followed by an inverter. But in AOI gate there is no need of separate inverter because it is an universal gate. Hence a single AOI gate is used to implement NAND and NOR gates. Therefore the complexity of a circuit can be minimized.

The equation (8) and equation (10) gives the NAND and NOR operations respectively. The NAND gate is realized by fixing the AOI gate input B=1 and B=0 for NOR gate. The AOI-NAND and NOR gate implementations are as shown in Fig.12.and 13.

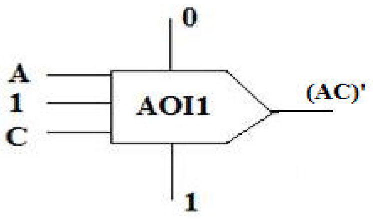


Fig.12. NAND gate using AOI

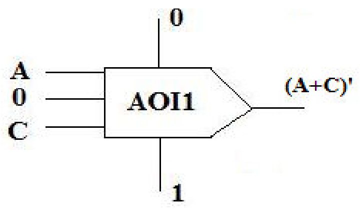


Fig.13. NOR gate using AOI

XOR and Ex-NOR implementation

The XOR is a logical operation on two operands that results in a logical value of true, if and only if one of the operands has a value of true. This forms a fundamental logic gate in many operations to follow. The Ex-NOR function is the complement of XOR function. It is realized by connecting XOR gate followed by an inverter. The logic function for XOR is,

$A\oplus B$ = A'B+B'A (11)

AOI Algorithm:

The AOI gate expression for above equation is obtained by the following algorithm,
1. If we set C=1 then, AOI1 (A, B, 1) = A'B
2. .If we set C=1and interchange variables A and B then AOI2 (B, A, 1) = B'A
3. The outputs of AOI1 and AOI2 are given as input to AOI3 and third input of AOI3 is set to zero. Then AOI3=AOI (AOI1, AOII2, 0) = (A'B+B'A)'
4. The outputs of AOI3 is given as input to AOI4 and second and third inputs of AOI3 is set to zero. Then AOII4=AOI (AOI3, 0, 0) = A'B+B'A

Therefore the complete NNI gate expression for $A\oplus B$ is,
$A\oplus B$ =AOII4((AOI3(AOI1(A,B,1),AOI2(B,A,1),0), 0,0) (12)
The logic function for EX-NOR is,
$(A\oplus B)'$=AB+A'B' (13)
The NNI gate realization of EX-NOR is similar to XOR gate. The third step is used to give the EX-NOR output. Therefore the complete AOI gate expression for $(A\oplus B)'$ is given by,
$(A\oplus B)'$ = (AOI3 (AOI1 (A, B, 1), AOI2 (B, A, 1), 0) (14)

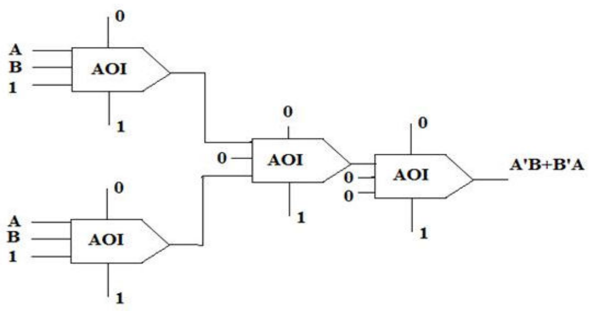


Fig.14. XOR gate using AOI

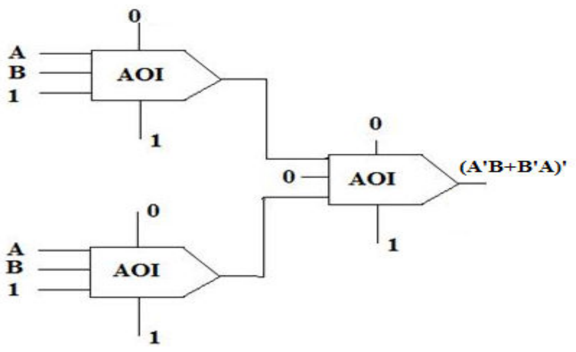


Fig.14. XNOR gate using AOI

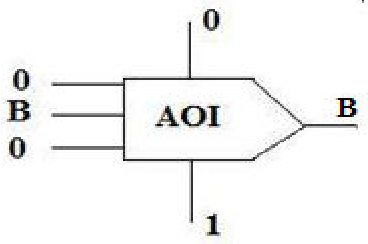
Table.1.Comparision of Logical Structures

Type of Design	Total Number of Gates Required for the Design of Logical structures						
	Inverter	AND	OR	NAND	NOR	XOR	Ex-NOR
CMOS Design	Transistors						
	2	4	4	4	4	8	8
Majority gate Design	Separate device	One majority Gate	One majority gate	1 majority Gate and 1 Inverters	1majority Gate and 1 Inverters	3 majority Gates and 2 Inverters	3 majority Gates and 3 Inverters
AOI Gate Design	One AOI Gate	2 AOI gates	2 AOI gates	One AOI gate	One AOI Gate	4 AOI Gates	3 AOI gates

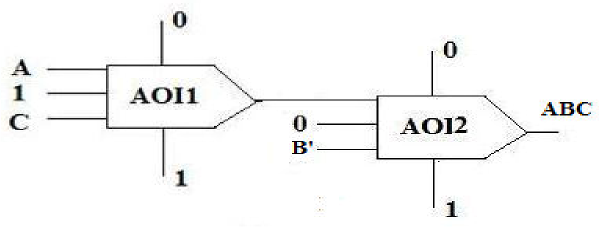
7. Standard Functions Using AOI

The AOI gate is a 5-input (A, B, C, D and E) and by setting the two inputs D and E are equal to 01 0r 10, then the three variables A, B and C to facilitate the conversion of a sum-of-products expression to minimized AOI logic. Based on that to obtain the efficient AOI expression for any given three-variable Boolean function amenable to QCA implementation. The simplified AOI expressions for some standard functions and the corresponding gate representations are given below.

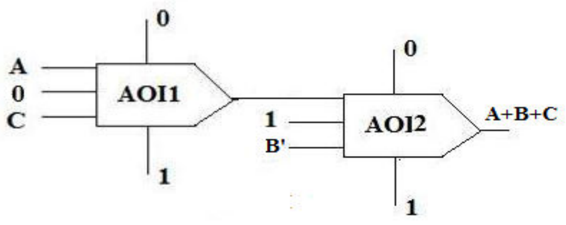
1. F=B
   F=AOI (0, B, 0)



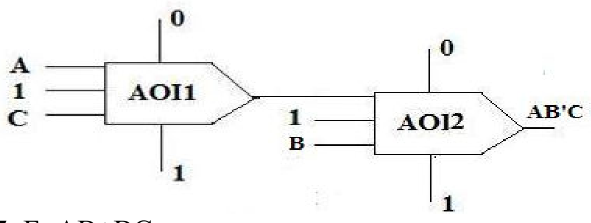
2. F=ABC
   F= AOI2 (AOI1 (A, 1, C), 0, B')



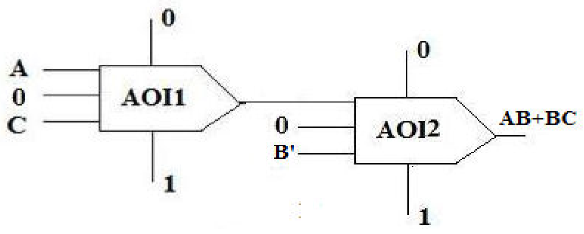
3. F=A+B+C
   F= AOI 2(AOI1 (A, 0, C), 1, B')



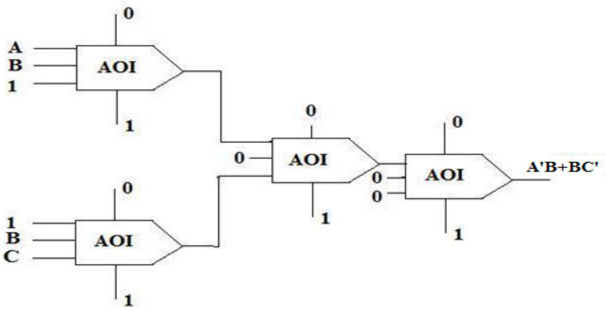
4. F=AB'C
   F= NNI 2(NNI1 (A, 1, C), 1, B)



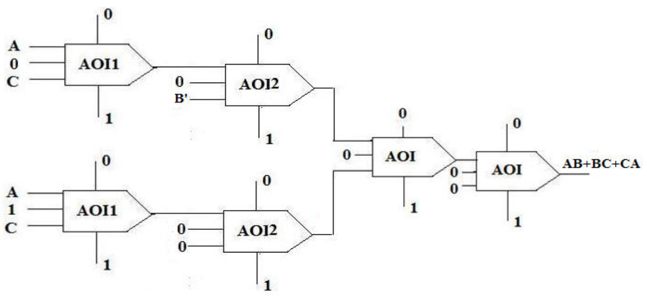
5. F=AB+BC
   F= AOI 2(AOI1(A, 0, C), 0, B')



6. F=A'B+BC' F= AOI4 (AOI3 (AOI 2(AOI1 (A, B, 1), AOI2 (1, B, C), 0), 0, 0)



7. F=AB+BC+CA
F= AOI6 (AOI5 (AOI 2(AOI1 (A, 0, C), 0, B'), AOI4 (AOI3 (A, 1, C), 0, 0), 0), 0, 0)



11. Conclusion

In this paper, we have proposed rules for easy design of circuits using AOI gates. We have developed some logical structures in this paper and these structures are designed based on the proposed rules. The characteristics of AOI gate have been defined and analyzed. The proposed rules and characteristics can be used for direct application to the design of logic functions in QCA. The standard three variable Boolean functions have been developed using AOI gates and the corresponding functions are implemented using AOI gates. Furthermore, an algorithm has been introduced for design of XOR and XNOR structures using AOI gates. The proposed algorithm can be used to develop arithmetic structures such as adder, subtractor and multiplier. Hence it can be used to construct arithmetic and logical unit. In future it can be used to build nano computers.

Reference

[1] International Technology Roadmap for Semiconductors. (ITRS) 2005 [Online]. Available: http://www.itrs.net

[2] K. Walus, G.A. Jullien, and V. Dimitrov. Computer arithmetic structures for Quantum cellular automata. *Asilomar Conference on Signals, Systems, and Computers*, November 2003.

[3] W. Wang, K. Walus, and G.A. Jullien. Quantum-dot cellular automata adders. *IEEE Nano Conference*, August 2003.

[4] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "A method of majority logic reduction for quantum cellular automata," *IEEE Trans.Nanotechnol.*, vol. 3, no. 4, pp. 443–450, Dec. 2004.

[5] Heumpil Cho, Student Member, and Earl E. Swartzlander, Jr., Fellow, IEEE. "Adder Design and Analyses for Quantum-Dot Cellular Automata", *IEEE Trans. Nano.*, Vol.6, No.3, may 2007.

[6] K. Walus, G. Schulhof, G. A. Jullien, R. Zhang, and W. Wang, "Circuit design based on majority gates for applications with quantum-dot cellular automata," in *Conf. Rec. 38th Asilomar Conf. Signals, Systems and Computers*, 2004, vol. 2, pp. 1354–1357.

[7] W. J. Townsend and J. A. Abraham, "Complex gate implementations for quantum dot cellular automata," in *Proc. 4th IEEE Conf. Nanotechnology*, 2004, pp. 625–627.

[8] Heumpil Cho, Student Member, and Earl E. Swartzlander, Jr., Fellow, IEEE. "Adder Design and Analyses for Quantum-Dot Cellular Automata", *IEEE Trans. Nano.*, Vol.6, No.3, may 2007.

[9] Kyosun Kim, Kaijie Wu, and Ramesh Karri "The Robust QCA Adder Designs Using Composable QCA Building Blocks"IEEE transactions on computer-aided design of integrated circuits and systems, vol. 26, no. 1, January 2007.

[10] K. Walus, G. Schulhof, and G. A. Jullien, "High level exploration of quantum-dot cellular automata (QCA)," in *Conf. Rec. 38th Asilomar Conf. Signals, Systems and Computers*, 2004, vol. 1, pp. 30–33.

[11] K. Walus, T. Dysart, G. Jullien, and R. Budiman, "QCADesigner: A rapid design and simulation tool for quantum-dot cellular automata,"*IEEE Trans.*

Nanotechnology, vol. 3, no. 1, pp. 26–29, Mar. 2004.

[12] Jing Huang, Mariam Momenzadeh, Mehdi B, "Design and Characterization of An And Or Inverter Gate for QCA Implementation "GLSVLSI'04, April 26–28, 2004,

[13] Characterization, test and logic synthesis of AOI gate design for QCA implementation .IEEE trans.comput. aided Des.Integrated ircuits.Vol.24.13.pp.1881-1893, Dec.05.

[14] E.N.Ganesh, Lal Kishore and M.J.S. angachar, "Implementation of Quantum cellular automata combinational and sequential circuits using Majority logic reduction method". International Journal of Nanotechnology and Applications, Volume 2, Number 1, 2008.

[15]Pijush Kanti Bhattacharjee, "Digital Combinational Circuits Design By QCA Gates" , International Journal of Computer and Electrical Engineering, Vol. 2, No. 1, February, 2010.

[16] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "Performance comparison of quantum-dot cellular automata adders," in *Proc. IEEE Int. Symp. Circuits and Systems*, 2005, vol. 3, pp. 2522–2526.

[17] Pijush Kanti Bhattacharjee, "Use of Symmetric Functions Designed by QCA Gates for Next Generation IC" International Journal of Computer Theory and Engineering", Vol. 2, No. 2 April, 2010, 1793-8201