Abstract—In order to design large digital circuits using Quantum-Dot Cellular Automata(QCA) cells, CAD tools and automated design methodology for QCA circuits are essential. This paper presents a QCA circuits design methodology based on traditional CMOS circuits design flow. This QCA circuits design methodology utilizes the current CMOS circuits design tools such as HSPICE and Synopsys Design Compiler. QCA cell SPICE model is built for timing check and accurate simulation. The basic QCA layout algorithm is developed for placing the QCA cells in the proper clock zones. Full adder design is shown as an example to demonstrate the whole design process.

I. INTRODUCTION

Semiconductor industry has achieved almost exponential scaling down in feature size for past few decades. However it will be very hard to sustain the trend using conventional lithography based VLSI technology. To replace conventional CMOS technology, broad and extensive researches have been done at nano-scale in recent years. Among the emerging technologies, QCA(Quantum-Dot Cellular Automata) plays an important role not only because it gives a solution at nano scale, but also offers a new methodology of computation and information transformation. In terms of feature size, it is claimed that the size of the basic QCA cell can achieve few nanometers fabricated by molecular implementation at room temperature[1]. A basic QCA cell consists of four quantum dots in a square array coupled by tunnel barriers. Electrons can tunnel between the dots, but can’t leave the cell[2]. There are two excess electrons in a cell, which are compulsively placed on the opposite corners of the cell due to the Coulomb repulsion. The two ground-state polarizations, as shown in Fig 2(a), represent the logic "0" and "1". Based on basic QCA cells, QCA majority gate, inverter, and QCA wire are demonstrated and designed [3] and more complicated circuits such as full-adder, H-memory structure[4], and 12-bit microprocessor[7] are also implemented. In order to build complicate circuits, there is an urgent need for QCA CAD tools. QCADesigner is a QCA layout and simulation tool developed by ATIPS laboratory[9]. Although this software enables users to draw QCA layout with basic cells and do logic simulation, it uses a bottom-up design methodology which is not efficient for large circuits design. The CMOS has been proved to be an effective design methodology, and it is shown in the left part of Fig 1. Since the current QCA-based circuits are mostly digital logic designs, the first two steps of the CMOS design procedure can be easily implanted into QCA design process. However the layout of QCA circuits partly determines the functionality of the QCA circuits, therefore layout of QCA cells and mapping gate-level circuits into QCA cells should be done in the same step. After this step is finished, the QCA cell-level circuits can be verified using SPICE and ready for fabrication. The right part of Fig 1 illustrates the flow of QCA circuits design process.

In this paper, a methodology for QCA circuit design using CMOS-based procedure is proposed. Logic synthesis tools such as Synopsys Design compiler and CMOS timing simulation tools such as HSPICE are used to verify this methodology. SPICE model of QCA cells and basic layout methods of QCA cells are presented.

II. QCA CELL SPICE MODEL

The QCA SPICE model we propose is based on the previous experimental demonstration and verification of a basic QCA cell done by [5] In those experiments, the researchers use the "push-pull" signal to polarize the input dots and then detect the polarization direction of the output dots by measuring the output voltages using voltage meters. On the basis of existing technology, a possible realization of a basic QCA cell would be composed of two series-connected metal dots separated by tunneling barriers and capacitively coupled to a second identical double dots as shown in Fig 2(b)[5]. The left part of a QCA cell and the right part of a QCA cell is exactly
symmetrical; in simplicity, we build the model for half-QCA cell first and connect two half-QCA cells together to get the model of the whole QCA cell. The schematic diagram of a simplified half-QCA cell is shown in Fig 2(c). The two black dots represent two quantum dots and T1 is a tunnel junction. Electrons are able to tunnel between the islands through the tunnel junction T1 but can’t leave these two islands[6]. The movement of the electrons through the tunnel is a birth-and-death process and the tunnel rate is formulated based on the orthodox theory. In a simple case, there is only one extra electron in one pair of quantum dots. Therefore, only two states of the transition exist: one state is the extra electron trapped in island \( i \) and the other state is in island \( j \) holding the extra electron. After some mathematical deductions, the average voltages of island \( i \) and \( j \) are given by

\[
V_i = P_{i=1,j=0}V_{i=1} + P_{i=0,j=1}V_{i=0} \quad (1)
\]

\[
V_j = P_{i=0,j=1}V_{j=1} + P_{i=1,j=0}V_{j=0}, \quad (2)
\]

where \( V_{i=1} \) is the voltage of node \( i \) with the extra electron and \( V_{i=0} \) is the voltage of node \( i \) with no extra electron. \( P_{i=a,j=b} \) represents the probability that node \( i \) holding \( a \) electrons while node \( j \) has \( b \) electrons (\( a, b \) can be either 0 or 1). Therefore, two voltage sources \( V_{S1} = V_{S2} = (V_i - V_j)/2 \) with the internal resistance \( RT1 \) can be used to replace the tunnel junction in SPICE macro model. The proposed SPICE model reflecting these idea is shwon in Fig 3(a). In the figure, \( C_{in} \) is the capacitance between the QCA cells and \( C_i \) is the capacitance between two pairs of metal dots. The node connecting two voltage sources \( V_{S1} \) and \( V_{S2} \) is grounded. Therefore, the voltage of node \( i \) equals to \( V_{S1} \) and the voltage of node \( j \) equals to \(-V_{S2} \). The polarization states of QCA cells corresponding to the different voltages of node \( i \) and \( j \) is shown in Fig 3(b). When the polarization state is ”0”, \( V_i \) is positive and \( V_j \) is negative; while the state is ”1”, \( V_i \) is negative and \( V_j \) is positive. The signs of voltage \( V_i \) and \( V_j \) are always opposite to each other. The voltages of node \( i \) and node \( j \) can change from negative to positive or positive to negative depending on the polarization of QCA cells.

### III. BASIC QCA GATES

With the QCA cell model, it is easy to build QCA basic gate in SPICE by connecting adjacent QCA cells. Unlike CMOS library that has lots of different logic gate implementation, QCA circuits only have three basic components: QCA wire, inverter, and majority voter(MV). MV is a basic logic gate in QCA circuits as shown in Fig 4(a). The polarization states of the cells on the top(cell B), left(cell A), and bottom(cell C) are fixed while the center cell(cell D) is free to react to the fixed charges[3]. When circuit is implemented , the polarization states of cell D’s three neighbors would not be fixed; they would be driven by other QCA cells. The function of MV can be expressed in terms of fundamental Boolean operators as:

\[
M(A, B, C) = AB + BC + AC. \quad (3)
\]

When any one of the three inputs is fixed to one, it performs OR operation; while any one of the three inputs is fixed to zero, it performs AND operation. Therefore, 2-input AND gate and 2-input OR gate are added to QCA library components. MV uses the same number of cells as 2-input AND gate and 2-input OR gate, so it is more efficient to use MV instead of AND/OR gate during the design. However the current logic synthesis tools can’t sufficiently use MV as shown in Table I.

### IV. QCA LAYOUT ALGORITHM

Unlike CMOS digital circuits, even for a small circuit like full adder, we have to set at least four-clock zones to make QCA circuits work properly[7]. There are four clock phases of the QCA circuits as shown in Fig 5 : switch phase, hold phase, release phase, and relax phase. Every QCA circuit can be divided into several clocking zones and all the cells within a clocking zone are in the same clock phase at the same
The names of the circuits & Synthesis Results from Design Compiler

<table>
<thead>
<tr>
<th>Gate Type</th>
<th>2-input AND</th>
<th>2-input OR</th>
<th>INV</th>
<th>MV</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit adder</td>
<td>26</td>
<td>18</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>16-bit adder</td>
<td>98</td>
<td>67</td>
<td>59</td>
<td>9</td>
</tr>
<tr>
<td>32-bit adder</td>
<td>202</td>
<td>140</td>
<td>112</td>
<td>13</td>
</tr>
<tr>
<td>4-bit multiplier</td>
<td>63</td>
<td>45</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>16-bit multiplier</td>
<td>1349</td>
<td>988</td>
<td>566</td>
<td>10</td>
</tr>
<tr>
<td>32-bit multiplier</td>
<td>5815</td>
<td>4384</td>
<td>2445</td>
<td>1</td>
</tr>
</tbody>
</table>

**TABLE I**

**THE SYNTHESIS RESULTS OF ADDERS AND MULTIPLIERS USING QCA CELL LIBRARY**

---

"Signal" passes from one clock zone to a consecutive clock zone with a phase shift of 45 degrees. Consequently, the clock frequency determines the delay of the whole circuit and pipeline can be easily implemented. Setting QCA cells into the proper clock zones is an important topic in QCA cell layout. The two key points in digital circuits design are delay and area. Delay depends on the number of clocking zones for the specific circuits and area not only depends on the number of clocking zones but also the width of each clocking zone (the width of the clocking zone is determined by the number of cells in that clocking zone). In the current QCA technology, a wire channel can take up to 3 cells in width while a majority voter and inverter take 5 cells. However, area optimization is a very complicated problem since in some cases wires are placed in vertical direction while the clock zones are divided in horizontal way as shown in Fig 6(c) (circled area). The number of the wires in vertical direction are hard to estimate. Therefore, the proposed layout algorithm is focused on delay optimization, which means the objective of this algorithm is to place the QCA cells into a minimum number of clock zones. The input file of the algorithm is the gate-level netlist generated by Synopsys design compiler and the output file is formatted gate-level netlist (which includes wire units) compatible with the QCA layout rule, it and can be mapped into SPICE netlist for further simulation. Wire unit is defined as the wire traversing the clock zone as shown in Fig 6(c). Based on gate-level netlist, a Directed Acyclic Graph(DAG) is used to model the circuit. V is the set of graph nodes v. Each v \( \in V \) represents a gate. Source and sink nodes are added to the graph. The source node \( v_0 \) is connected to all primary input pins, like a, b. The sink node \( v_N \) is connected to all primary outputs such as p. E is the set of graph directed edges e. Each edge e \( \in E \) represents the interconnection of the gates. A path p \( \in G \) is a sequence of nodes and edges. The goal of our algorithm is to place the QCA gates in the proper clock zones and add wire units between the QCA gates which are not in the adjacent clock zones. In each clock zone, there is usually only one wire unit or QCA gate in horizontal direction. Therefore, we can assume that each QCA gate or wire unit has one unit delay. Fig 6 shows a conversion example of a small circuit to our graph representation. It can be seen obviously that the path delay from a \( \rightarrow \) p is longer than the delay from b \( \rightarrow \) p. To properly divide the circuit into four clock zones, two wire units will be added to the edge from V0 to V3 as shown in Fig 6(b). This algorithm is similar to the algorithm used to balance the wave-pipelined circuits [8]. The algorithm levelizes a DAG \( G = (V, E) \) according to the data dependencies among nodes. A levelized DAG is denoted as \( G = (V, E, l) \), where each node \( v \in V \) has a level \( l(v) \). The levelizing algorithm is described as following:

Given a graph \( G = (V, E) \), obtain a levelized graph \( G = (V, E, l) \) as follows:

1) Perform a topological sort on \( G \). If there is an edge from node u to node v, then u precedes v in the total order.

2) Traverse the nodes in the order defined by the topological sort. On visiting each node, compute its level \( l(v) \) as follows:

a) If there is no incoming edge, set \( l(v) \leftarrow 0 \).

b) Otherwise, set \( l(v) \leftarrow 1 + \max(l(u))(\text{there is an edge from } u \text{ to } v) \)

---

**Fig. 5.** The four phases of the QCA clock

**Fig. 6.** An example of converting from a circuit to a DAG and QCA cell layout. (a) The gate-level circuit (b) Levelized DAG representation (c) The placement of QCA cells corresponding to the circuit
The result of applying the above algorithm is an input-to-output levelized graph. An equivalent graph (Fig 7(b)) is constructed by inserting wire units along those edges connecting nodes from non-consecutive levels. Given a levelized graph $G$, a homogeneous levelized graph $G_h$ is constructed by adding wire units to $G$, as follows:

1) Let $G'$ be the subgraph of $G$ that contains those edges $u \rightarrow v$ with $l(v) - l(u) > 1$

2) For every edge $e \in G'$, add $P$ wire units to the graph $G$, so that edge $u \rightarrow v$ is transformed to $u \rightarrow v_1 \rightarrow v_2 \ldots v_P \rightarrow v$. Wire unit $v_i$ is levelized to $l(v_i) = l(u) + i$, where $i = 1, \ldots, P$ and $P = l(v) - l(u) - 1$.

After levelizing DAG and adding the wire units, the levelized DAG can easily be mapped into four-clock-zone QCA circuits as shown in Fig 6(c).

V. QCA CIRCUIT DESIGN EXAMPLE

In this section, a QCA full-adder is designed to demonstrate the design methodology we proposed. First, we only have full-adder behavioral model. This behavioral verilog code is used as the input file for Synopsys design compiler and the result is a gate-level netlist including the basic QCA gates such as 2-input AND/OR gate, 3-input MV and inverter. Then this gate-level netlist is converted to DAG representation for further processing. Our layout algorithm is used to construct

the levelized DAG as shown in Fig 7. Each level represents one clock zone. Wire units are added to the edges connecting the gates from non-consecutive clock zones. After that step, the levelized DAG netlist including the wire units is converted to SPICE netlist based on our QCA cell model. Finally, HSPICE is used to verify the function of the full adder. In HSPICE simulation results shown in Fig 8, $V > 0$ means the polarization of the output cell is logic "0" while $V < 0$ means the state of the QCA cell is logic "1". Therefore in this example, the results of sum are in the logic sequence of 01101001 and the results of $C_{out}$ are in the logic sequence of 00010111.

VI. CONCLUSIONS

This paper proposed a design methodology for QCA circuits based on the conventional CMOS design methodology. This methodology fully utilizes the current popular CMOS design tools such as design compiler and HSPICE. The layout algorithm is adopted from the one used to balance the wave-pipelined circuits. The proposed design methodology will speed up and facilitate the design of large QCA-based digital circuits much more. A design example proves the correctness and accuracy of this design methodology. The proposed design flow will be a sound reference and basis for the future QCA based NANO circuit design. As the development and understanding of QCA cell continues, the SPICE QCA model will become more accurate and the layout algorithm will be further optimized in area and delay aspects.

REFERENCES