

IET Circuits, Devices & Systems



Special issue Call for Papers

Be Seen. Be Cited.
**Submit your work to a new
IET special issue**

Connect with researchers and
experts in your field and share
knowledge.

Be part of the latest research
trends, faster.

Read more



The Institution of
Engineering and Technology

Systematic cell placement in quantum-dot cellular automata embedding underlying regular clocking circuit

Dhrubajyoti Bhowmik¹  | Jayanta Pal²  | Mrinal Goswami³  | Pinaki Sen¹  |
Apu Kumar Saha¹  | Bibhash Sen⁴ 

¹National Institute of Technology, Agartala, Tripura, India

²Department of Information Technology, Tripura University, Suryamaninagar, Tripura, India

³Department of Systemics, School of Computer Science, University of Petroleum and Energy Studies, Bidholi, Dehradun, India

⁴Department of Computer Science & Engineering, National Institute of Technology, Durgapur, West Bengal, India

Correspondence

Jayanta Pal, Department of Information Technology, Tripura University, Suryamaninagar, Tripura, India.
Email: jayantapal@tripurauniv.in

Bibhash Sen, Department of Computer Science & Engineering, National Institute of Technology, Durgapur, West Bengal, India.
Email: bibhash.sen@cse.nitdgp.ac.in

Abstract

Quantum-dot cellular automata (QCA) is gaining worldwide popularity due to its higher device concentration, lower power indulgence, and better switching speed. The information flows in QCA with the polarization state defined by the placement of electrons instead of current flow. The cell interaction principle under the influence of the clocking zone controls the cell placement during QCA circuit design. Most of the designs reported so far used random cell placement, which has no fabricating sense. Moreover, it is equally important for a cell placement algorithm to follow a realistic, regular underlying clocking scheme to maintain proper routing channels. In this regard, a systematic cell placement for the combinational logic circuit in QCA is proposed using the widely accepted underlying regular clocking scheme, universal, scalable and efficient (USE). The proposed method traverses all possible paths of a combinational logic circuit from output to input to build the desired circuit generating the automatic cell layout. A grid of clock zone of USE clock scheme is considered in the initial layout driving different paths of the circuit and finally evolves as the desired circuit. The generated automatic QCA layouts are competitive in occupied area, delay, and its cell count, which advocates the significance of such a scheme.

1 | INTRODUCTION

Over the most recent couple of decades, the Complementary Metal Oxide Semiconductor (CMOS) based digital industry enjoys fast switching speed of digital transistors reducing the feature size from micron to sub-micron, sub-micron to the nano-meter level. In recent years, CMOS technology has faced physical limitations in feature sizing [1,2] such as, short channel impact at the nano-scale period, ultra-thin gate oxides, leakage currents etc. Consequently, in the recent past, several new emerging nanotechnologies have been extensively studied, which can be scaled beyond CMOS level. Quantum-dot cellular automata (QCA) is one of the evolving nanotechnologies which having the property of extremely higher device density, lesser power dissipation, and higher switching speed [3].

In the recent past, QCA designs have been extensively investigated, realizing arithmetic circuits [4,5], RAM [6–8], Flip-Flop [9–11], etc. However, most of these circuits have

been realized with manual cell placement without using any placement algorithm [12–14]. A few works have been investigated targeting the automatic synthesis of QCA circuits [15–17]. However, these works have used irregular/random clocking zones, which restrict data flow to only one direction hence decreasing the scalability and flexibility of these designs.

Trindade et. al. proposed a placement algorithm for QCA circuits utilizing regular clocking [18]. This is the first attempt where they have adopted a regular clocking scheme that supports bi-directional data flow. The algorithm starts processing from output to input, but there is no methodology for multi-output circuits. Moreover, the algorithm cannot handle re-convergence of fan-outs without using some prior knowledge of cell placement; thus, unable to realize an automatic cell Placement and Routing (P & R) is an important part of the VLSI physical design cycle [15,16,19]. Placement includes the exact placement of modules minimizing the total area, delay, congestion, etc. In contrast, routing provides the

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2021 The Authors. *IET Circuits, Devices & Systems* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

interconnection of these modules. However, the same P&R techniques cannot be applied in QCA due to its cell interaction principle.

In QCA, cell placement on substrates is determined by the placement algorithm, whereas the QCA clock controls the routing of these cells. A four-phased irregular clocking scheme is utilized to synchronize the flow of information. However, irregular/random clock zones cannot maintain proper routing channels due to its unidirectional behaviour. It is necessary to follow a regular clocking system to facilitate proper routing paths [14,17]. A Universal, Scalable, and Efficient (USE) clocking scheme has been introduced in [20], which has the advantage of bi-directional routing of information, maintaining the integrity of QCA clocking.

All these aforementioned factors have inspired us to inculcate an automatic cell placement algorithm for the QCA logic circuit towards the real-time realization/fabrication. The key aspects of the proposed work is given below:

- An automatic cell placement algorithm for the QCA logic circuit is proposed utilizing the USE regular clocking scheme.
- The proposed algorithm can handle re-convergence fan-out paths as well as multi-outputs efficiently.
- The automatically generated QCA layouts (XOR, 2:1 MUX) are area efficient when compared with the previous design approaches.

The remaining part of the manuscript is arranged in the following way: Section 2 presents the elementary ideas of QCA technology. Some of the prior work on the QCA cell (P & R) algorithm and regular clocking are presented in Section 3. The suggested automatic cell P & R algorithm followed by logic synthesis and comparison are presented in Section 4. The paper is concluded in Section 5.

2 | QCA BACKGROUND

QCA applications encode and process the information stored in binary mode as charged groups of quantum-points coupled charges [21–23]. A quantum cell is actually four charging dots situated at four corners of a square [24,25], as shown in Figure 1a. The elementary unit of any QCA system is its QCA cell. It can involve a pair of free electrons, which can burrow inside a couple of quantum-dots in the same cell. The two electrons will in general possess the diagonal dots due to Columb's law of electrostatic repulsion. Figure 1b shows the orientation of a QCA cell and its possible two polarizations. Considering the position of pair of charged electron inside the cell, polarizations are differentiated with the following two conditions: (-1) for binary logic '0' and $(+1)$ for binary logic for that of '1' [26,27]. A QCA cell can just reside in either of the two potential states of the cell polarization.

A QCA wire is the amalgamation of few QCA cells positioned in a linear manner. When two or more cells are arranged in a row, then they can relax either in the polarization state, and

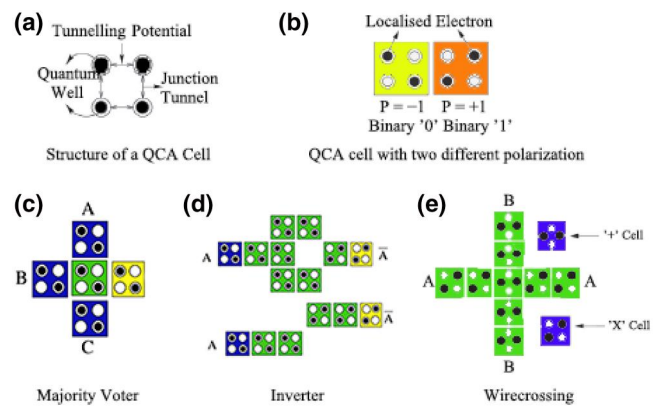


FIGURE 1 (a) A structure of QCA cell (b) Cell polarization (c) Majority voter, (d) Inverter and (e) Wire crossing

this can be explained using the coulombic interaction between the adjacent QCA cells. Two types of wires are generally found in QCA; the one with 90° cells and the other using all 45° cells. The fundamental elements of the QCA logic include a QCA Majority Voter (MV) or Majority Gate (Figure 1c), QCA inverter or NOT gate, shown in Figure 1d and wires. MV gate carries out the logic expression $MV(X, Y, Z) = \text{Maj}(X, Y, Z) = XY + YZ + ZX$, and produces outputs '1' if two or more 1's are present in the input sequence. Here, the symbols X, Y, Z represent the inputs to the MV. Moreover, basic gates like AND gate, OR gate can be realized by one fixed polarized input in the majority gate [28]. The wire crossing can be achieved with different cell types crossed each other, as depicted in Figure 1e.

2.1 | QCA clocking

Unlike the typical CMOS circuit, the clock, a crucial component in QCA circuits is very much fundamental for both the logic circuits that is for combinational as well as sequential. The main benefit of clocking in the QCA circuits lies in the fact that information loss and diminution can be re-established. In QCA, an electric field helps to produce the clocking mechanism [20,29,30].

The switching of QCA arrays is a big concern to design a stable QCA system. The physical ground states provide an advantage to QCA technology [3]. But, it needs to map with the logical solution of the problem for which the device is designed to solve. Any QCA system can be in some excited states (i.e. mixer of polarization $P = -1$ and $P = +1$ in the same QCA system) if the input of the system is switched abruptly. This leads to a meta-stable state, and it could cause delay to arrive at its stable ground state [18,31]. Thus to avoid such problems, adiabatic switching comes into the picture. According to [3], a QCA clock required four phases namely switch, hold, release, and relax for an adiabatic pipe-lining cycle, as shown in Figure 2. In the first phase, the inter-dot barriers are upraised, which polarizes the QCA cell. In the hold

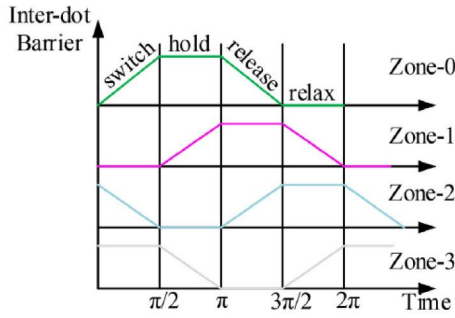


FIGURE 2 QCA clocking

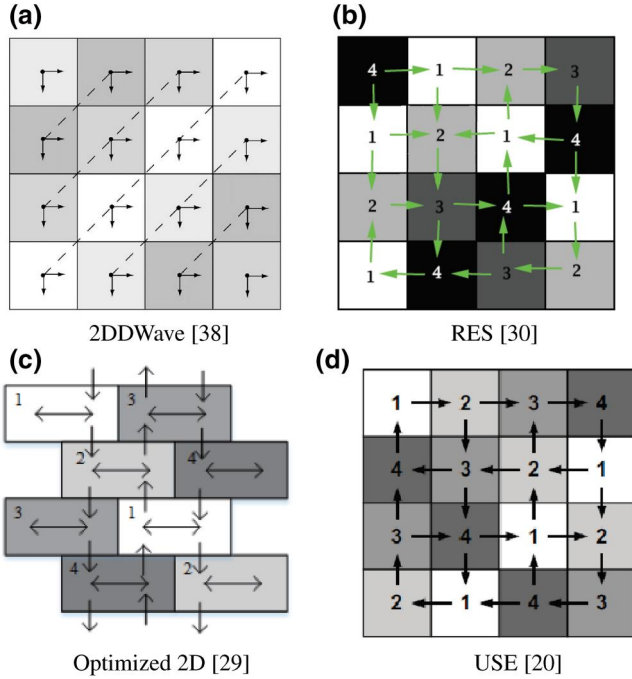


FIGURE 3 Different existing Regular clocking scheme. (a) 2DDWave [38]; (b) RES [30]; (c) Optimized 2D [29]; (d) USE [20]

phase, the inter-dot barriers are so high that the polarization of the QCA cells cannot be changed from an external source; hence it retains its polarization [32].

3 | LITERATURE REVIEW/RELATED WORKS

Some of the recently established works, as reported in [15,33–36] use an arbitrary clocking scheme with irregular size clock zones and unidirectional routing paths. The issues in the realization of feedback/cyclic paths and the fabrication of the circuit are the concern here.

Regular Clocking Scheme: In QCA, few renowned clocking schemes are available, as proposed in [20,29,30,37]. In Two-Dimensional Wave (2DDWave) clocking scheme [37], the flow of information is unidirectional, and due to this implementation of feedback paths are very hard (Figure 3a). In [30], a

Robust, Efficient & Scalable (RES) clocking scheme with three-directional input at one point (Figure 3b) has been introduced. But, an irregular placement of clock zone '3' can be observed in this case. An optimized two-dimensional clocking can be found in [29], claims to reduce underlying metal wire crossings (Figure 3c). However, it doubles the metal wire complexity with the row expansion. Moreover, the tile architecture with a shifted clock zone block intern increases the fabrication overhead of the circuit. On the other hand, a USE [20] realistic clocking system has the uniform, regular and bounded shapes clock zones that provide symmetric bi-directional data flow. The main idea behind this clocking scheme is that adjacent clock zones are placed sequentially one after another to drive correct information from input to output. For example, clock zone 3 always comes after clock zone 2 and before clock zone 4. Clock zone 2 and 4 (or 1 & 3) not to be placed together as these zones do not interact directly, as shown in Figure 3d.

Partitioning and Placement algorithm: In [15], a partitioning and placement algorithm is presented based on zone partition, zone placement, and cell placement. In zone partition, the path with the highest number of gates in a specific zone represents the total delay of the respective zone. Dummy nodes are placed in between re-convergent paths in the same zone to maintain the clocking inconsistency, which leads to a wire-crossing complex. Regarding zone placement, two techniques have been introduced in order to overcome the wire-crossing problem named Simulated Annealing based and Analytical solutions. However, all the experimental results are graph-based, and no layout simulation is provided, thus ends up with an abstract model. Another P&R algorithm is proposed where more focus is given to optimize the cross wire by swapping nodes at the same level in [17]. But, they do not evaluate their proposed algorithm in layout level; hence layout functional correctness and feasibility are missing. In [38], a novel design exploration method for the QCA circuit is proposed considering its design objectives and/or physical constraints of QCA. An automatic QCA Layout Generator (QCA-LG) tool is proposed in [16] and describes how the tool is capable of generating automatic layouts with respect to QCA technology.

Nevertheless, no regular clocking scheme is found to be utilised in the mentioned algorithms, except [18]. Further, the automatically generated QCA layouts are logically tested in QCA Designer. However, it utilizes irregular-size clock zones and unidirectional routing paths. We found the USE clocking scheme to be more suitable and has been utilized for the synthesis of the proposed work. For the realization of larger and complex circuit, the USE clocking scheme can further be extended (8×8) grid using scaling of 4×4 grids as illustrated in Figure 4.

4 | AUTOMATIC CELL PLACEMENT FOR QCA CIRCUIT

This section describes the implementation of systematic path placement algorithm using USE clock which supports bi-directional flow of information in which adjacent clock zones

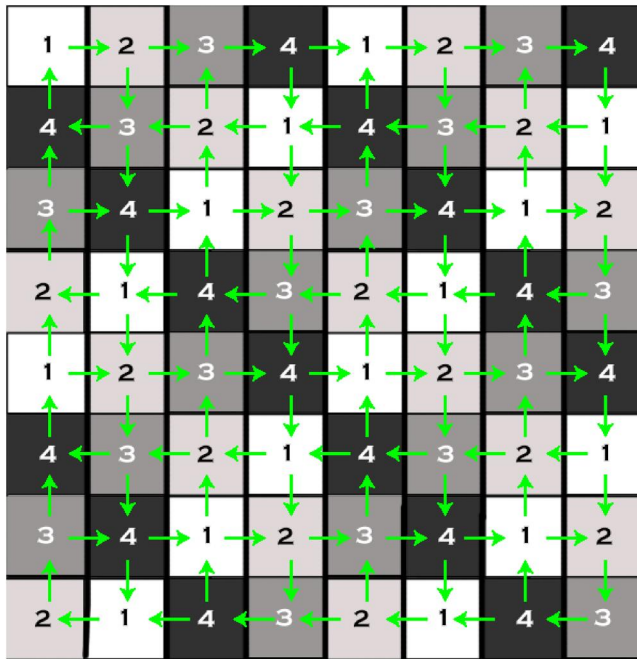


FIGURE 4 Expanded view of USE clocking scheme

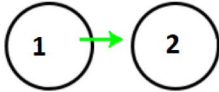

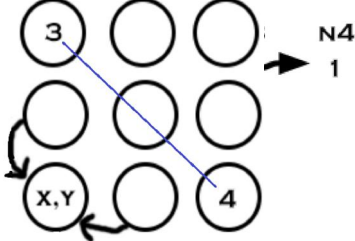
are placed sequentially to drive the information from one end to another. The common notations and the corresponding illustrations are represented in Table 1, for the ease of understanding.

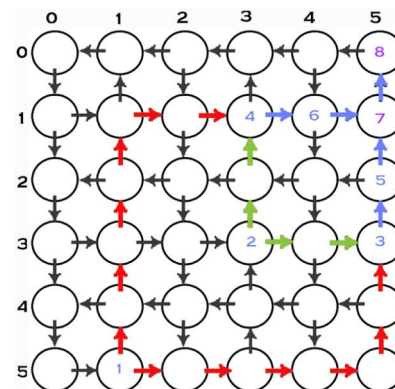
4.1 | Generalized algorithm for cell placement

A directed acyclic graph with necessary dummy nodes is the starting point of the algorithm. The algorithm starts with evaluating all possible paths starting from each input to the output nodes. Number of all possible paths are saved in a variable named *paths* and number of nodes in *nodes*. All possible paths then have been stored in a list named *pathLists*. All these variables (*paths*, *nodes*, *pathLists*) are used in the *defineMatrix()* function which returns a matrix containing information of all paths in a tabular form. Table 2 shows that matrix is returned by the *defineMatrix()* function. Suppose there are five nodes like n_1 , n_2 , n_3 , n_4 , n_5 , and three paths such as p_1 , p_2 , p_3 . These denote that the Path1 is denoted by $n_2 \rightarrow n_4 \rightarrow n_5$, Path2 is reported here is $n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow n_5$ and like-way Path3 is reported by $n_1 \rightarrow n_4 \rightarrow n_5$.

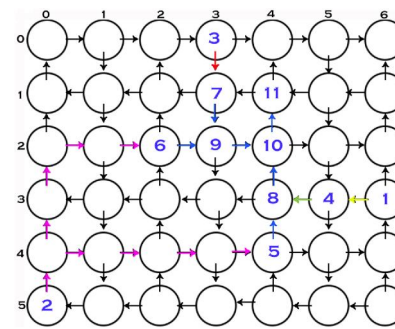
The *defineGrid()* function creates a three-dimensional matrix to place the node and information about available input as well as output directions for data flow of a specific node according to the applied clock (refer to Figure 5). The first two dimensions denote the position of a node placed in the grid. But the third dimension that is a linear array behind each element denotes available data flow directions, which is required to place nodes later on.

TABLE 1 Notation table

Notation	Description
P_1, P_2, \dots, P_n	Denotes the paths
N_1, N_2, \dots, N_m	Represents the nodes
$N_1 \rightarrow N_2 \rightarrow N_3$	Sequence of nodes in a path (starting from node 1 to node 3 via node 2)
	Represents the direction from node 1 to node 2
	Placement of node 4 at input direction to node 5
	Denotes the direction from node 2 to node 4 in path P_1
	Finding equidistant node between node 3 & node 4



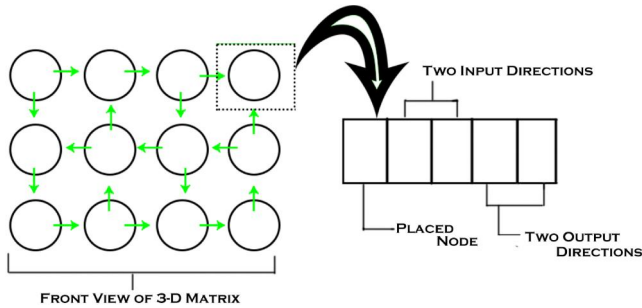
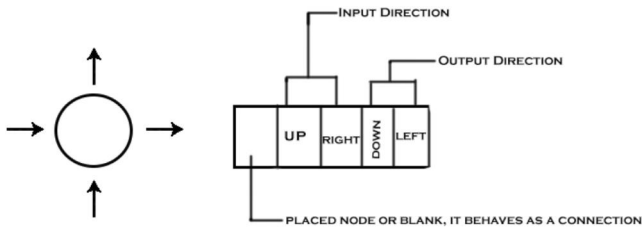
Colour combination while updating the node placement in Ex-OR in the order as: (Green \rightarrow Red \rightarrow Violet). The blue-coloured line represents previously connected nodes.



Colour combination while updating the node placement in MUX: (Green \rightarrow Red \rightarrow Violet \rightarrow Yellow). In this case also, blue-coloured line denotes the previously connected nodes.

TABLE 2 Path-node matrix

Path/Nodes	n1	n2	n3	n4	n5
p1	0	1	0	1	1
p2	1	1	1	0	1
p3	1	0	0	1	1

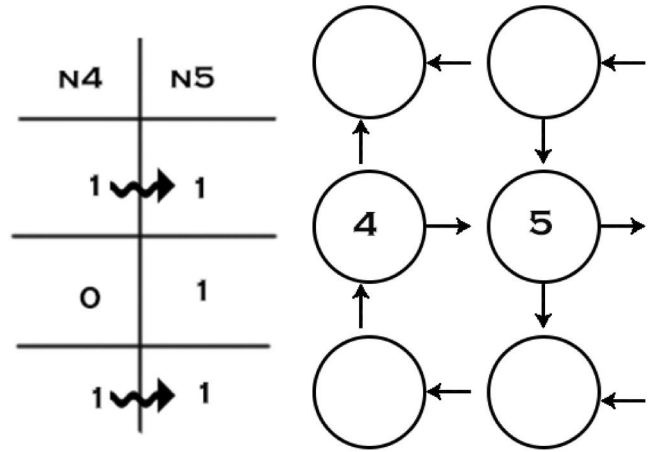
**FIGURE 5** Hypothetical representation of Grid**FIGURE 6** Information stored behind each node

Here, while calling the *defineGrid()* function all directions in all elements have been assigned implicitly regardless that it contains a node or behave as a connection between two nodes. The linear array in each element contains input directions in 2nd and 3rd position and output direction in 4th and 5th, which is shown in Figure 6.

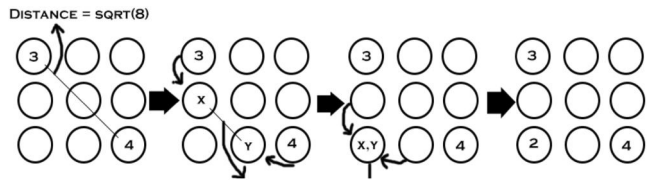
The *cellPlacement()* function then starts from the output node, which is in the rightmost column of the matrix returned by the *defineMatrix()* function. The output node is placed anywhere randomly in the grid. After that, the algorithm comes one by one towards the input node placing each and every node. As shown in the example (Figure 7) n4 column has two 1's, which denote that n4 is present in path p1 and p3 but not in p2. Secondly, both n4 in p1 and p3 points to n5, which leads the algorithm to place n4 in any available input direction just next to n5, (as presented in Figure 7). A similar thing has occurred for n3, which is placed in another input direction of n5.

When the algorithm comes to n2 in the matrix, it finds two 1's but n2 in path p1 points to n4 and p2 points to n3 shown in Figure 8. As such, it is called the *Node()* function to find a node at equal distance from both n3 and n4.

The function by name *equidistantNode()* finds a place at equal distance from two pointed nodes by proceeding

**FIGURE 7** Placing a node giving input to single node

	N2	N3	N4
P1 =>	1	1	1
P2 =>	1	1	0
P3 =>	0	0	1

FIGURE 8 A node pointing to more than one node**FIGURE 9** Placing a node giving input to more than one node

from two pointed nodes in a direction where diagonal distance gradually decreases until both converge in a single node and places n2 in the node where it converges (Figure 9).

The Flow chart of the *cellPlacement()* has been shown diagrammatically in Figure 10, and the supporting functions of the proposed algorithm are explained as:

- The *all_1()* function in the *cellPlacement()* function is a boolean type function that returns True when a column for a specific node in the matrix has all 1's.
- The *samePointed()* function used in *cellPlacement()* is also a boolean type function that returns True only when a specific node gives input to not more than one node.
- The function *shift_input()* returns the coordinate value for the input to the neighbour node coordinate, which will be passed as an argument through the function.
- The *pointedNode()* function returns the node(s) pointed by the current node, that is the node(s) that can be routed from current node.

The generalized algorithm is presented in Algorithm 1.

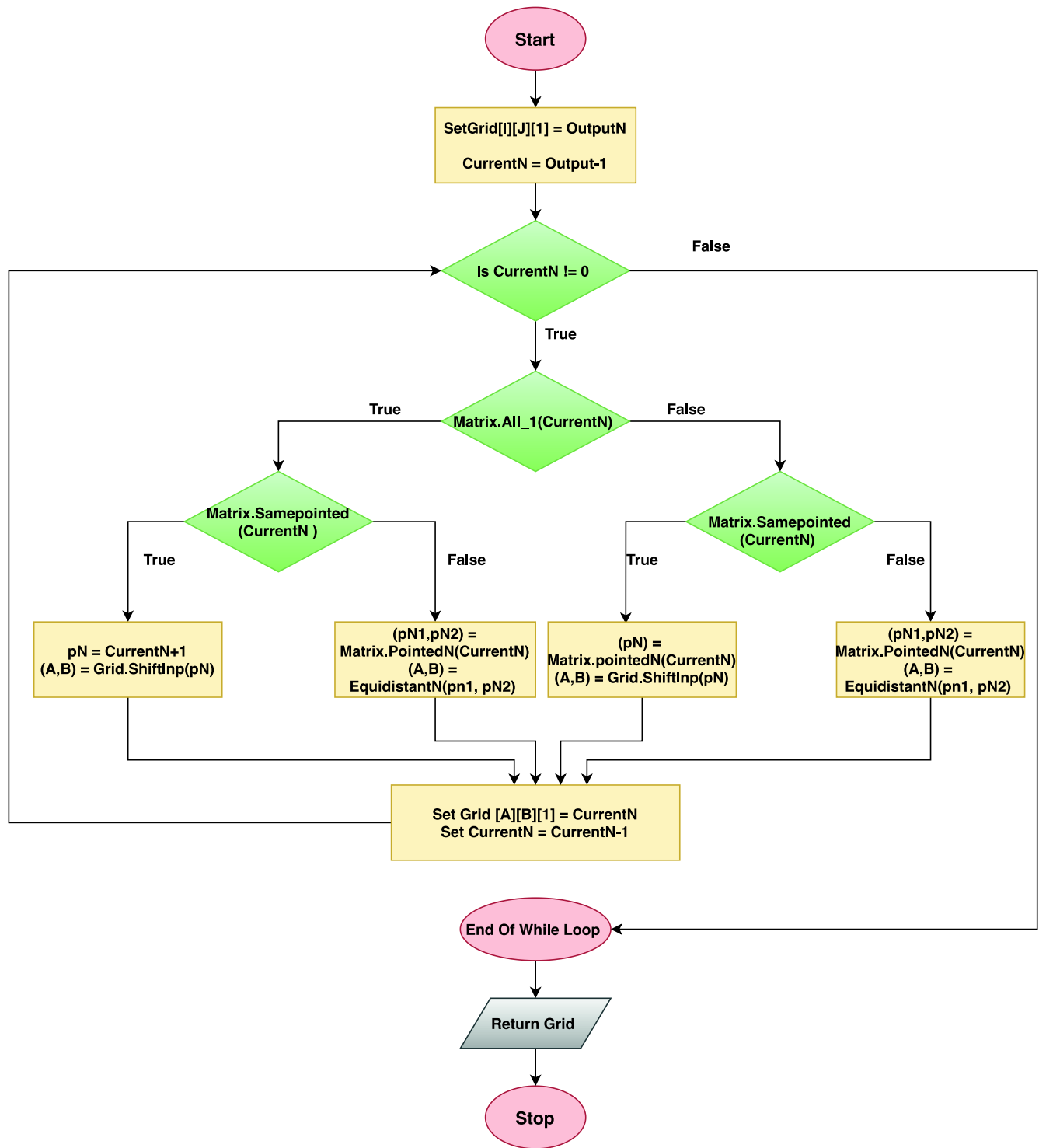


FIGURE 10 Overall synthesis methodology of the proposed work

4.2 | USE clock based logic gate synthesis

4.2.1 | Exclusive-OR

The proposed path based Cell algorithm is applied to the XOR circuit, which is shown in Figure 11. This algorithm starts by evaluating all possible paths from every primary input node to

the output node. In these path representations, the constants represent the corresponding gate numbering in the circuit synthesis, following the possible sequence of information flow (as per Figure 11). Whereas, Path1, Path2, Path3, Path4 represent different possible directions of information from input to output.

Path1 → 1, 3, 5, 7, 8; Path2 → 1, 4, 6, 7, 8

$Path3 \rightarrow 2, 3, 5, 7, 8$; $Path4 \rightarrow 2, 4, 6, 7, 8$

These paths are stored in a list, and then a two-dimensional matrix shown in Table 3 is created to store information about all possible paths. Rows of the matrix denote no. of possible paths, and columns denote each node present in the circuit. If a specific node is present in a specific path, then the element denoting that every node and path in the two-dimensional matrix denoted by 1 as shown in Table 3, for example, node 3 is present in path 1 so the element denoting Node 3 and Path 1 is 1. Node 3 is not included in path 2, so the element has been left the same as zero.

The Cell Placement algorithm starts from the output node in the two-dimensional matrix shown. The grid (i, j) represents QCA layout at USE clock, as shown in Figure 12a where 'i' denotes a specific row in the grid, and 'j' denotes a specific column. For any single output circuit, the output node definitely can be reached from all possible paths. So, the output can be placed in any position of the grid without any difficulty. Now, the algorithm continues in backward direction in the two-dimensional matrix, that is, towards the input nodes until such a column occurs that contains at least one zero. Up to that non-zero column, the nodes are placed on after just considering the USE clock directions starting from the output node in the backward direction. While placing each node in the grid, two parameters of that node must be updated:

1. Its position in the grid that is in the form of (i, j)
2. The direction(s) for its output towards the next node(s)

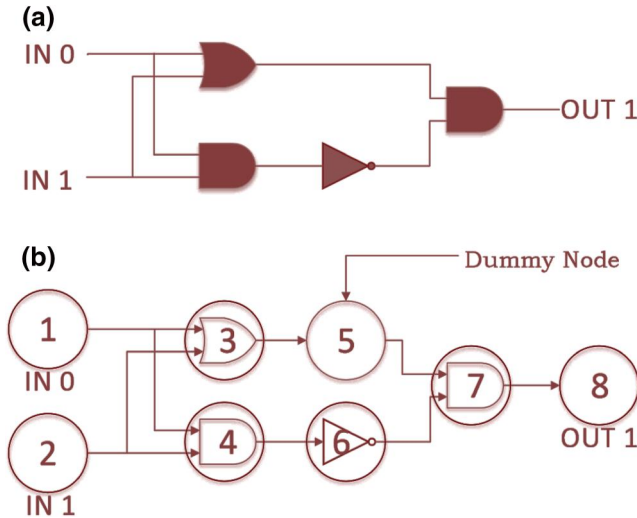


FIGURE 11 (a) XOR Circuit (b) Graph Representation of XOR circuit

TABLE 3 Path-node matrix for XOR

Path/Nodes	n1	n2	n3	n4	n5	n6	n7	n8
p1	1	0	1	0	1	0	1	1
p2	1	0	0	1	0	1	1	1
p3	0	1	1	0	1	0	1	1
p4	0	1	0	1	0	1	1	1

Whenever a zero element column occurs, the previous process is terminated, and the algorithm starts with a new process. The algorithm sorts out all 1's in that column and checks which node all those are pointing to. If all 1's in that row point to the same next node, then this node can easily be placed anywhere in the grid, maintaining the USE clock without any difficulty. If two or more 1's present in the column point for two or more different nodes, then `equidistantNode()` function is called. It determines an equidistant node from that two pointed nodes to place the present node each time calculating the diagonal distance. The colour combination while updating the node placement is maintained as; Green colour arrow placed first followed by Red and so on. Blue colour denoted previously connected nodes (Green \rightarrow Red \rightarrow Violet).

In this specific circuit of the XOR gate, the algorithm starts with an empty grid shown in Figure 12a. Node 8 has been placed in the (0, 5) position in the grid in Figure 12b. Thus, it is checked for Node 7. It is also an all 1 column which can be

Algorithm 1 Algorithm for the proposed work

```

Algorithm Main ()
1  Grid G= defineGrid();
   int paths, nodes;
2  Matrix M= defineMatrixPathL,paths, nodes;
3  G = cellPlacement (M, G);
Procedure defineGrid ()
4  Grid [m][n][5];
   for m = 1 to M do
       Horizontal_out = 'left';
       Vertical_out = 'down';
       for n = 1 to N do
           Grid[m][n][2] = Horizontal_out; Grid[m][n][3] =
               Vertical_out;
           Grid[m][n][4] = ! Horizontal_out; Grid[m][n][5]
               = ! Vertical_out;
           Vertical_out = ! Vertical_out;
       end
       Horizontal_out = ! Horizontal_out;
       Vertical_out = 'down';
   end
5  return Grid;
Procedure cellPlacement (M, G)
   Grid [i][j][1] = outputN;
   nextN = nodes;
   currentN = nodes - 1;
   while currentN != 0 do
       if M.all_1(currentN) == True then
           if M.samePointed(currentN) == True
               then
                   pN = currentN + 1;
                   (a, b) = Grid . shift_input(pN);
                   if M.samePointed(currentN) == False
                       then
                           (pN1, pN2) = M.pointedNode(currentN);
                           (a, b) = equidistantNode(pN1,pN2);
                   if M.all_1(currentN) == False then
                       if M.samePointed(currentN) == True
                           then
                               (pN) = M.pointedNode(currentN);
                               (a, b) = Grid.shift_input(pN);
                       if M.samePointed(currentN) == False
                           then
                               (pN1, pN2) = M.pointedNode(currentN);
                               (a, b) = equidistantNode(pN1,pN2);
                   Grid [a][b][1] = currentN;
                   currentN = currentN - 1;
       end
6  return Grid;

```

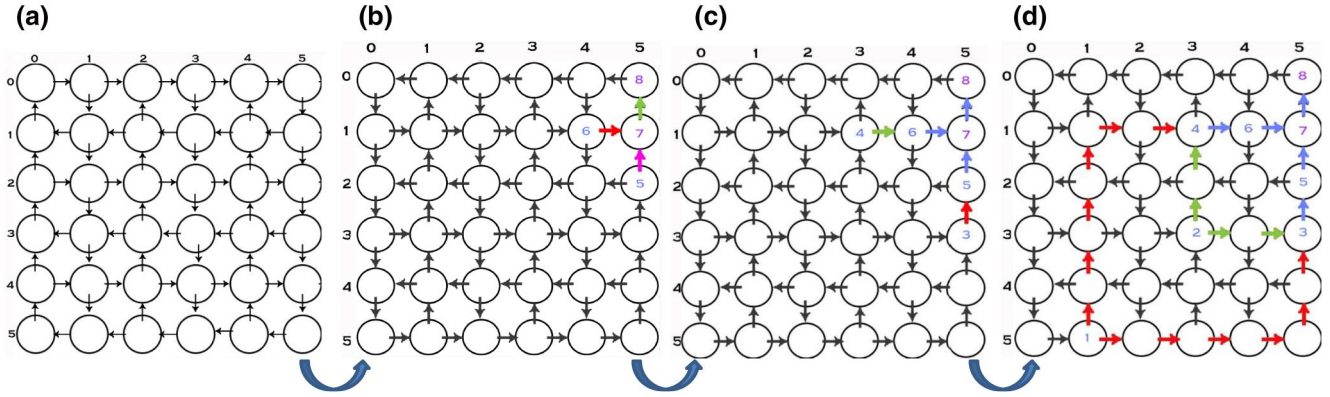



FIGURE 12 (a–d) Node placement for the XOR Graph on USE layout, each node represents a clock zone

Algorithm 2 Supporting Procedures

```

Procedure equidistantNode ((a1, b1), (a2, b2))
    nodeA = (a1, b1);
    nodeB = (a2, b2);
    min_d = diagonalDistance(nodeA, nodeB);
    while nodeA != nodeB do
        A = Grid . shift_input(nodeA);
        B = Grid . shift_input(nodeB);
        if diagonalDistance(A, B) < min_d then
            nodeA = A;
            nodeB = B;
            continue;
        nodeA = newNodeA;
        nodeB = newNodeB;
        (a, b) = index of newNodeA on grid;
    end
    return a, b;

Procedure samePointed (samePointed (currentN))
    List pN;
    pN = Matrix.pointedNode(currentN);
    node = pN[0];
    for i = 1 in pN do
        if node != i then
            return False;
    end
    return True;

Procedure defineMatrix (defineMatrix(PathL, paths,
nodes))
    Int matrix [paths][nodes];
    for i = 1 to paths do
        for j = 1 to nodes do
            Matrix [i][j] = 0;
        end
    end
    for path in PathL do
        for i = 1 to length(path) do
            Matrix [path][node] = 1;
        end
    end
    return matrix;

```

directly connected to Node 8. According to the clock system, Node 8 now has only one input option, so Node 7 has been placed in (1, 5). Now coming to Node 6, there are only two 1's – for Path 2 and Path 4 and both points to Node 7. So Node 6 is connected to Node 7. Node 7 has two directions for receiving the input. Node 6 has been placed in (1, 4) position.

Following the same process, the algorithm place Node 5 in (2, 5) as in Figure 12c. Following the same process Node 4, and Node 3 are placed in (1, 3) and (3, 5), respectively.

Since coming to Node 2, the algorithm gets two 1's, but both are not pointing to the same node. Node 2 in path 3 is pointing to Node 3, and that in Path 4 is pointing to Node 4. In such cases, the algorithm uses specific conditions to find a node from both Node 3 and Node 4 and place that Node 2 there. Node 4 in the grid has two options for taking input from left, and from the bottom, the same applies to Node 3 also. The algorithm calculates the diagonal distance and decides the direction, which leads to the nearest convergence. Here from Node 4, if we go in the bottom and from Node 3 in left, the diagonal distance reduces. Again following the same, it converges to (3, 3) position. Node 2 is placed in (3, 3), shown in Figure 12d. Node 1 has the same property as Node 2, that is, it points to both Node 3 and Node 4 in path 1 and path 2, respectively. So now, the algorithm again starts searching for a Node from Node 3 & 4. Now Node 4 has only left, and Node 3 has bottom available for input. Due to no option, the algorithm goes for this, although it increases the calculated diagonal distance. It comes to (1, 1) and (5, 5) according to the available direction. Then again, calculating the diagonal distance and reducing it, the algorithm converges in the position (5, 1) and place Node 1 at that location. The final diagram depicting the placement of the nodes in the grid has been shown in Figure 12d.

Node placement for the XOR gate on USE regular clocking layout is depicted in Figure 13. Each node represents a clock zone from Figure 12. The QCA simulator for the circuit designed for the XOR gate using the USE clocking scheme, as shown in Figure 14.

4.2.2 | Multiplexer

The proposed automatic cell layout algorithm is applied to the 2:1 MUX circuit, which is shown in Figure 15. The algorithm starts by evaluating all possible paths from each primary input to the output node. The following paths are stored in *pathLists* using which the *defineMatrix()* creates a matrix shown in Table 4. Likewise, XOR, the constants correspond to gate numbering as per

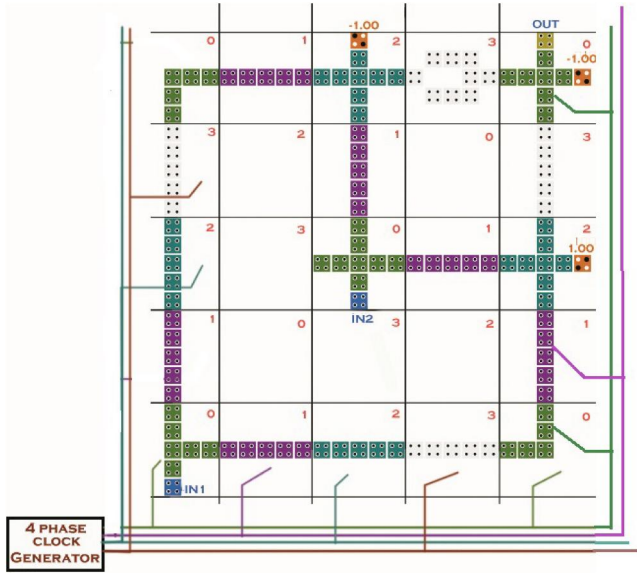


FIGURE 13 Circuit of XOR Gate implemented using USE

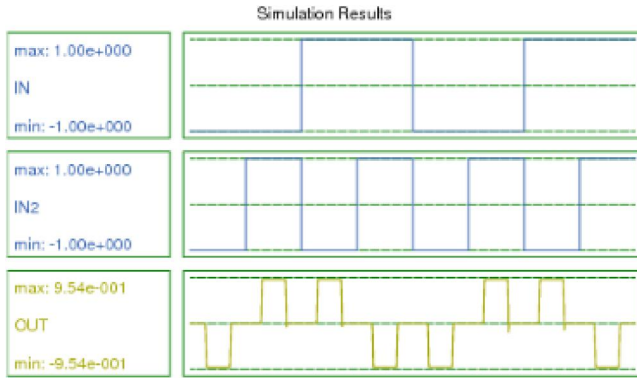


FIGURE 14 Output of XOR Gate using USE as obtained in QCA simulator

Figure 15, and possible information flows are represented by variable Path.

Path1 \rightarrow 1, 4, 8, 10, 11; Path2 \rightarrow 2, 5, 8, 10, 11;

Path3 \rightarrow 2, 6, 9, 10, 11; Path4 \rightarrow 3, 7, 9, 10, 11;

In this specific circuit of MUX, the algorithm starts with an empty grid shown in Figure 16a. Node 11 has been placed at (1, 4), as in Figure 16b. Then Node 10, which is directly giving input to Node 11, is placed at (2, 4) just next to Node 11. Node nine and Node eight each are directly connected to Node 10. So they are placed at (2, 3) and (3, 4), respectively. Node seven and Node six have been placed in two input sides of Node 9, shown in Figure 16c and the same Node 4 and 5 in two input sides of Node 8 shown in Figure 16d. Node 3 is only connected to Node 7, so placed at (0, 3). Node 2 is connected to both Node 5 and Node 6. So the *cellPlacement()* function calls the *equidistantNode()* function to find a place for Node 2. Although (4, 2) is the nearest equidistant node from Node 5 and 6, but due to the design of NOT gate, it does not allow any turn at the

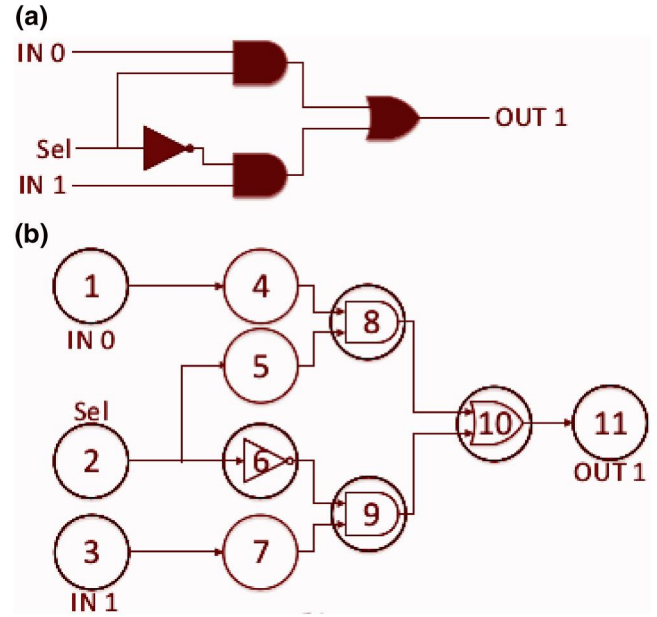


FIGURE 15 (a) 2:1 MUX Circuit (b) Graph Representation of 2:1 MUX circuit

TABLE 4 Path -node matrix for 2:1 MUX

Node	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11
p1	1	0	0	1	0	0	0	1	0	1	1
p2	0	1	0	0	1	0	0	1	0	1	1
p3	0	1	0	0	0	1	0	0	1	1	1
p4	0	0	1	0	0	0	1	0	1	1	1

position of NOT gate. So the function finally converges at (4, 0). Node() function has been terminated placing Node 2 at (4, 0). Now node 1 has been placed just after Node 4 at (3, 6). According to the Clock System, Input Nodes must be in clock zone 0. If Node 1 and 3 were placed at zone 0, Node 2 would be a zone 1. So the position of Node 2 has been shifted to (5, 0) to maintain the clocking rule. The colour combination while updating the node placement is maintained as green colour arrow placed first followed by red and so on. Blue colour denoted previously connected nodes (Green \rightarrow Red \rightarrow Violet \rightarrow Yellow).

Node placement for the 2:1 MUX Graph on USE layout, each node represents a clock zone (Figure 16). The circuit design layout in the QCA simulator following the USE clocking scheme shown in Figure 17 and Figure 18, respectively.

4.3 | Result analysis & discussion

Comparative result analysis of the proposed Exclusive-OR (XOR) and 2-to-1 Multiplexer (MUX) is presented, and the result discussion is made in this section. The systematic, automatic cell placement algorithm, as proposed in this paper, has been applied in both the circuit. Consequently, the USE

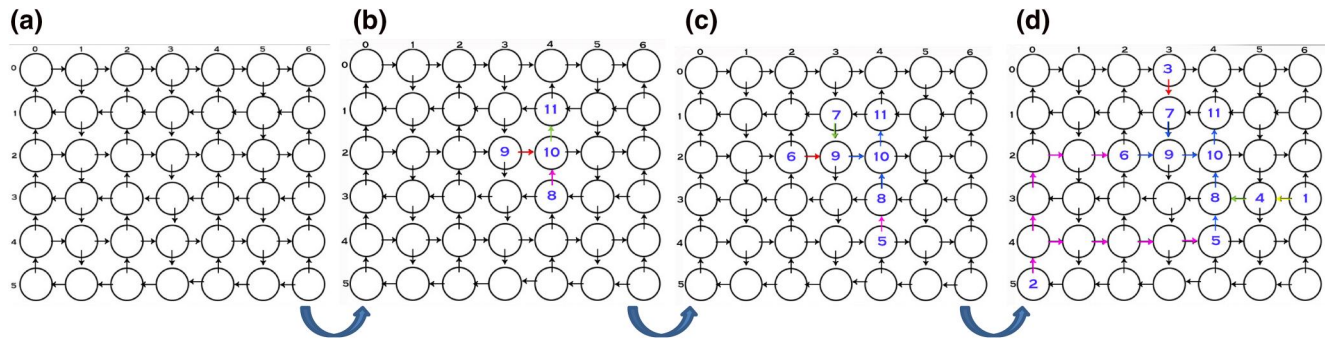


FIGURE 16 (a–d) Node placement for the 2:1 MUX on USE layout

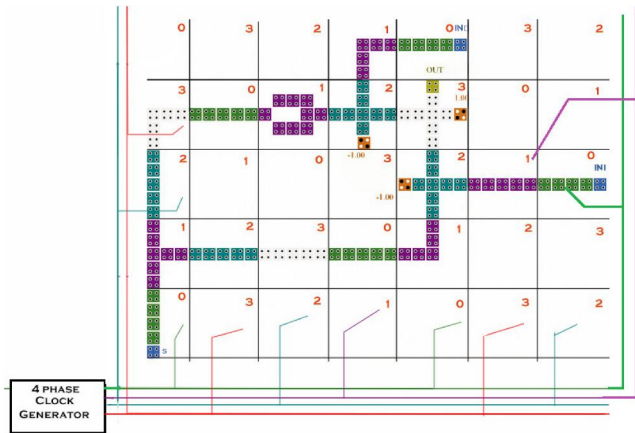


FIGURE 17 Circuit of 2:1 MUX implemented using USE

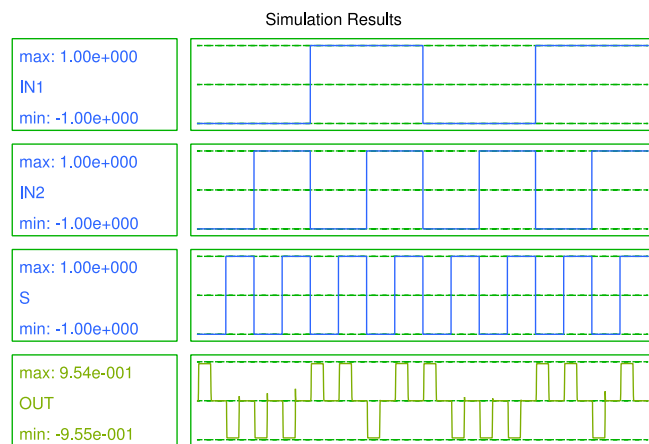


FIGURE 18 Output of the 2:1 MUX as obtained in QCA simulator

regular clocking is also used for the realization of practical implementation. The comparison of the proposed XOR and 2:1 Mux circuit design with the latest or most relevant counterparts are recorded in Tables 5 and 6, respectively. The simulation tool QCADesigner version 2.0.3 has been employed for the implementation of the proposed designs.

It can be observed from the comparative study that the QCA circuit proposed in this paper uses the automatic cell placement technique. Most of the previous designs in the

TABLE 5 Comparison of XOR gate using systematic cell Placement technique

XOR Design	Cell Count	Area (μm^2)	Latency	Auto Cell Placement	Regular Clock
In [39]	8	0.006	2	No	No
In [40]	9	0.01	2	No	No
In [41]	13	0.02	2	No	No
In [42]	9	0.01	1	No	No
In [43]	17	0.015	2	No	No
In [44]	67	0.06	1.25	No	No
In [45]	29	0.03	0.75	No	No
In [46]	54	0.08	1.5	No	No
In [47]	60	0.09	1.5	No	No
Proposed	119	0.26	2.25	Yes	Yes

TABLE 6 Comparison of 2:1 MUX gate using systematic cell Placement technique

MUX Design	Clock Zone	Cell Count	Area (μm^2)	Auto Cell Placement	Regular Clock
In [48]	1	9	0.0076	No	No
In [49]	1	13	0.01	No	No
In [50]	2	17	0.01	No	No
In [37]	8	138	0.44	No	Yes
In [20]	6	84	0.16	No	Yes
In [30]	4	48	0.10	No	Yes
In [16]	11	143	—	Yes	No
Proposed	8	103	0.32	Yes	Yes

literature have focused on low complexity or low device density concerning cell count or area. However, the critical parameter, like regular clocking, is somehow avoided and relies only on arbitrary clocking, which results in a circuit next to unfit for real-time fabrication. Some of the designs have considered regular clocking scheme and successfully simulated the same.

In the case of Exclusive-OR gate design, a significant change in cell count, area, and latency may be noticed.

However, focusing on practical implementation, the one without regular clocking cannot be considered as a proper candidate. Some of them might have ultra-low complex, but they can neither be used in any complex circuit nor can be converted to regular clock based design. Most importantly, none of the paper has used automatic cell placement and routing techniques. The proposed XOR generates a QCA circuit layout from a direct acyclic graph obeying the main QCA concerns and design rules.

Similarly, for the design of multiplexer (2:1 MUX), most of the proposals have come out with the least cell number, lesser area, and the minimum number of clock zone. However, some of the papers have reported the regular clock based design [20,29,30,37]. But it lacks in proposing any automated layout in QCA. The tool known as QCA-LG have proposed in [16], and the same tool has applied in generating automated cell placement for the multiplexer. In this case, the concept of regular clocking has not been considered. The multiplexer design proposed in this paper not only considers the automatic cell placement & routing technique, but at the same time, the use of regular clocking scheme have also been considered. This technique leads to a new paradigm of computing in nanoscale for the real-time circuit design.

5 | CONCLUSION

In this article, a novel cell placement algorithm for the QCA circuit having acyclic paths is proposed which enables the automatic cell layout of the circuit. Most desired regular clocking scheme, like USE, for the QCA logic circuit is embedded in proper order during the cell placement. The successful implementation of the Exclusive-OR gate (EX-OR) and 2:1 multiplexer (MUX) circuit advocates the accuracy of the algorithm. Experimental result outperforms the existing design by properly synthesizing the circuit following an automatic layout and clocking scheme. Unlike existing design, the proposed methodology is fully capable of generating automatic cell layout augmenting the complex clocking circuit under the cell layout driving proper function. The proposed algorithm gives the cost effective design of acyclic circuit in terms of area and cell count. The proposed algorithm is the first one of its kind to realize automatic QCA cell layout incorporating regular clocking scheme.


In near future, the proposed algorithm can be utilized to implement the automatic cell layout for synthesis of sequential logic circuit. It can also be utilized as a tool for automatic cell layout.

ORCID


Dhrubajyoti Bhowmik  <https://orcid.org/0000-0003-3051-8137>

Jayanta Pal  <https://orcid.org/0000-0002-0719-6080>

Mrinal Goswami  <https://orcid.org/0000-0001-5856-6830>

Pinaki Sen  <https://orcid.org/0000-0003-3939-5284>

Apu Kumar Saha  <https://orcid.org/0000-0002-3475-018X>

Bibhash Sen  <https://orcid.org/0000-0003-4803-3074>

REFERENCES

- Qadir, F., Ahmad, P.Z., Wani, S.J., Peer, M.: Quantum-dot cellular automata: theory and application. In: 2013 International Conference on Machine Intelligence and Research Advancement, pp. 540–544. IEEE (2013)
- Sandhu, A., Gupta, S.: A majority gate based ram cell design with least feature size in QCA. *Gazi Univ. J. Sci.* 32(4) (2019)
- Lent C.S., Tougaw P.D. A device architecture for computing with quantum dots. *Proceedings of the IEEE* 85, (4), 541–557 (1997)
- Goswami, M., Sen, D.B., Mukherjee, R., Sikdar, B.: Design of testable adder in quantum-dot cellular automata with fault secure logic. *Microelectron. J.* 60, 1–12 (2017)
- Pal, J., Bhattacharjee, S., Saha, A.K., Dutta, P.: Study on temperature stability and fault tolerance of adder in quantum-dot cellular automata. In: 2019 5th International Conference on Signal Processing, Computing and Control (ISPCC), pp. 69–74. IEEE (2019)
- Fam, S.R., Navimipour, N.J.: Design of a loop-based random access memory based on the nanoscale quantum dot cellular automata. *Photonic. Netw. Commun.* 37(1), 120–130 (2019)
- Chougule, P., Sen, B., Dongale, T.D.: Realization of processing in-memory computing architecture using quantum dot cellular automata. *Microprocess. Microsyst.* 52, 49–58 (2017)
- Song, Z., et al.: An ultra-low cost multilayer ram in quantum-dot cellular automata. In: IEEE Transactions on Circuits and Systems II. Express Briefs (2020)
- Goswami, M., Choudhury, M.R., Sen, B.: A realistic configurable level triggered flip-flop in quantum-dot cellular automata. In: International Symposium on VLSI Design and Test, pp. 455–467. Springer (2019)
- Sen, B., et al.: Design of sequential circuits in multilayer QCA structure. In: 2013 International Symposium on Electronic System Design, pp. 21–25. IEEE (2013)
- Singh, R., Das, S.S., Sarada, V.: Design of a compact negative-edge triggered t flip-flop in QCA technology. *J. Electri. Eng. Technol.* 11(2), 139–146 (2020)
- Ferreira, R., et al.: A run-time graph-based polynomial placement and routing algorithm for virtual fpgas. In: 2013 23rd International Conference on Field programmable Logic and Applications, pp. 1–8. IEEE (2013)
- Formigoni, R.E., et al.: Evaluating nanomagnetic logic circuit layouts using different clock schemes. *Analog. Integr. Circuits Signal Process.* (2020)
- Formigoni, R.E., Ferreira, R.S., Nacif, J.A.M.: Ropper: a placement and routing framework for field-coupled nanotechnologies. In: 2019 32nd Symposium on Integrated Circuits and Systems Design (SBCCI), pp. 1–6. IEEE (2019)
- Lim, S.K., Ravichandran, R., Niemier, M.: Partitioning and placement for buildable QCA circuits. *ACM J. Emerg. Technol. Comput. Syst.* 1(1), 50–72 (2005)
- Teodósio, T., Sousa, L.: QCA-LG: a tool for the automatic layout generation of QCA combinational circuits. In: Norchip 2007, pp. 1–5. IEEE (2007)
- Bubna, M., Roy, S., Shenoy, N., Mazumdar, S.: A layout-aware physical design method for constructing feasible QCA circuits. In: Proceedings of the 18th ACM Great Lakes symposium on VLSI, pp. 243–248 (2008)
- Trindade, A., et al.: A placement and routing algorithm for quantum-dot cellular automata. In: 2016 29th Symposium on Integrated Circuits and Systems Design (SBCCI), pp. 1–6. IEEE (2016)
- Chaudhary, A., et al.: Eliminating wire crossings for molecular quantum-dot cellular automata implementation. In: ICCAD-2005 IEEE/ACM International Conference on Computer-Aided Design, 2005, pp. 565–571. IEEE (2005)
- Campos, C.A.T., et al.: Use: a universal, scalable, and efficient clocking scheme for QCA. *IEEE Trans. Comput. Aided Des. Integrated Circ. Syst.* 35(3), 513–517 (2015)
- Kohavi, Z., Jha, N.K.: In: Switching and Finite Automata Theory. Cambridge University Press (2009)
- Lieberman, M., et al.: Quantum-dot cellular automata at a molecular scale. *Ann. N. Y. Acad. Sci.* 960(1), 225–239 (2002)

23. Zhang, R., et al.: A method of majority logic reduction for quantum cellular automata. *IEEE Trans. Nanotechnol.* 3(4), 443–450 (2004)
24. Rahimi, E., Nejad, S.M.: “A novel architecture for quantum-dot cellular rom,”. In: 2010 7th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP 2010), pp. 347–350. IEEE (2010)
25. Wang, W., Walus, K., Jullien, G.A.: Quantum-dot cellular automata adders. In: 2003 Third IEEE Conference on Nanotechnology IEEE-NANO 2003, 1, 461–464. (2003)
26. Kavitha, S., Kaulgud, N.: Quantum dot cellular automata (QCA) design for the realization of basic logic gates. In: 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), pp. 314–317. IEEE (2017)
27. Nanditha, V., et al.: Design and analysis of digital circuits using quantum cellular automata and verilog. In: 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom), pp. 205–210. IEEE (2020)
28. Pal, J., Dutta, P., Saha, A.K.: Realization of basic gates using universal gates using quantum-dot cellular automata. In: Proceedings of the International Conference on Computing and Communication Systems, pp. 541–549. Springer (2018)
29. Wang, L., et al.: An optimized clocking scheme for nanoscale quantum-dot cellular automata circuit. In: 2019 IEEE 14th International Conference on Nano/Micro Engineered and Molecular systems (NEMS), pp. 336–341. IEEE (2019)
30. Goswami, M., et al.: An efficient clocking scheme for quantum-dot cellular automata. *Int. J. Electron. Lett.*, 1–14 (2019)
31. Welland, M.E., Gimzewski, J.K.: Ultimate limits of fabrication and measurement, vol. 292. Springer Science & Business Media (2012)
32. Sen, B., Sengupta, A., Dalui, M., Sikdar, B.K.: Design of universal logic gate targeting minimum wire-crossings in QCA logic circuit. In: 2010 53rd IEEE International Midwest Symposium on Circuits and Systems, pp. 1181–1184. IEEE (2010)
33. Seyedi, S., Ghanbari, A., Navimipour, N.J.: New design of a 4-bit ripple carry adder on a nano-scale quantum-dot cellular automata. *Moscow Univ. Phys. Bull.* 74(5), 494–501 (2019)
34. Seyedi, S., Navimipour, N.J.: Design and evaluation of a new structure for fault-tolerance full-adder based on quantum-dot cellular automata. *Nano Commun. Netw.* 16, 1–9 (2018)
35. Sherizadeh, R., Navimipour, N.J.: Designing a 2-to-4 decoder on nano-scale based on quantum-dot cellular automata for energy dissipation improving. *Optik.* 158, 477–489 (2018)
36. Singh, G., Raj, B., Sarin, R.K.: Fault-tolerant design and analysis of QCA-based circuits *IET Circ. Dev. Syst.* 12(5), 638–644 (2018)
37. Vankamamidi, V., Ottavi, M., Lombardi, F.: Two-dimensional schemes for clocking/timing of QCA circuits. *IEEE Trans. Comput. Aided Des. Integrated Circ. Syst.* 27(1), 34–44 (2007)
38. Walter, M., et al.: An exact method for design exploration of quantum-dot cellular automata. In: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 503–508. IEEE (2018)
39. Abutaleb, M.: A unique cell-based configuration of XOR gates in quantum-dot cellular automata nanotechnology. In: 2019 IEEE International Conference on Sensors and Nanotechnology, pp. 1–4. IEEE (2019)
40. Gassoumi, I., Toul, L., Ouni, B., Mubaa, A.: An ultra-low power parity generator circuit based on QCA technology. *Electri. Comp. Eng.* 2019 (2019)
41. Xingjun, L., et al.: A new design of QCA-based nanoscale multiplexer and its usage in communications. *Int. J. Commun. Syst.* 33(4), e4254 (2020)
42. Majeed, A.H., et al.: “Full adder circuit design with novel lower complexity XOR gate in QCA technology *Trans. Electron. Mater.* 1–10 (2020)
43. Safoev, N., Jeon, J.-C.: A novel controllable inverter and adder/subtractor in quantum-dot cellular automata using cell interaction based XOR gate. *Microelectron. Eng.* 222, 111–197 (2020)
44. Angizi, S., Alkaldy, E., Bagherzadeh, N., Navi, K.: Novel robust single layer wire crossing approach for exclusive or sum of products logic design with quantum-dot cellular automata. *J. Low Power Electron.* 10(2), 259–271 (2014)
45. Chabi, A.M., et al.: Efficient QCA exclusive-or and multiplexer circuits based on a nanoelectronic-compatible designing approach. *International scholarly research notices.* (2014)
46. Hashemi, S., Farazkish, R., Navi, K.: New quantum dot cellular automata cell arrangements. *J. Comput. Theor. Nanosci.* 10(4), 798–809 (2013)
47. Niemier, M.T.: Designing digital systems in quantum cellular automata. In: Ph.D. Dissertation, University of Notre Dame (2000)
48. Majeed, A.H., et al.: Optimal design of ram cell using novel 2: 1 multiplexer in QCA technology. *Circ. World* (2019)
49. Jeon, J.-C.: Low-complexity QCA universal shift register design using multiplexer and d flip-flop based on electronic correlations. *J. Supercomput.*, 1–15 (2019)
50. Khan, A., Mandal, S.: Robust multiplexer design and analysis using quantum dot cellular automata. *Int. J. Theor. Phys.* 58(3), 719–733 (2019)

How to cite this article: Bhowmik D, Pal J, Goswami M, Sen P, Saha AK, Sen B. Systematic cell placement in quantum-dot cellular automata embedding underlying regular clocking circuit. *IET Circuits Devices Syst.* 2021;15:156–167. <https://doi.org/10.1049/cds2.12015>