CHAPTER 5

DESIGN OF COMBINATIONAL LOGIC CIRCUITS IN QCA

5.1 INTRODUCTION

A combinational circuit consists of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs. The combinational logic circuit performs a specific information processing operation fully specified logically by a set of Boolean functions.

The combinational circuit consists of input variables, logic gates and output variables. The logic gates accept signals from the inputs and generate signals to the outputs. This process transforms binary information from the given input data to the required output data. Therefore, both input and output data are represented by binary signals, i.e., they exist in two possible values, one representing logic-0 and the other logic-1.

Any combinational circuit can be designed by the following design procedure:

1. Identify the number of input variables and required output variables.
2. Assign letter symbols to input and output variables.
3. Derive the truth table that defines the required relationship between input and output variables.
4. Obtain the simplified Boolean functions for each output variable by using K-map.
5. Draw the logic diagram for above simplified expression by using logic gates.

Digital computers and calculators consist of arithmetic and logical circuits that add, subtract, multiply and divide binary numbers. The basic combinational circuits are arithmetic circuits. In this thesis, three different single bit arithmetic structures are presented. They are adders, subtractors and multipliers. These structures are popular designs in the transistor technology. The circuit designs in QCA follow the conventional design approaches, but due to the technology differences, they are modified and optimized for the best performance in QCA.

The following design rules are used for QCA implementation of all circuits. A nominal cell size of 20 nm by 20 nm is assumed. The cell has a width and height of 18- and 5-nm-diameter quantum-dots. The cells are placed on a grid with a cell center-to-center distance of 20 nm. Because there are propagation delays between cell-to-cell reactions, there should be a limit on the maximum cell count in a clock zone. This ensures proper propagation and reliable signal transmission.

5.2 DESIGN OF ADDERS USING MG

Digital computers perform various arithmetic operations. The most basic operation is the addition. The addition operation is achieved by majority logic that can reduce the overall number of gates required to create the adder. The first adder is the Half adder (HA) and is designed with 4 majority gates and 2 inverters and the full adder is designed directly and by using half adders. The direct design of full adder is based on QCA addition algorithm and was proposed by Walus et al (2004). The full adder is implemented by two different wire crossings and a serial bit adder is also designed using the full adder. These adders are implemented by QCA cells using the proposed
cell minimization techniques. Hence the proposed adders minimize the area as well as complexity of the circuits. The adders are simulated and results are verified according to the truth table. The performance analyses of those adders are compared with the existing methods.

5.2.1 Design of Half-adder

The half adder is a simple combinational circuit that performs addition of two bits. It is designed conventionally by EXOR and AND gates. When two inputs A and B are added, the Sum and Carry outputs are produced according to the truth table. From the truth table of the half adder as in Table 5.1, one can understand that the Sum output is 1 when either of the inputs (A or B) is 1, and the Carry output is 1 when both inputs (A and B) are 1.

![Schematic diagram of half adder](image)

**Figure 5.1 Schematic diagram of half adder**

**Table 5.1 Truth table of half adder**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The logic function for half adder is,

\[
\begin{align*}
\text{Sum} &= A'B + AB' \quad (5.1) \\
\text{Carry} &= AB \quad (5.2)
\end{align*}
\]

The majority gate expression for above equation is,

\[
\begin{align*}
\text{Sum} &= M (M (A, B', 0), M (A', B, 0), 1) \quad (5.3) \\
\text{Carry} &= M ((A, B, 0) \quad (5.4)
\end{align*}
\]

### 5.2.2 Design of Full-adder

A half adder has only two inputs and there is no provision to add a carry coming from the lower bits when multibit addition is performed. For this purpose, a full adder is designed. A full adder is a combinational circuit that performs the arithmetic sum of three input bits and produces a sum and carry output.

![Figure 5.2 Schematic diagram of Full adder](image)

**Figure 5.2 Schematic diagram of Full adder**

**Table 5.2 Truth table of full adder**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
QCA Addition Algorithm

In the QCA research area, there are two approaches. One is a physical design and the other is an algorithmic design. High level designs focus on the logical and algorithmic design in addition to the physical design. Even though the actual QCA circuit designs need to manage considerable physical interactions which are possibly undesirable and disruptive, the algorithmic approach is also an important aspect in large systems.

Majority Logic of Carry:

\[
\text{Cout} = AB + BC + AC
\]
\[
= M (M (B, M(A,C,1), 0), M(A,C,0), 1)
\]
\[
= M(A,B,C)
\]

Majority Logic of Sum:

\[
\text{Sum} = ABC + A'B'C + A'BC' + AB'C'
\]

One bit full adder has been implemented in two different methods. The first method was conventional (Direct implementation) and consumed a lot of hardware. The second method is (Majority gate reduction) simple and it had less hardware requirement. Here we apply the Majority logic method for constructing QCA adders. The proposed adders are implemented with QCA cells and number of cells has reduced by cell minimization techniques. Hence this implementation further reduces the area and complexity.

The direct implementation of full adder is,

\[
\text{Sum} = M( M(A,M(M(B,C,0), M(B',C',0), 1), 0), 11 \text{ Majority gates}
\]
\[
M (A' M (M (B', C, 0), 1), 0, 1)
\]
The full adder implementation using reduction technique is,

\[
\text{Sum} = ABC'\text{in}+A'B'C'\text{in}+A'B'C'\text{in}+A'B'C'\text{in}
\]

\[
= (A.B + A'.B')C'\text{in} + (A'.B + A.B')C'\text{in}
\]

\[
= [A.B + A'.B' + A.C'\text{in} + A'.C'\text{in} + B.C'\text{in} + B'.C'\text{in}] C'\text{in} +
\]

\[
(A'.B + A.B')C'\text{in}
\]

\[
= [(A'.B' + A'.C'\text{in} + B'.C'\text{in}) + (A.B + A.C'\text{in} + B.C'\text{in})] C'\text{in} +
\]

\[
(A'.B + A.B')C'\text{in}
\]

\[
= [(A'.B' + A'.C'\text{in} + B'.C'\text{in}) + (A.B + A.C'\text{in} + B.C'\text{in})] C'\text{in} +
\]

\[
(A.C'\text{in} + B.C'\text{in}) + (A'.C'\text{in} + B'.C'\text{in})
\]

\[
= [(A'.B' + A'.C'\text{in} + B'.C'\text{in}) C'\text{in} + (A.B + A.C'\text{in} + B.C'\text{in})] C'\text{in}+
\]

\[
(AB + A.C'\text{in} + B.C'\text{in}) (A'.C'\text{in} + B'.C'\text{in} + A'B')
\]

\[
= M(A, B', C'\text{in}) C'\text{in} + M(A, B, C'\text{in}) C'\text{in} + M(A', B', C'\text{in})
\]

\[
M(A, B, C'\text{in})
\]

\[
= M[M(A', B', C'\text{in}), C'\text{in}, M(A, B, C'\text{in})]
\]

Therefore,

\[
\text{Sum} = M[C'\text{out}, C'\text{in}, M(A, B, C'\text{in})] \rightarrow 3 \text{ majority gates} \quad (5.9)
\]

This reduction technique is used to reduce the number of majority gates from 11 to 3.

### 5.3 QCA IMPLEMENTATION OF ADDERS

The adders are first designed by using the majority gate. Then the designed majority gate structures are implemented by QCA cells using cell minimization techniques and simulated by QCADesigner.
5.3.1 Half Adder Implementation

The half adder is designed with 4 majority gates and 2 inverters as shown in Figure 5.3. The QCA implementation of the half adder is shown in Figure 5.4. This is implemented by cell minimization techniques. The half adder layout consists of two, two cells inverter which is indicated by circle. The total number of cells required to implement a half adder is 77, with an area of 83160 nm$^2$ which is much lesser than the previous implementations. The previous implementation has 105 cells with an area of 108000nm$^2$.

![Figure 5.3 Half adder schematic](image1)

![Figure 5.4 Layout of Half adder](image2)

5.3.2 Full Adder Implementation

The full adder is designed directly and by using half adders. The direct design of full adder is based on QCA addition algorithm which was proposed by Zhang et al (2004) and Wang et al (2003). The full adder is designed directly by using reduction technique with 3 majority gates and 2 inverters as shown in Figure 5.5. The full adder is implemented by two different wire crossings such as coplanar and multilayer crossings. These are used for wire crossings.
The coplanar “crossovers” are easier to realize and they can be used with some modification to the basic designs. The QCA implementation of full adder using coplanar crossover is shown in Figure 5.6. The QCA implementation requires 111 cells, with an area of 114300 nm$^2$ and this also required less number of cells than previous implementations. This is achieved by using two cells inverter and the rules for proper alignment of cells. The specific rules in the QCA implementation are (1) the number of cells in the columns need not be equal and (2) the minimum distance between the adjacent rows of cells is width of two cells.

Figure 5.5 Full adder schematic  

Figure 5.6 Layout of Full adder  

The full adder majority gate design in Figure 5.7 is similar to Figure 5.5. But the position of majority gates and inverters are changed for easy implementation of QCA cells. The multilayer crossover design is straightforward. It uses more than one layer of cells like a bridge. The corresponding QCA implementation is shown in Figure 5.8. The QCA implementation requires 98 cells, with an area of 100800 nm$^2$ and this also required less number of cells than previous implementations.
A one bit full adder circuit is constructed by the two half adder circuits and an OR gate. The full adder is designed with 9 majority gates and 4 inverters as shown in Figure 5.9. The corresponding QCA implementation is shown in Figure 5.10. A one bit full adder QCA implementation in Figure 5.9 requires 192 cells, with an area of 208000 nm$^2$. The previous implementation has 218 cells with an area of 286880 nm$^2$. 

**Figure 5.7 Full adder schematic**

**Figure 5.8 Layout of Full adder**

**Figure 5.9 Full adder schematic using half adders**
5.4 SERIAL BIT ADDER (SBA)

The majority gate schematic for a full adder is shown in Figure 5.6. This design can be used to create a bit-serial adder by simply feeding the carry back into the adder (Walus et al 2003). Bits from A and B are entered serially into the circuit, LSB first. The full adder adds the two bits, \( A_i \) and \( B_i \), with the carry \( C_i \) which is saved from the previous bit calculation and produces the partial results \( S_{i+1} \) and \( C_{i+1} \). \( S_{i+1} \) is sent to the output while \( C_{i+1} \) is stored into the Flip Flop (FF) to be used in the next add cycle.
The serial bit adder is designed with 3 majority gates and 2 inverters as shown in Figure 5.11. The corresponding QCA implementation is shown in Figure 5.12. The QCA implementation requires 140 cells, with an area of 147200 nm².

The performance analyses of different types of adders using majority gates are shown in Table 5.3. The performance analyses of those circuits are compared according to the complexity, area, and number of clock cycles and the proposed designs are compared with existing majority gate method.

**Table 5.3 Comparison of Adders using Majority gate**

<table>
<thead>
<tr>
<th>QCA adders</th>
<th>Previous structure</th>
<th>Proposed structure</th>
<th>Number of Clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Complexity</td>
<td>Area</td>
<td>Complexity</td>
</tr>
<tr>
<td>Half adder</td>
<td>105 cells</td>
<td>300 nm x 360 nm</td>
<td>77 cells</td>
</tr>
<tr>
<td>Full adder (Coplanar crossing)</td>
<td>145 cells</td>
<td>439 nm x 367 nm</td>
<td>111 cells</td>
</tr>
<tr>
<td>Full adder (Multilayer crossing)</td>
<td>137 cells</td>
<td>435 nm x 300 nm</td>
<td>98 cells</td>
</tr>
<tr>
<td>Full adder using Half adders</td>
<td>218 cells</td>
<td>652 nm x 440 nm</td>
<td>192 cells</td>
</tr>
<tr>
<td>Bit serial adder</td>
<td>158 cells</td>
<td>500 nm x 382 nm</td>
<td>140 cells</td>
</tr>
</tbody>
</table>

5.5 **SUBTRACTOR DESIGN USING MAJORITY GATE**

A subtractor is a combinational logic circuit. It performs the subtraction operation. The subtraction can be achieved by two different methods. First method, the subtraction of two binary numbers may be accomplished by taking the complement of the subtrahend and adding it to the minuend. By this method, the subtraction operation becomes an addition
operation requiring full adders for its implementation. In second method, the subtraction is done directly, as done with paper and pencil. By this method, each subtrahend bit of the number is subtracted from its corresponding significant minuend bit to form a difference bit. If the minuend bit is smaller than the subtrahend bit, a 1 is borrowed from the next significant position. The fact that a 1 has been borrowed must be conveyed to the next higher pair of bits by means of a binary signal coming out of the given stage and going into the next higher stage. In the QCA technology the subtraction operation is achieved by majority logic using reduction technique that can reduce the overall number of gates required to create the subtractor.

5.5.1 Design Half Subtractor

A Half-subtractor subtracts two bits and produces their difference. It also has an output to specify if a 1 has been borrowed. The half subtractor needs two inputs: minuend (A) and subtrahend (B) and two outputs borrow and difference.

![Half Subtractor Diagram](image)

Figure 5.13 Schematic diagram of half subtractor

Table 5.4 Truth table of half subtractor

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The logic function for half-subtractor is,

\[ \text{Difference } D = A'B + AB' \] \hspace{1cm} (5.10)

\[ \text{Borrow } B_0 = A'B \] \hspace{1cm} (5.11)

The majority gate expression for above equation is,

\[ D = M (M (A, B', 0), M (A', B, 0), 1) \] \hspace{1cm} (5.12)

\[ B_0 = M ((A', B, 0)) \] \hspace{1cm} (5.13)

### 5.5.2 Design of Full Subtractor

A full subtractor circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage. This circuit has 3 inputs and two outputs. The three inputs A, B, and C denotes the minuend, subtrahend and previous borrow, respectively. The two outputs D and B0 represent the difference and output barrow respectively.

![Schematic diagram of full subtractor](image)

**Figure 5.14 Schematic diagram of full subtractor**

**Table 5.5 Truth table of full subtractor**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The logic function for full-subtractor is,

\[
\text{Difference } D = ABC + A'B'C + A'BC' + AB'C' \quad (5.14)
\]

\[
\text{Borrow } B_0 = A'C + A'B + BC \quad (5.15)
\]

The full-subtractor is designed based on QCA addition algorithm given in and (Zhang et al 2004). The majority gate expression for borrow equation is,

\[
B_0 = M(M(B, M(A', C, 0), 0), M(A, C, 0), 1)
\]

Therefore, \( B_0 = M((A', B, C) \) \quad (5.16)\)

The majority gate expression for difference equation is,

**Direct Implementation:**

\[
D = M(M(A, M(M(B, C, 0), M(B', C', 0), 1, 0),
M(A', M(M(B', C, 0), 1), 0), 1)
\]

This direct implementation requires 11 numbers of gates. But the proposed reduction algorithm further reduces the number of gates which in turn reduces the area and complexity.

**Reduction Technique**

\[
D = ABC + A'B'C + A'BC' + AB'C'
\]

\[
= (A.B + A'.B') C + (A'.B + A.B') C'
\]

\[
= [A.B + A'.B' + A.C ' + A '.C ' + B.C' + B ' C '] C + (A '.B + A.B ') C'
\]

\[
= [(A '.B ' + A '.C ' + B '.C ') + (A.B + A C ' + B C ')](A '.B + A.B ')C ' in
\]

\[
= [(A '.B ' + A '.C ' in + B '.C ' in) + (A.B + A C ' in + B C ')] C + (A.C ' + B.C ') + (A '.C + B'.C ')
\]
\[
[(A \cdot \bar{B} + A \cdot \bar{C} + B \cdot \bar{C}) \cdot C + (A \cdot B + A \cdot C + B \cdot C \cdot \bar{C})] \cdot \bar{C} + (A \cdot B + A \cdot C + B \cdot C \cdot \bar{C}) \\
= M (A, B, C) \cdot \bar{C} + M (A, B, C) \cdot \bar{C} + M (A, B, C) \cdot \bar{C}
\]

\[D = M [B' \text{ out}, C, M (A, B, C')]\]

(5.18)

### 5.6 QCA IMPLEMENTATION OF SUBTRACTORS

#### 5.6.1 Half Subtractor

The half subtractor is designed with 4 majority gates and 3 inverters as shown in Figure 5.15. The corresponding QCA implementation is shown in Figure 5.16. In our implementation the total number of cells required is 77 cells, with an area 83160 nm². This is much lesser than the previous QCA implementation. This is achieved by cell minimization techniques. The previous implementation has 112 cells with an area of 108000 nm². Similarly the full subtractor circuit is designed and implemented using QCA cells as shown in Figure 5.17 and Figure 5.18.
5.6.2 Full Subtractor

One bit full subtractor has been implemented in two different methods. The first method was conventional (Direct implementation) and consumed a lot of hardware. The second method is (Majority gate reduction) simple and had less hardware requirement. Here the Majority logic method is applied for constructing QCA subtractor. This implementation has reduced number of cells. The number of cells has reduced by suitable arrangement of cells without overlapping of neighbouring cells and especially 2cell inverter is used instead of 7cell inverter. Hence our implementation further reduces the area and complexity.

The full subtractor is designed with 3 majority gates and 2 inverters as shown in Figure 5.17. The corresponding QCA implementation is shown in Figure 5.18. In the QCA implementation, coplanar crossings are used for interconnections. The QCA implementation requires 114 cells, with an area of 208000 nm².

![Figure 5.17 Full subtractor schematic](image1)

![Figure 5.18 Layout of Full subtractor](image2)
The full subtractor design in Figure 5.19 is similar to Figure 5.17. But the position of majority gates and inverters are changed for easy implementation of QCA cells. The subtractor is implemented by QCA cells using multilayer crossovers as shown in Figure 5.20. The QCA implementation requires 98 cells, with an area of 100800 nm$^2$.

Figure 5.19 Full subtractor schematic  Figure 5.20 Layout of Full subtractor

A one bit full subtractor circuit is constructed by the two half subtractors and an AND gate. The full adder is designed with 9 majority gates and 6 inverters as shown in Figure 5.21. The corresponding QCA implementations are shown in Figure 5.22. In this implementation the total number of cells required is 192 cells, with an area 208000nm$^2$.

Figure 5.21 Full subtractor schematic using half subtractors
The performance analyses of different types of subtractors using majority gates are shown in Table 5.6. The performance analyses of those circuits are compared according to the complexity, area, and number of clock cycles.

Table 5.6 Comparison of Subtractors using Majority gate

<table>
<thead>
<tr>
<th>Subtractors</th>
<th>Proposed structure</th>
<th>Number of Clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Complexity</td>
<td>Area</td>
</tr>
<tr>
<td>Half subtractor</td>
<td>77 cells</td>
<td>297nm x 280nm</td>
</tr>
<tr>
<td>Full subtractor (Coplanar crossover)</td>
<td>114 cells</td>
<td>381nm x 300nm</td>
</tr>
<tr>
<td>Full subtractor (Multilayer crossover)</td>
<td>98 cells</td>
<td>360nm x 280nm</td>
</tr>
<tr>
<td>Full subtractor using HS</td>
<td>188 cells</td>
<td>650nm x 320nm</td>
</tr>
</tbody>
</table>

5.7 MULTIPLIER DESIGN USING MAJORITY GATE

Multiplication of binary numbers is done by successive additions and shifting. The multiplication is defined by the following procedure.
Procedure

1. Multiplicand is multiplied by each bit of the multiplier starting from the LSB
2. Each multiplication forms a partial product.
3. Successive partial products are shifted one position to the left.
4. The final product is obtained from the sum of the partial products.

Consider the multiplication of two-2-bit numbers as shown in Figure 5.23. The multiplicand bits are \(B_1\) and \(B_0\), the multiplier bits are \(A_1\) and \(A_0\) and the product is \(M_4\), \(M_3\), \(M_2\), \(M_1\). The multiplication of two bits such as \(A_0\) and \(B_0\) produces a 1 both bits are 1; otherwise it produces 0. This is identical to \(A_1\) by \(B_1\) \(B_0\) and is shifted one position to the left. The partial products are added by using full adders and produce the sum of partial products. Note that the product obtained from the multiplication of two binary numbers of \(n\) bits each can be up to \(2n\) bits long.

![Figure 5.23 Bit product matrix for 2 bit multiplication](image)

This thesis describes the design of bit-serial multiplier and carry delay multiplier circuits using the adders created in the section 5.4 and 5.3. The design is based on a multiplier schematic created for conventional FET based logic circuits. With this one of the inputs is broadcast across the adder serially and other is loaded in parallel. The partial products are calculated and immediately added to the sum. The multiplier is implemented by QCA cells.
using the cell minimization techniques. The multipliers can easily be scaled by adding more full adder blocks and partial product generators in a continuous array. The multipliers can be extended to 32 bit or more. The size of the multiplier grows linearly with the number of bits, making it efficient in area. Walus and Jullien (2003) have done research on the design of multipliers.

5.7.1 Bit Serial Multiplier

With this design one of the inputs is broadcast across the adder serially and the other is loaded in parallel. The partial products are calculated and immediately added to the sum. The schematic for this multiplier is shown in Figure 5.23. The D-latches in this schematic are required for the proper operation of the device. In order to transfer this design into a QCA circuit, many more D-latches are introduced from the very nature of a QCA circuit. A schematic for a QCA multiplier is generated and the required D latches introduced appropriately to maintain proper circuit operation. The delay for this 2-bit multiplier is 3 clock cycles, where a clock cycle is represented by the four clocking zones.

The bit-serial/parallel multiplier can easily be scaled by adding more full-adder blocks and partial product generators in a continuous array. The size of the multiplier grows linearly with the number of bits, making it efficient in area. As the size of the multiplier is increased, the delay from input to output also increases linearly.

The QCA layout for the 2-bit multiplier is shown in Figure 5.25. The schematic is drawn to match the layout as much as possible. Although there is a delay between the signal entering the multiplier and the first bit at the output, there is no latency introduced between the output bits. As well, because of the pipelined nature of QCA, a new multiplication can start before the previous one is completed.
Figure 5.24 Schematic of Bit serial multiplier

The bit serial multiplier circuit is constructed by the two full adders and two AND gates. The full adder is designed with 3 majority gates and 2 inverters as shown in Figure 5.7. Therefore a bit serial multiplier is designed with totally 8 majority gates and 4 inverters. The corresponding QCA implementations are shown in Figure 5.24.

Figure 5.25 QCA implementation of Bit serial multiplier.

The QCA implementation requires 321 cells, with an area of 369600nm$^2$. The number of cells required to implement the multiplier is less which is achieved by cell minimization techniques. The specific rules used in
this QCA implementation are similar to rules used in the full adder. Here the coplanar wire crossings are used to interconnect the devices.

5.7.2 Carry Delay Multiplier

Bit-serial adders are used to realize the carry delay multiplier. The bit-serial adder is modified from the full adder so that the carry-in and carry-out are connected internally with one clock delay. The general schematic diagram for the carry delay multiplier as shown in Figure 5.26. Heumpil Cho and Earl E. Swartzlander (2007) have worked on the multiplier design. Using these adders, a 2-bit CDM multiplier according to Figure 5.8 is implemented as shown in Figure 5.26. Multipliers for larger word sizes can be implemented easily by adding additional bit slices. For N-bit inputs, the multiplier receives N+1 inputs (a serial input and N parallel inputs) and produces a serial output. The serial input and output are ordered from LSB to MSB and parallel inputs are repeated whenever a new serial input is provided (N cycles). For initialization of the multiplier, zero bits are input for N clock cycles. Zero bits are provided between successive inputs. The time to complete an N-bit multiplication is 2N cycles.

![Figure 5.26 Schematic of carry delay multiplier](image)

The carry delay multiplier circuit is constructed by the two full adders and two AND gates and the each full adder is designed with 3 majority gates and 2 inverters as shown Figure 5.7. Therefore a carry delay multiplier
is designed with totally 8 majority gates and 4 inverters. The corresponding QCA implementations are shown in Figure 5.27.

![QCA implementation of carry delay multiplier](image)

Figure 5.27 QCA implementation of carry delay multiplier

The QCA implementation requires 177 cells, with an area of 192,600nm$^2$. The number of cells required to implement the multiplier is less which is achieved by cell minimization techniques. The specific rules used in this QCA implementation are similar to rules used in the full adder. Here the multilayer crossings are used to interconnect the devices.

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>Complexity</th>
<th>Area</th>
<th>Number of clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit-serial multiplier</td>
<td>321 cells</td>
<td>660nm x 560nm</td>
<td>2</td>
</tr>
<tr>
<td>Coplanar crossover</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carry delay multiplier</td>
<td>177 cells</td>
<td>535nm x 360nm</td>
<td>2</td>
</tr>
<tr>
<td>Multilayer crossover</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.7 Comparison of Multipliers

5.8 RESULT AND DISCUSSION

The QCA concept is generic in that there may be several different implementation possibilities. At the core of the concept is a bi-stable cell,
which must interact locally with its neighbours such that information processing can be performed as described previously.

The simulated waveform of half adder is shown in Figure 5.28. The circuit has four clocking zones. Initially clock 0 is used to get the inputs A and B. Clock 1 is used to route inputs for majority gate logic, clock 2 is used for finding majority logic and clock 3 is used to compute output. The output is available at clock 0 again. Clock 1 to 3 considered here is a sequence of setup for hold, relax and release phase, to control the flow of information in QCA circuits. Similar to half adder, the full adder and bit serial adder also required 4 clock zones to produce required output. The simulated results of full adder and serial bit adder are shown in Figure 5.29 and Figure 5.30.

The simulated waveforms of half subtractor and full subtractor are shown in Figure 5.31 and Figure 5.32. Similar to adder, the subtractors also required 4 clock zones to produce required output. The simulated waveform of multiplier is shown in Figure 5.33. The delay for this 2-bit multiplier is 3 clock cycles, where a clock cycle is represented by the four clocking zones. Hence 12 clock zones are required to produce the output.
Figure 5.29 Simulation result of full adder

Figure 5.30 Simulation result of bit serial adder

Figure 5.31 Result of half subtractor
5.9 CONCLUSION

In this chapter the different arithmetic circuits have been designed using majority gates. The layouts and functionality checks were done using QCADesigner and the designs are compared according to the complexity, area and number of clock cycles. The operations of these circuits have been verified according to the truth table. The performance analyses of those circuits are compared. The proposed layouts are significantly smaller than the circuits using CMOS technology and it reduces the area as well as complexity required for the circuit than the previous QCA circuits.