NANOSYSTEMS: PHYSICS, CHEMISTRY, MATHEMATICS, 2021, 12 (5), P. 583–597

Realization of combinational logic circuits using standard functions in quantum dot cellular automata

Ratna Chakrabarty, Niranjan Kumar Mandal

Institute of Engineering & Management, Department of Electronics & Communication Engineering, Salt Lake Electronics Complex, Sector V, Kolkata, 700091, India

ratna.chakrabarty@iemcal.com, niranjanmandal54@gmail.com

DOI 10.17586/2220-8054-2021-12-5-583-597

A set of functions commonly used in basic circuits are defined as standard functions. They are so called because different combinations of different logic circuits are designed using these functions. They are also used to realize combinational logic circuits and most Boolean expressions. In this paper, 13 standard functions are discussed with their various applications using Quantum Dot Cellular Automata known as QCA which is currentlya familiar nanotechnology for its ultra-low power consumption and high speed operations. The designed functions are analyzed with area, latency and cell count. Energy calculations have been done with the suitable input for its stable operation. Algorithms are also established for the designed functions to realize in QCA technology.

Keywords: standard functions, majority gate, universal gate, encoder, adder, comparator.

Received: 16 September 2021

Revised: 27 September 2021

1. Introduction

Quantum Dot Cellular Automata technology aimed to provide an alternative to current silicon transistor technology for its low power operation. It has created a system in which electrons are bound in cells; polarization is the alignment of electrons within them. Each piece of information will be transmitted through the wire as a change in polarization. It also developed majority voters, inverters and wires. When the first phase of development was completed, it became the priority of designers and technologists to surpass previous work. The major challenge of the development phase is increasing spatial and temporal efficiency. Scientists aimed to find a more sophisticated design that could performthe same task faster. Also, during the dawn of the VLSI era, using less and less space to implement more and more logic would become unavoidable in order to stay competitive in the shrinking race. Quantum Dot Cellular Automata is mainly implemented in software program. But some researchers in early 1990's, mainly C. S. Lent, exclaimed for a physical realization of quantum dot cells. This technique, since its inception, gained tremendous popularity and it was first fabricated in 1993 [1]. By using the concepts of physics and nanotechnology, Lent realized quantum cells for computation using very low power. The fundamental structural unit of Quantum Dot Cellular Automata is the QCA Cell. Each cell consists of four conductors in the form of metal islands [2, 3] that forms the quantum dots. The paper is framed with the overview of the basic concept of Quantum dot cellular automata, and then it discussed 13 standard functions. The designs are implemented with the help of layered NAND and NOR gates in QCA. Next, it describes the QCA based circuit implementation and energy analysis of the 13 standard functions and different parameters of the designed circuits. The last section concludes the paper with the future applications.

2. Materials and methods

2.1. QCA cell

A Quantum Dot cell has 4 dots with two additional electrons. It has two configurations; one of which is used to transmit the +1 binary information and another as 0. The four phase clock scheme is needed to carry the information through the cells. Cells and clocks together can act as a medium to create device-device interaction with very low power dissipation (on the order of 10 to 20 Joules). QCA devices exist and multi-device circuits have been demonstrated at low temperatures. Fig. 1 shows the four-dot QCA cell.

In a QCA cell, two ground state polarizations are possible. They are labeled as $+1$ and -1. The formula for calculating polarization is given below:

$$P = \frac{(p_1 + p_3) - (p_2 + p_4)}{p_1 + p_2 + p_3 + p_4}.$$

The equation given above shows the calculation of polarization of a QCA cell. Here, p_i is the charge of the i-th quantum-dot ($p_i = 1$ if an electron is present, otherwise it is 0). These polarizations are represented by logic-0 and logic-1 (in other words, binary 0 and binary 1).

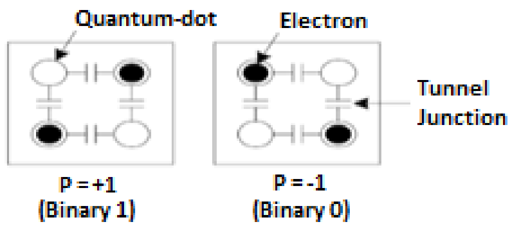


FIG. 1. QCA cells

2.2. Basic components of QCA

The QCA cells and wires can be used to realize the basic components of Quantum dot cellular automata which are used to implement all the combinational and sequential logic circuits of QCA. The first stepping stone of digital electronics is a basic gate. The tools to make basic gates are Majority Gate or Majority voter and Inverter.

Majority Gate or Majority Voter is the fundamental component of QCA [4]. It takes odd numbers of inputs and processes them to find the majority of them and sends that bit to the output pin. Depending on the application, different kinds of majority gates are possible: 3-input majority gate A three-input majority gate has three inputs A, B and C and output Y. The output expression Y can be given by Y = AB + BC + CA. Therefore, if more than one input of them is HIGH, output is HIGH. Fig. 2 shows the majority gates implemented in QCA.

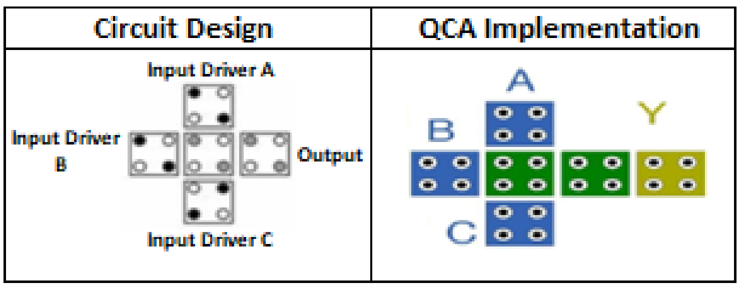


FIG. 2. 3 input Majority gate in QCA

2.3. QCA Clocking

Clocking is an integral part of Quantum-dot Cellular Automata logic circuit design. It not only provides the synchronization and power to run the circuit, it also helps to propagate the input in a pre-defined way. A proper electric field should be introduced for clocking to regulate the electron flow through the channel. By applying the correct time-varying voltage, we can clock the circuit's activity to produce a chain of information. The clocking electric field at the QCA surface is referred to as the clocking field and the signal on the buried wires that will provide such a field is the four-phase signal. Every cycle consists of 4 phases, namely SWITCH, HOLD, RELEASE and RELAX [5]. In traditional electronic systems, timing is controlled through a reference signal (i.e., the clock); however, timing in QCA is accomplished by clocking in four distinct and periodic phases [6, 7]. Fig. 3 shows the clocking scheme of QCA.

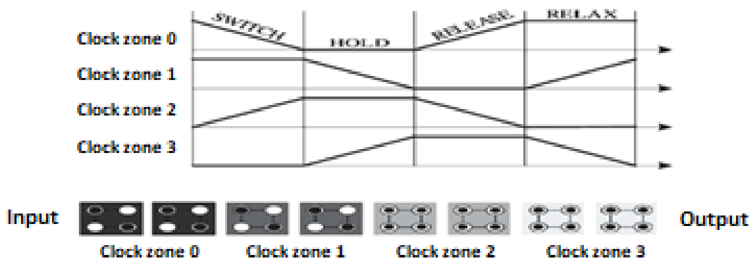


FIG. 3. Different clocking zones in QCA

3. Need for creation of Universal Logic

The two basic needs behind the modification of QCA designs are space complexity and time complexity. Space complexity happens if a simple design takes many cells and hence large space. In that case, it becomes the target of a designer to reduce the circuit without sacrificing its efficiency or input output relation. Moreover, the larger the design more will be the potential energy. Therefore, smaller circuits are also required to attain stability. Time complexity is the amount of time for the input to reach output. If a circuit becomes big, the time to reach the input state to the output increases. This also increases the delay in the circuit and the computation time. In today's world, we always seek for a design with better speed and low computation time, i.e., the speed of operation must be high. Also, the big circuit increases the latency, i.e., the number of clocks required to accomplish any job. NAND and NOR gates are considered to be universal gates, i.e., any logic gate can be generated using NAND and NOR gate. Though the majority voter actually can contain 3 elements, we generally use only 2 elements to generate AND and OR gate. To generate this, we need 6 cells. But we can get the NAND and NOR gates using our modified method using 5 cells [8]. Thus, in a big circuit, this new approach reduces the cell number and the area covered drastically. Also, any Boolean function is expressed using canonical sum-of product (SOP) or canonical product-of sum (POS) form. SOP and POS are generally realized using NAND logic and NOR logic respectively. Therefore, it is always better to use universal gates, because not only it simplifies Boolean expression, it reduces area covered and cell count as well. Fig. 4 shows the QCA implementation of Layered NAND and NOR gate.

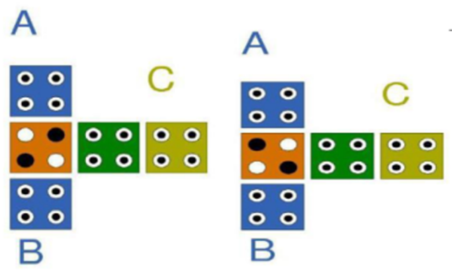


FIG. 4. QCA implementation of (a) Layered NAND Gate (b) Layered NOR Gate

4. Standard functions

The commonly used standard functions [9] are listed below:

1. D = AB'C;
2. D = AB;
3. D = A'B'C' + A'BC;
4. D = A'BC + AB'C';
5. D = A'B + BC';
6. D = AB' + A'BC;
7. D = A'B'C' + A'BC + ABC';
8. D = A;
9. D = AB + BC + CA;
10. D = A'B + B'C;
11. D = A'B + BC + AB'C';
12. D = AB + A'B';
13. D = A'B'C' + ABC' + AB'C + A'BC.

Implementation of Combinational Logic Digital Circuits using Standard Functions:

- *Half Adder*: Here we are using standard function 2 i.e. C = AB for carry and complement of function 12 i.e. S = A'B + AB' for sum.
- *Full Adder*: The sum is S = A XOR B XOR C, which is the complement of function 13 and carry is D = AB + BC + CA which is standard function 9.
- *Half Subtractor*: Here we are using complement of standard function12 i.e. D = A'B + AB' for difference and an indirect application of standard function 2, B$_{out}$ = A'B for borrow.
- *Full Subtractor*: The difference is D = A XOR B XOR C, which is the complement of function 13 and borrow is D = C$\times$(A XNOR B) + A'B. (A XNOR B) comes from standard function 12.
- *Binary to Gray Code Converter*: The basic logic gate used is XOR gate which is the complement of function 12.
- *Gray to Binary Code Converter*: 4 bit gray to binary code converter uses 3 XOR gates which is the complement of standard function 12.
- *1 bit Comparator*: It has 3 outputs; equal, greater than and less than; denoted by E, G & L respectively. E = AB + A'B', which is standard function 12. G = AB' and L = A'B which are indirect application of standard function 2.
- *1: 2 Demultiplexer*: It has two outputs; D0 and D1 where D0 = S'D and D1 = SD which are indirect application of standard function 2.
- *1: 4 Demultiplexer using 1:2 Demultiplexer*: 1:4 demultiplexer requires 3 1:2 demultiplexers. It has 4 output lines which are chosen according to the status of control lines and uses the application of standard function 1.
- *2: 4 Decoder*: A 2:4 decoder circuit has 2 inputs and 4 outputs; denoted by X, Y, A, B, C and D respectively. Here A = X'Y', B = X'Y, C = XY', D = XY which are indirect application of standard function 2.
- *4: 2 Encoder*: A 4: 2 encoder has 4 inputs and 2 outputs; namely A, B, C, D, X & Y respectively. Here X = A + B [i.e. (A'B')'] and Y = NOT (B + D) [i.e.(B'D')] which are indirect application of standard function 2.
- *Even Parity Generator/Checker*: If number of 1's present in input data is even, the parity bit is 1, otherwise 0. This is established using XOR gate which is the complement of standard function 12. Therefore, for a 3 bit number, the parity generator output will be P = A XOR B XOR C.
- *Odd Parity Generator/ Checker*: If number of 1's present in input data is odd, the parity bit is 1, otherwise 0. This is established using XOR gate which is the standard function 12. Therefore, for a 3 bit number, the parity generator output will be P = NOT (A XOR B XOR C).
- *2:1 Multiplexer*: A 2:1 multiplexer has 2 inputs and 1 select line denoted by A, B and S respectively and only 1 output denoted by Y. The output is written as Y = S'A + SB which is an indirect application of standard function 10.
- *Basic Gates*: All the basic Gates, e.g. AND, OR, XOR etc. can be realized using the standard functions.

5. Analysis of 13 standard functions

13 standard functions have been discussed previously [9] with the use of NAND and NOR gates in QCA.
Table 1 show the functions and algorithm behind it for making that function in QCA based circuits.

Table 1: Realization of standard functions

Function	**Algorithm**
Function: D = AB'C Inputs A and C will be sent to a universal gate with polarization +1 to create intermediate value i1 = NAND (A, C). Then, the output will be passed through a NOR gate along with input B to get final result. D = NOR (i1, B).	S1: i1 = UG (A, C, $+1$); S2: D = UG (B, i1, -1).
Function: D = AB First of all, inputs A and B will be sent to a universal gate with polarization $+1$ to create intermediate value i1 = NAND (A, B). Then, the output will be passed through a NOT gate along to get final result. D = NOT (i1).	S1: i1 = UG (A, B, $+1$); S2: D = NOT (i1).

Function: D = A'B'C' + A'BC
First of all, inputs C and B will be sent to a universal gate with polarization $+1$ to create intermediate value i1 = NAND (C, B). Then, this intermediate value will be used to form another two intermediate values i2 and i3. i2 = NAND (C, i1); i3 NAND (B, i1). Next, these two values will be passed through a NAND gate to get another intermediate result. i4 = NAND (i2, i3). Finally, the output will be passed through a NOR gate along with input A to get final result. D = NOR (i4, A).
	S1: i1 = UG (C, B, $+1$);
	S2: i2 = UG (C, i1, $+1$);
	S3: i3 = UG (i1, B, $+1$);
	S4: i4 = UG (i2, i3, $+1$);
	S5: D = UG (i4, A, -1).

Function: D = A'BC + AB'C'
First of all, inputs C and B will be sent to 2 universal gates with polarization $+1$ and -1 to create intermediate value i1 = NAND (C, B) and i2 = NOR (C, B). Then, these intermediate values will be used to form another two intermediate values i2 and i3. i3 = NOR (A, i1); i4 = NAND (A, i2). Next, i3 will be passed through a NOR gate to get another intermediate result. i5 = NOT(i3). Finally, i5 and i4 will be passed through a NAND gate along with input A to get final result. D = NAND (i5, i3).
	S1: i1 = UG (C, B, $+1$);
	S2: i2 = UG (C, B, -1);
	S3: i3 = UG (i1, A, -1);
	S4: i4 = UG (i2, A, $+1$);
	S5: i5 = NOT (i3);
	S6: D = UG (i4, i5, $+1$).

Function: D = A'B + BC'
First of all, inputs A and C will be sent to a universal gate with polarization $+1$ to create intermediate value i1 = NAND (A, C). Then, the output will be passed again through a NAND gate along with input B to get final result. i2 = NAND (i1, B), finally, the output will be passed through a NOT gate along to get final result. D = NOT (i2).
	S1: i1 = UG (A, C, $+1$);
	S2: i2 = UG (C, B, $+1$);
	S3: D = NOT (i2).

Function: D = AB' + A'BC
Let us consider AB'as d1 and A'BC as d2.
For decomposing d1, firstly, inputs A and complement of B will be sent to a universal gate with polarization $+1$ to create intermediate value i1 = NAND(A, NOT(B)).
Then, the output d1 is generated by complementing the previous output i1 i.e. d1 = NOT(i1). For decomposing d2, firstly, inputs B and C will be sent to a universal gate with polarization $+1$ to create intermediate value i2 = NAND(B, C).
Then, the output i3 is generated by complementing the previous output i1 i.e. i3 = NOT(i2). Next, i3 will passed through another NAND gate along with complement of A i.e. i4 = NAND(i3,A').
Then, the output d2 is generated by complementing the previous output i1 i.e. d2 = NOT(i4). The final output D is produced by passing the outputs d1 and d1 through an NOR gate using polarization -1 and complementing the output. D = NOT(NOR(d1, d2)).
	S1: i1 = UG (A, NOT (B), $+1$);
	S2: d1 = NOT (i1);
	S3: i2 = UG (B, C, $+1$);
	S4: i3 = NOT (i2);
	S5: i4 = UG (i3, NOT(A), $+1$);
	S6: d2 = NOT (i4);
	S7: D = UG (d1, d2, -1).

Function: D = A'B'C'+ A'BC + ABC'
Firstly, we calculate the XOR function, in a similar way, as in function no. 3 (up to step 4). Inputs C and A will be sent to a universal gate with polarization +1 to create intermediate value i1 = NAND (C, A) and at the same they are sent to NOR gate using polarization −1 i.e. i5 = NOR(A,C).
Then, this intermediate value will be used to form another two intermediate values i2 and i3. i2 = NAND (C, i1); i3 = NAND (A, i1). Next, these two values will be passed through a NAND gate to get another intermediate result. i4 = NAND (i2, i3).
Next, the i4 is passed through a NAND gate along with B i.e. i6 = NAND(B,i4) and its complement is passed along with i5 i.e. i7 = NAND(i5,NOT(B)).
Finally, both the outputs are passed through another NAND gate. D = NAND(i6,i7).
	S1: i1 = UG (C, A, +1);
	S2: i2 = UG (C, i1, +1);
	S3: i3 = UG (i1, A, +1);
	S4: i4 = UG (i2, i3, +1);
	S5: i5 = UG (C, A, −1);
	S6: i6 = UG (i4, B, +1);
	S7: i7 = UG (NOT(B), i5, +1);
	S8: D = UG (i6, i7, +1).

Function: D = A Firstly the single input A is passed through a NAND gate i.e. polarization +1 with positive Boolean value 1. Then the output is complemented.	S1: i1 = UG (A, 1, +1); S2: D = NOT (i1).

Function: D = AB + BC + CA
First of all, inputs A, B and C will be sent to different combinations of universal NAND gate with polarization +1 to create intermediate value i1 = NAND (A, C), i2 = NAND (A, B) and i3 = NAND (B, C). The 2 out of 3 outputs are then passed through another NAND gate to produce another output which is passed with the remaining of the first 3 output to give the final one.
	S1: i1 = UG (C, A, +1);
	S2: i2 = UG (C, B, +1);
	S3: i3 = UG (B, +1, A);
	S4: i4 = UG (i2, i3, +1);
	S5: D = UG (i4, i1, +1).

Function: D = A'B + B'C
First of all, inputs A and B will be sent to a universal gate with polarization +1 to create intermediate value i1 = NAND(A, B). Then, another output will be produced using a NAND gate using complement of B and C as input. i2 = NAND(C, B). Finally, the 2 outputs will be passed through a NAND gate along to get final result. D = NAND(i1,i2).
	S1: i1 = UG (B, A, +1);
	S2: i2 = UG (A, NOT (B), +1);
	S3: D = UG (i1, i2, +1).

Function: D = A'B + BC + AB'C'
Firstly, let us take M as NAND (A,C') i.e., the inputs A and the complement of C are passed through a NAND gate i.e. with polarization +1.
Then, this resultant is passed with B twice, once through a NOR gate and another through NAND gate i.e. i1 = NAND(B,M) and i2 = NOR(B,M).
The final output is formed when i1 and i2 are passed again through a NAND gate.
	S1: M = UG (NOT(C), A, +1);
	S2: i1 = UG (M, B, +1);
	S3: i2 = UG (B, M, −1);
	S4: D = UG (i2, i3, +1).

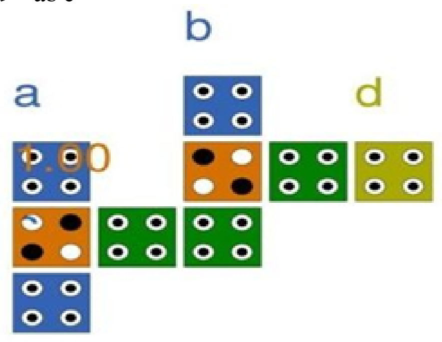
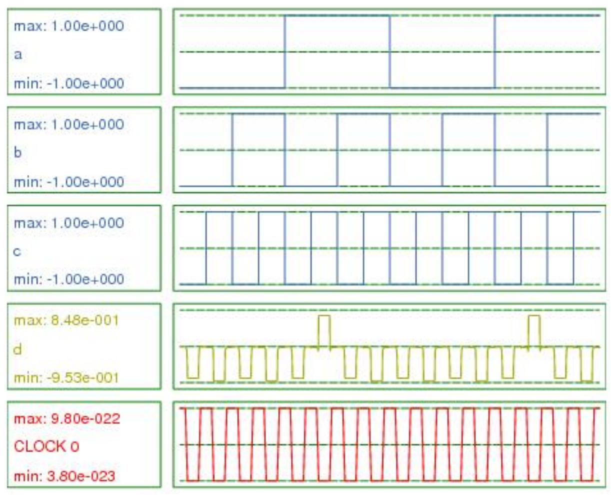
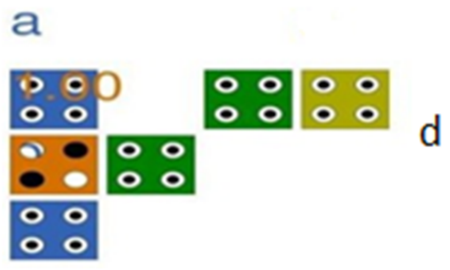
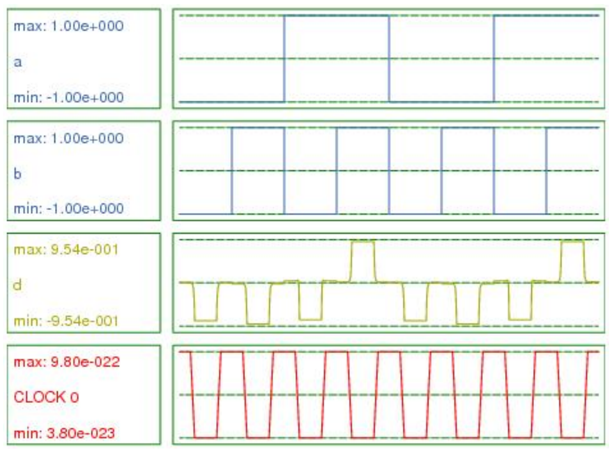
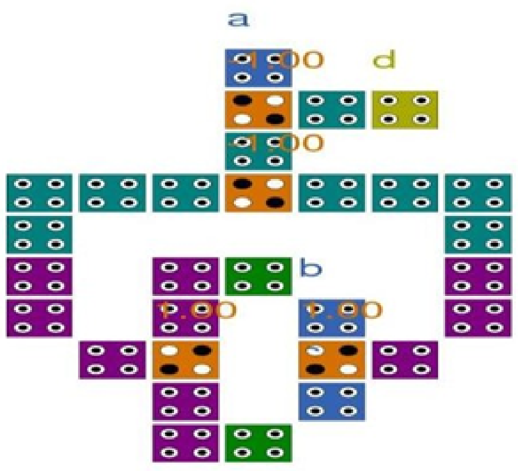
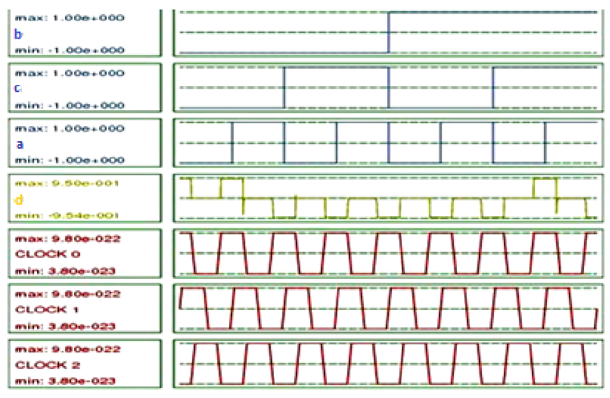
Function: D = AB + A'B'
First of all, inputs A and B will be sent to a universal gate with polarization +1 to create intermediate value i1 = NAND (A, B). Then, this intermediate value will be used to form another two intermediate values i2 and i3. i2 = NAND (A, i1); i3 = NAND (B, i1). Next, these two values will be passed through a NAND gate to get final result. i4 = NAND (i2, i3). Finally, the output i4 is complemented to produce the desired outcome.
	S1: i1 = UG (C, A, +1);
	S2: i2 = UG (C, i1, +1);
	S3: i3 = UG (i1, A, +1);
	S4: i4 = UG (i2, i3, +1);
	S5: D = NOT (i4).

Function: D = A'B'C'+ ABC'+ AB'C + A'BC
Firstly let us take M as NOR(NAND(B',C'), NAND (B,C)) i.e., the inputs B and C and their complements are passed through 2 NAND gates i.e., with polarization +1 i.e., i1 = NAND(B,C) and i2 = NAND(B',C').
After that, the resultants i1 and i2 is passed through a NOR gates (−1 polarization) i.e. M = NOR(i1,i2). Then this resultant is passed with A twice, once through a NOR gate and another through NAND gate i.e. i3 = NAND(A,M) and i4 = NOR(A,M). The final output is formed when i3 and i4 are passed again through a NAND gate.
	S1: i1 = UG (C, B, +1);
	S2:
	i2 = UG(NOT(C),NOT(B),+1);
	S3: M = UG (i1, i2, −1);
	S4: i3 = UG (A, M, +1);
	S5: i4 = UG (A, M, −1);
	S6: D = UG (i3, i4, +1).

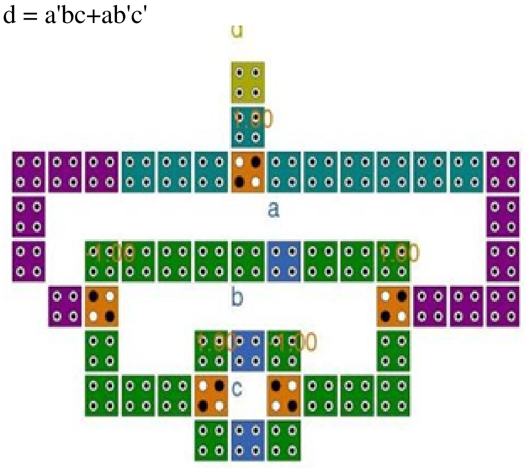
6. Results and discussion

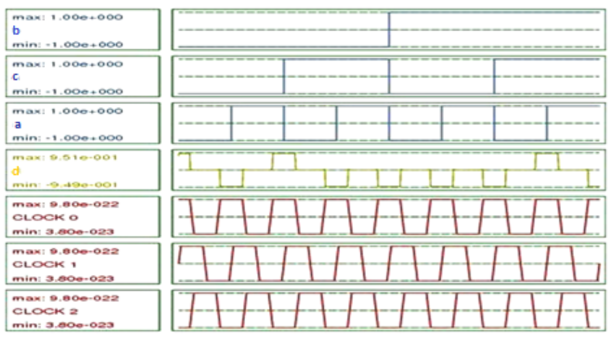
Table 2 shows the circuit diagrams using QCA and the simulated outputs of the corresponding circuits

Table 2: Circuit implementation in QCA and their simulated output

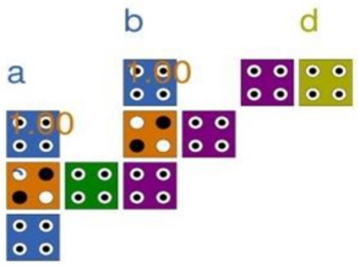
QCA Layout	**Simulation Result**
d = ab'c 	
d = ab 	
d = a'b'c'+a'bc 	

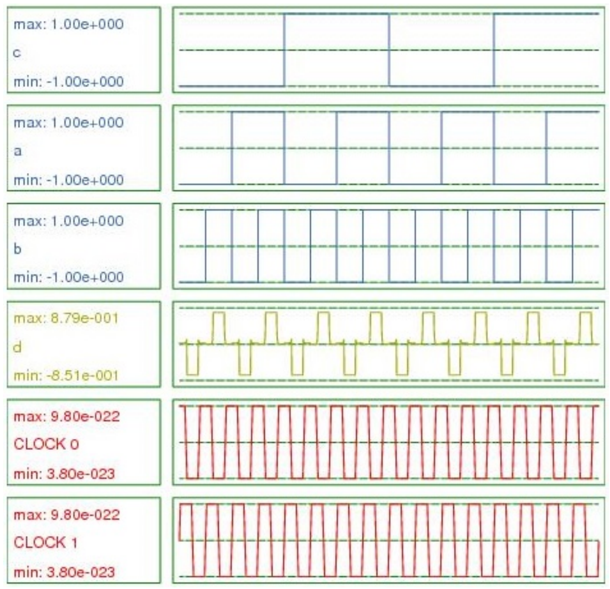
d = a'bc+ab'c'



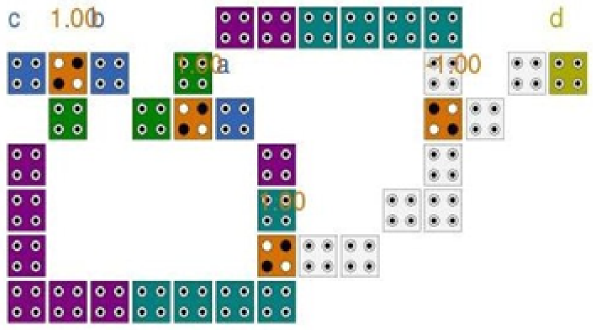


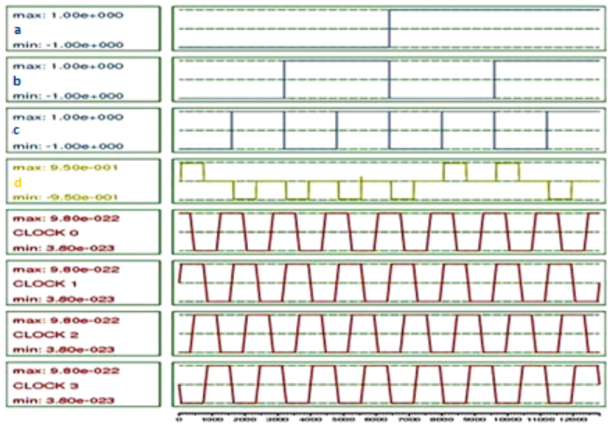
d = a'b+bc'



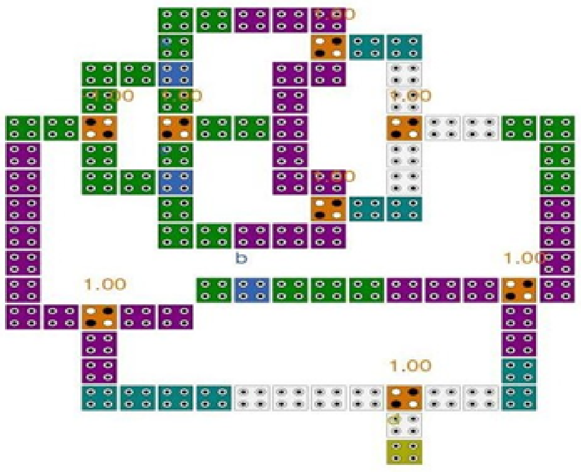


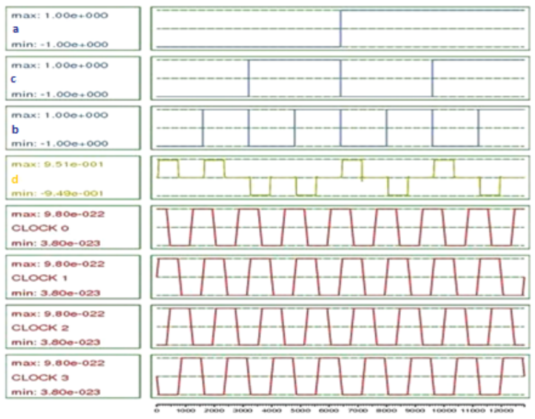
d = ab'+a'bc



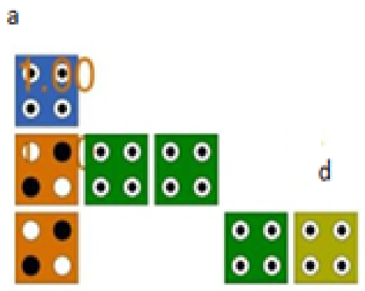


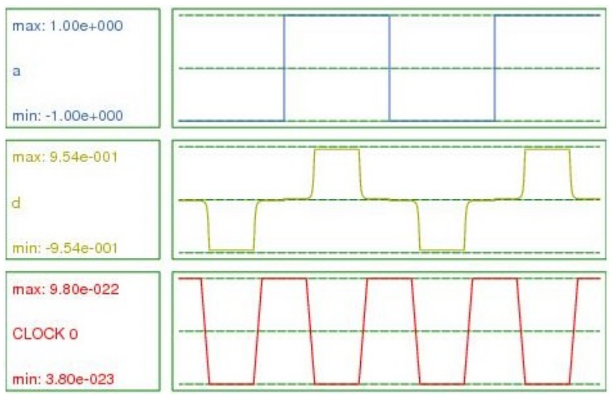
d = a'b'c'+a'bc+abc'



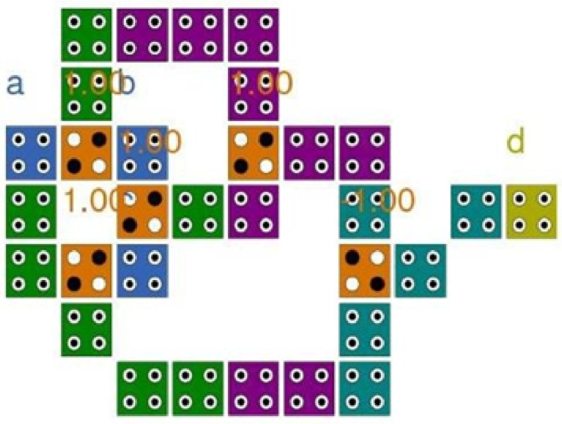


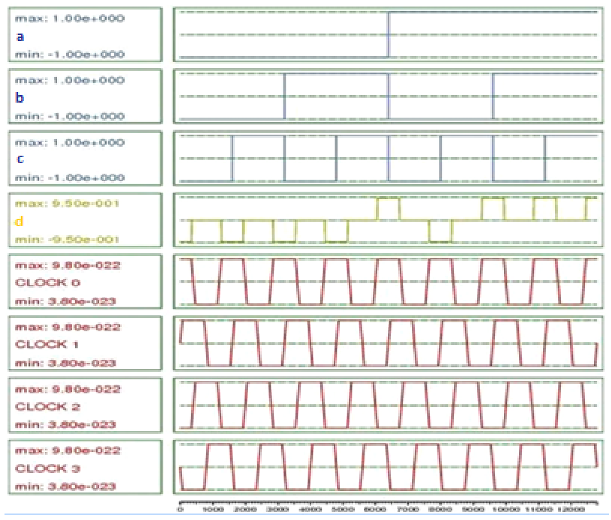
d = a



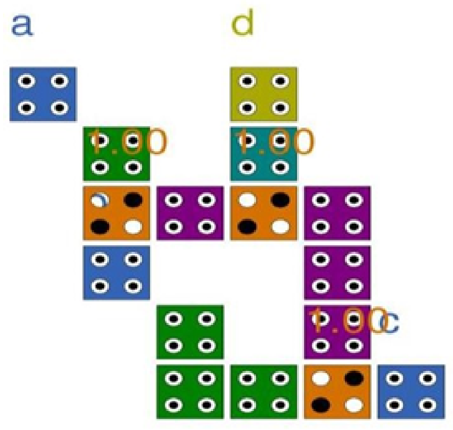


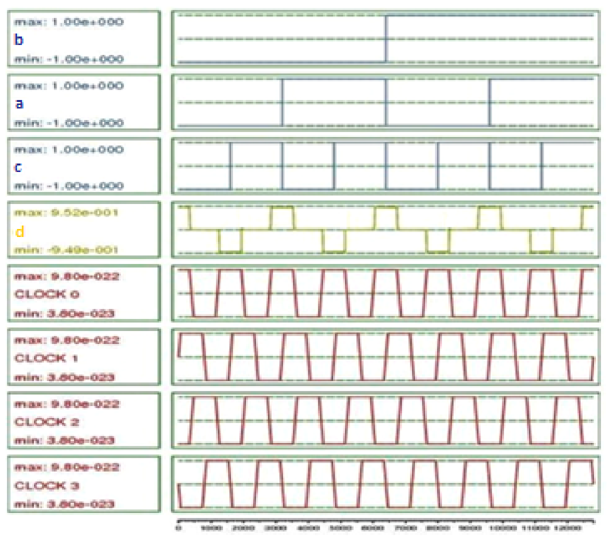
d = ab+bc+ca



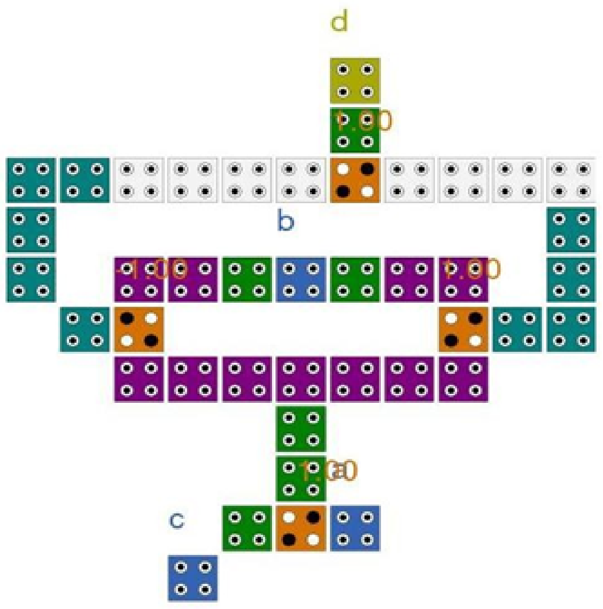


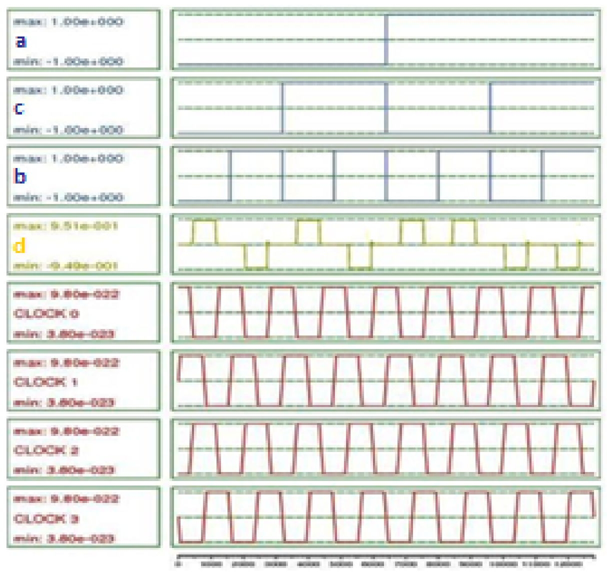
d = a'b+b'c

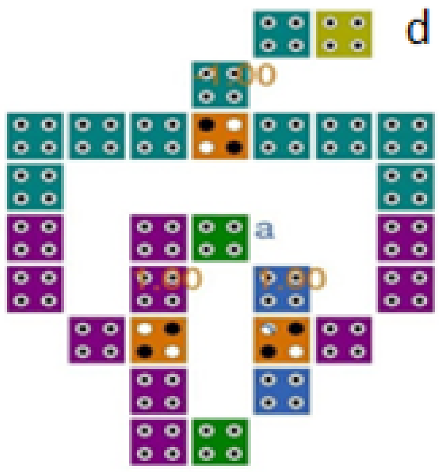
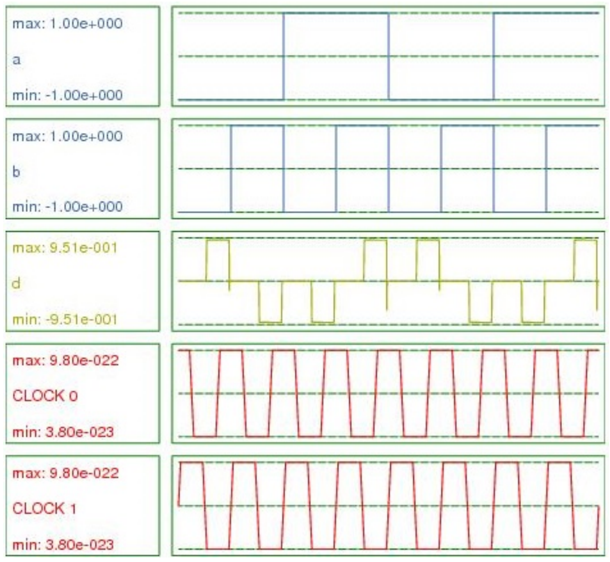
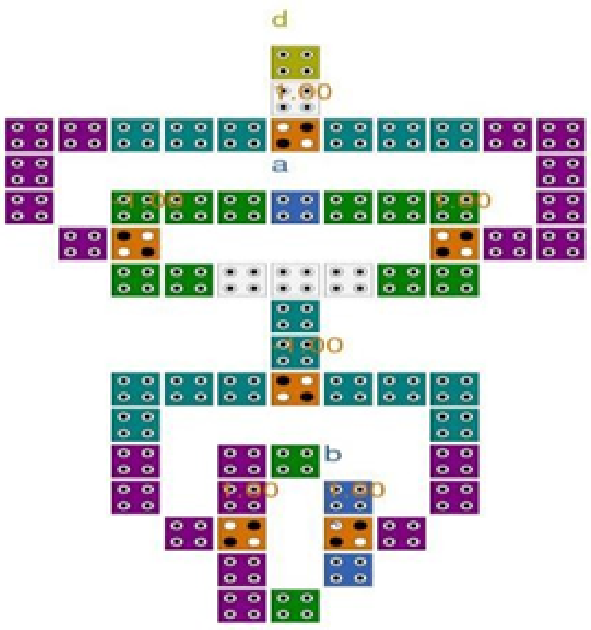
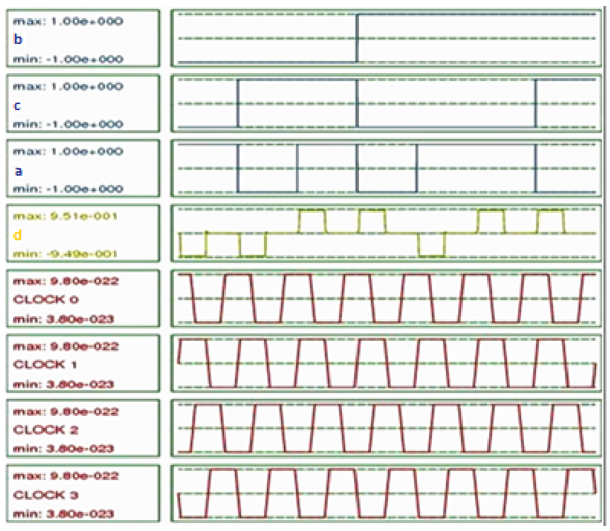




d = a'b+bc+ab'c'





d = ab+a'b' 	
d = a'b'c'+abc'+ab'c+a'bc 	

7. Observation

From the implementation of standard functions, the following parameters can be found which is shown in Table 3.

8. Energy Analysis of 13 standard functions using QCA

In this section, we deal with the energy calculations and analysis based on said calculations of 13 standard functions as specified below.

8.1. Analysis for D = AB'C

Figure 5 shows the energy calculation bar chart for the function D = AB'C.

As we see, this function has 3 inputs, A, B, and C. So we have to find the potential energy for the $2^3 = 8$ different resultant states. Firstly, we fix the input as (0, 0, 0), and determine the configuration of the electrons in the cells. This gives us the total potential energy for this particular state of device cells. Similarly, we find the total potential energy for all the remaining states of device cells.

TABLE 3. Information obtained from 13 Standard Function design using QCA designer tool

Design Name	Length (in nm)	Breadth (in nm)	Cell Count	Area(in nm^2)	Latency
d = ab'c	105	98	9	10290	0.25
d = ab	85	78	6	6630	0.25
d = a'b'c'+a'bc	221	138	30	30498	0.75
d = a'bc+ab'c'	278	201	51	55878	0.75
d = a'b+bc'	118	101	10	11918	0.50
d = ab'+a'bc	278	141	37	39198	1.00
d = a'b'c'+a'bc+abc'	341	304	95	103664	1.00
d = a	81	78	6	6318	0.25
d = ab+bc+ca	198	138	26	27324	0.75
d = a'b+b'c	145	134	16	19430	0.75
d = a'b+bc+ab'c'	241	218	42	52538	1.25
d = ab+a'b'	201	138	28	27738	0.75
d = a'b'c'+abc'+ab'c+a'bc	341	218	63	74338	1.00

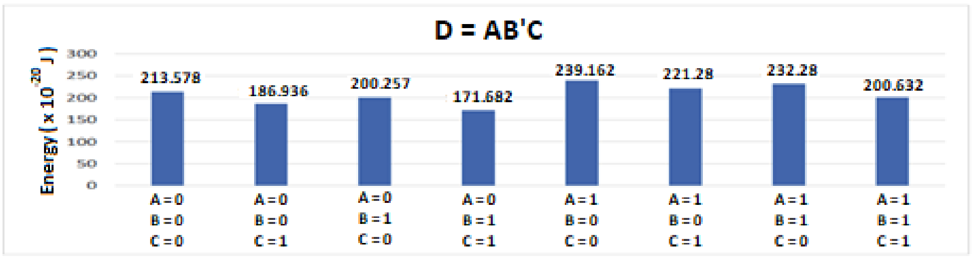


FIG. 5. Energy calculation for the function D = AB'C

The graph showing the energy (10^{-20} Joules) and the state configuration is represented in Fig. 5.

So, we see that the potential energy associated with the state (A = 0, B = 1, C = 1) is minimum (171.688 u), in comparison to the energy of the other device states.

Thus, we can conclude that it is the most stable operating state for this particular function, D = AB'C.

8.2. Analysis for D = AB

The function D = AB, has 2 inputs, A and B. So we have to find the potential energy for the $2^2 = 4$ different resultant states. Firstly, we fix the input as (0, 0), and determine the configuration of the electrons in the cells. This gives us the total potential energy for this particular state of device cells. From Fig. 6, we see that the potential energy associated with the state (A = 0, B = 1) is minimum (120.863 u), in comparison to the energy of the other device states. Thus, we can conclude that it is the most stable operating state for this particular function, D = AB.

8.3. Analysis for D = A'B'C'+A'BC

The function D = AB'C'+A'BC has 3 inputs, A, B, and C. So we have to find the potential energy for the $2^3 = 8$ different resultant states. As, mentioned above, after finding the potential energy for different cell states of the device, the resulting output is plotted. The graph from Fig. 7 is showing the energy axis as (10^{-20} Joules) in y axis and the state configuration as x axis

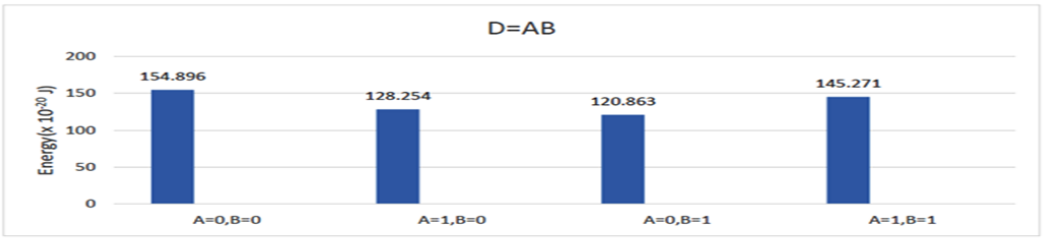


FIG. 6. Energy calculation for the function D = AB

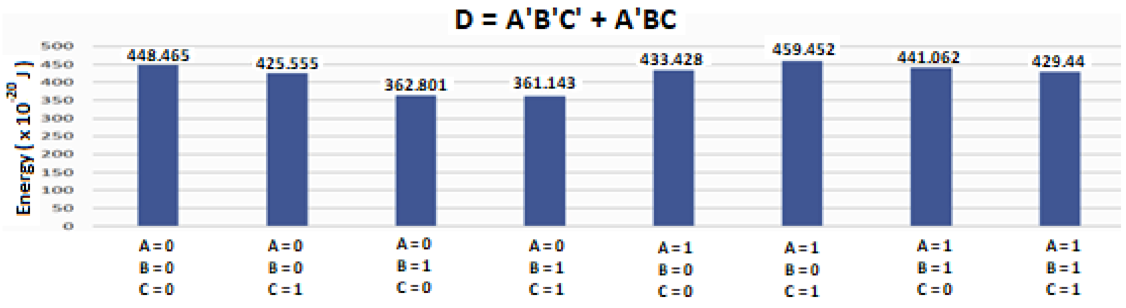


FIG. 7. Energy calculation for the function D = A'B'C'+A'BC

So, we see that the potential energy associated with the state (A = 0, B = 1, C = 1) is minimum (361.143 u), in comparison to the energy of the other device states. Thus, we can conclude that it is the most stable operating state for this particular function, D = A'B'C'+A'BC.

8.4. Analysis for D = A'BC+AB'C'

The function D = A'BC+AB'C'has 3 inputs, A, B, and C. So we have to find the potential energy for the $2^3 = 8$ different resultant states. Depending on the configuration of the input states, which defines the configuration of the electrons in the cells, we'll again have different potential energy for those states. The total energy for those states is then plotted. The graph showing the energy (10^{-20} Joules) and the state configuration is represented as Fig. 8.

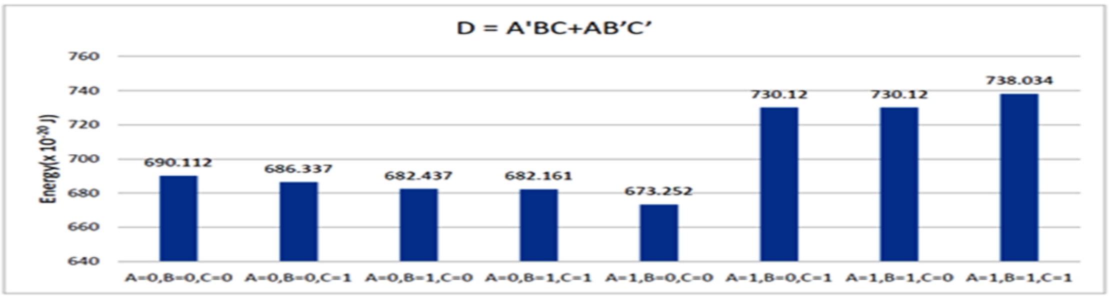


FIG. 8. Energy calculation for the function D = A'BC+AB'C'

So, we see that the potential energy associated with the state (A = 1, B = 0, C = 1) is minimum (673.252 u), in comparison to the energy of the other device states. Thus, we can conclude that it is the most stable operating state for this particular function, D = A'BC+AB'C'.

In this way, we calculate the energy analysis for every standard functions which is shown in Table 4.

9. Comparative study

A comparative study of the existing methodology and the proposed methodology is given in tabular form. The other methods are already implemented and they are reversible gates. Different numbers of unit blocks required to

TABLE 4. Energy analysis of the standard function

Standard function	**Input condition**	**Minimum energy**
D = AB'C	A = 0, B = 1, C = 1	171.688 u
D = AB	A = 0, B = 1	120.863 u
D = A'B'C'+A'BC	A = 0, B = 1, C = 1	361.143 u
D = A'BC+AB'C'	A = 1, B = 0, C = 1	673.252 u
D = A'B+BC'	A = 0, B = 1, C = 1	220.574 u
D = AB'+A'BC	A = 1, B = 0, C = 1	480.87 u
D = A'B'C'+A'BC+ABC'	A = 1, B = 1, C = 1	1004.012 u
D = A	A = 1	111.931 u
D = AB+BC+CA	A = 0, B = 1,C = 1	494.544u
D = A'B+B'C	A = 0, B = 0,C = 1	299.402 u
D = A'B+BC+AB'C'	A = 1, B = 0, C = 1	439.262 u
D = AB+A'B'	A = 1, B = 0	536.628 u
D = A'B'C'+ABC'+ AB'C+ A'BC	A = 0, B = 1, C = 1	985.084 u

implement 13 standard functions for each of them as well as the number of blocks used by proposed logic design are given in Table 5.

TABLE 5. Comparison with different gates [11]

Functions	**CQCA**	**FREDKIN**	**PERES**	**Proposed design**
D = AB'C	2	2	2	2
D = AB	1	1	1	1
D = A'B'C'+ A'BC	6	3	4	4
D = A'BC+AB'C'	4	4	3	5
D = A'B+BC'	3	5	5	2
D = AB'+A'BC	3	3	8	4
D = A'B'C'+A'BC+ABC'	7	5	5	8
D = A	1	1	1	1
D = AB+BC+CA	1	8	11	5
D = A'B+B'C	3	1	4	3
D = A'B+BC+AB'C'	6	4	2	4
D = AB+A'B'	3	2	3	3
D = A'B'C'+ABC'+ AB'C+ A'BC	9	11	4	6
Average number of gates	**3.7692**	**3.8462**	**4.0769**	**3.6923**

Here, it can be observed that an average number of gates in our proposed design are mostly less than the average number of gates in other logic realization. The less the average number of gates used, the more will be the efficiency and less will be the space complexity.

10. Conclusions

In this article, we have described 13 standard functions which are the necessary building blocks for various combinational and sequential circuits. This work is an extension of the previous work as referred in [9]. Here, we have shown the energy calculations to verify the robustness of the design and also calculated the cell count, area consumption and latency to complete the design. We have also discussed the implementation of various combinational and sequential circuits using the standard functions. Comparative study has been done with the other gates. In future work, we will show the applications of standard functions for other logic circuits and detail design and simulations using QCA based nanotechnology.

References

[1] Lent C.S., Tougaw P.D., Porod W., Bernstein G.H. Quantum cellular automata. *Nanotechnology*, 1993, **4**, P. 49–57.

[2] Majeed A.H., Zainal M.S., Bernstein E.A. Quantum-dot Cellular Automata: Review Paper. *Int. Journal of Integrated Engineering*, 2019, **11** (8), P. 143–158.

[3] Laajimi R. Nano architecture of Quantum-Dot Cellular Automata (QCA) Using Small Area for Digital Circuits. In *Advanced Electronic Circuits, Principles, Architectures and Applications on Emerging Technologies*, 2018.

[4] Jayalakshmi R., Amutha R. An Optimized High Input Majority Gate Design in Quantum-Dot Cellular Automata. *Int. Journal of Engineering and Manufacturing Science*, 2018, **8** (1), P. 63–75.

[5] Sheikhfaal S., Angizi S., et al. Designing efficient QCA logical circuits with power dissipation analysis. *Microelectronics Journal*, 2015, **46** (6), P. 462–471.

[6] Sridharan K., Pudi, Vikramkumar. *Design of Arithmetic Circuits in Quantum Dot Cellular Automata Nanotechnology*, Springer, 2015.

[7] Macucci M. *Quantum Cellular Automata. Theory, Experimentation and Prospects*. Imperial college Press, 2006.

[8] Mukherjee C., Sukla A.S., et al. Layered T full adder using Quantum-dot Cellular Automata. *IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2015, P. 1–6.

[9] Chakrabarty R., Bhattacharjee A., Kundu A., Ganguly R. Implementation of Standard Functions Using Universal Gate in QCA Designer. *1st Int. Conf. on Electronics, Materials Engineering and Nano-Technology (IEMENTech)*, 2017, 8076966.

[10] QCADesigner 2.0. `URL: https://qcadesigner.software.informer.com/2.0/`.

[11] Das J., De D. Optimized Design of Reversible Gates in Quantum Dot-Cellular Automata: A Review. *Reviews in Theoretical Science*, 2016, **4** (3), P. 279–286.