A STUDY ON A QUANTUM-DOT CELLULAR AUTOMATA BASED ASYNCHRONOUS CIRCUIT DESIGN

By

MYUNGSU CHOI

Bachelor of Science
Oklahoma State University
Stillwater, Oklahoma
2003

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2005

A STUDY ON A QUANTUM-DOT CELLULAR AUTOMATA BASED ASYNCHRONOUS CIRCUIT DESIGN

Thesis Approved:

Nohpill Park

Thesis Advisor

K.M.George

Vanketesh Sarangan

A. Gordon Emslie

Dean of the Graduate College

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The CMOS (Complementary Metal Oxide Semiconductor) technology has allowed for a dramatic development of microelectronics since the 1960s. Even though the reduction of physical size in electronic devices provides for a much higher speed and density for the systems, it brings some serious related problems such as extremely high power consumption, heat dissipation and unreliability. According to some studies, CMOS technology may hit the physical limitation around middle of 2010s. The QCA (Quantum-Dot Cellular Automata) has been introduced as one of the six alternative technologies [1]. It not only provides nanoscale system implementation, but also provides a new way of computation and data transformation [2, 6, 7, 18, 19]. In the QCA paradigm, a regular array of cells each interacting with its neighboring cells, is employed in a locally interconnected architecture [8, 9]. The coupling between the cells occurs due to their electrostatic interactions. Such arrays are, principally, capable of encoding digital information. The fundamental unit of QCA is the QCA cell created with four quantum-dots. A QCA cell is a nano-scale device which can store logic states and transfer information using Coulomb interaction. The conventional CMOS digital logic holds its logic state due to the voltage level, but QCA holds its logic state due to the position of electrons. A QCA cell consists of a square with four dots at each vertex and two electrons. Due to the Coulomb repulsion, these electrons can only be in the diagonal vertex. So, these two different polarization states are associated with binary 0 and 1. Figure 1.1 shows two possible polarizations of QCA cells. Proper allocation of the number of QCA cells is the first step in building logic devices. The most popular QCA devices such as Majority Voter gate, Inverter gate, and even binary wires consist of the number of QCA cells only.

Clocking is important in most computational technologies and a requirement for the

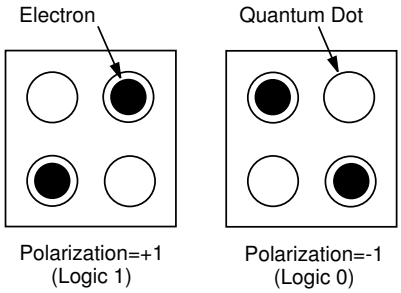


Figure 1.1: The fundamental unit of QCA is the QCA cell created with four quantum-dots positioned at the vertices of a square. The cell is loaded with two extra electrons which tend to occupy the diagonals of the cell. Binary information is encoded in the two possible polarizations. The cell will switch from one polarization to the other when the electrons quantum mechanically tunnel from one set of dots to the other.

synchronization of information flow in QCA. Presently, all QCA circuit proposals require a clock not only to synchronize and control information flow, but also to actually provide the power to run the circuit [12, 3]. Apart from the clock, the QCA cells are not powered by any other external source. Therefore, it is difficult to imagine a QCA circuit that can avoid using a clock.

As the concept of QCA clocking, the 4-phase clocking has been proposed and used in [4, 5]. The 4-phase clocking signal is applied to four adjacent buried wires. Each wire has voltage that raises and lowers linearly in order to switch the QCA cells placed above it. The 4-phases are: ”Switch”, ”Hold”, ”Release”, and ”Relax.” Adjacent wires have a $\pi/2$ phase shift so that every fourth wire has an identical signal. With CMOS clocking, clock high means logic 1 and clock low means logic 0. But with 4-phase clocking, clock high means ”DATA” (either logic 0 or 1) and clock low means ”NULL” (empty data). Basically, QCA cells in the ”Hold” phase of 4-phase clocking retain transmitted data, but they lose the data in the ”Relax” phase.

The differences between QCA and CMOS clocking, such as the way to hold data, the way to synchronize data flows, and the way to power QCA cells makes the design of a QCA circuit quite different from that of a VLSI. Due to the 4-phase clocking, overall timing of a QCA circuit is mainly dependent on its layout. This fact is commonly refereed to as the ”Layout=Timing” Problem [16]. One of most severe limitations of ”Layout=Timing” is

that of wasted buffers to synchronize QCA circuit. For example, as shown in Figure 1.2 that is a QCA circuit for a simple logic function F = $\bar{A}$B, consists of one QCA inverter gate, one MV gate and four QCA wires. The resulting waveform in Figure 1.2 is incorrect, because the QCA wire for input B is in the clock zone 0, and MV gate is in the clock zone 2.

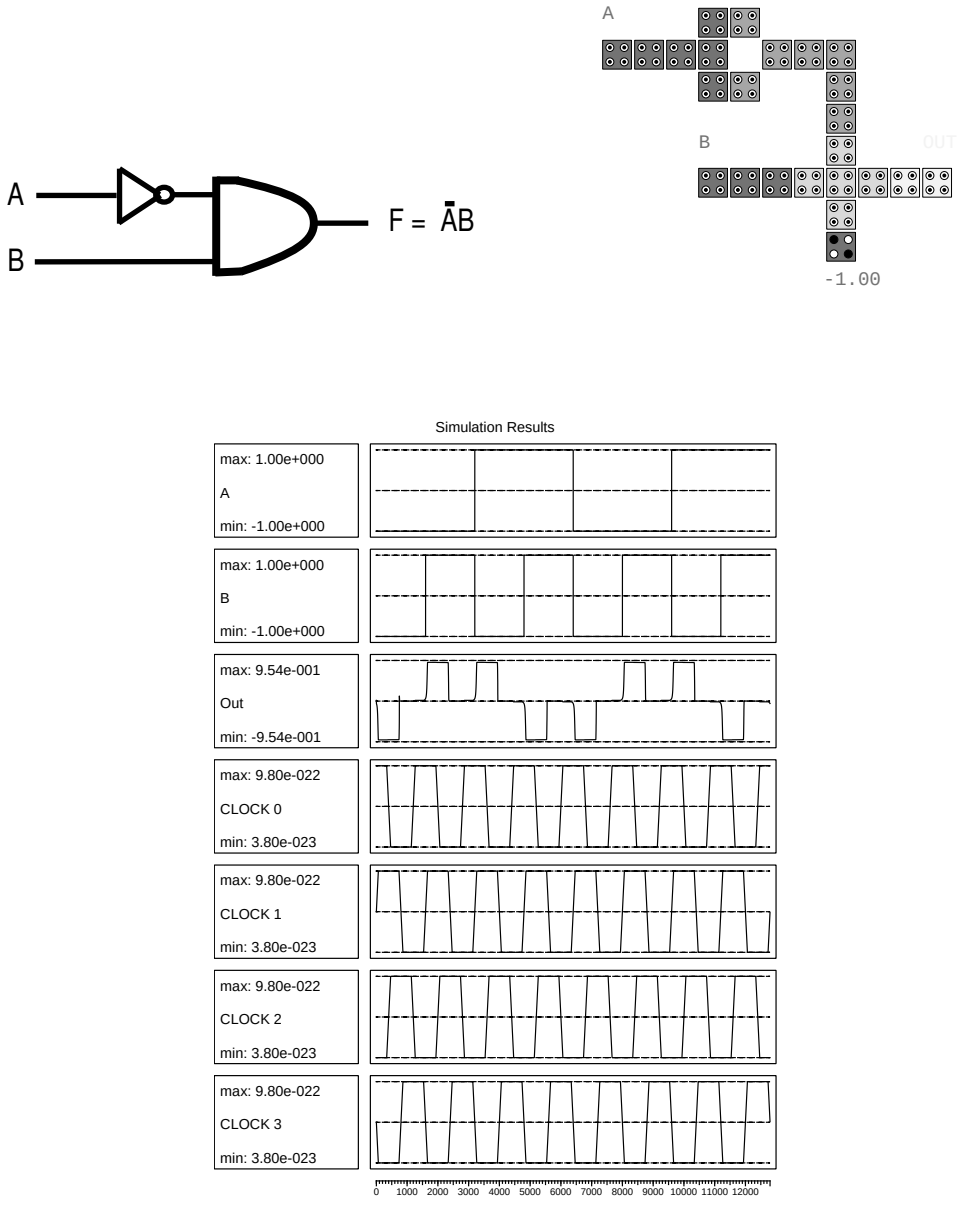


Figure 1.2: Layout and waveform of logic function F = $\bar{A}B$ without proper synchronization.

When the MV gate is in the ”Switch” phase, the wire of input B should be in the

”Hold” phase. Since data can be transferred only between ”Hold” clock phase QCA cells and ”Switch” clock phase QCA cells, a buffer (i.e. an array of cells within the next clock zone) which delays the input B by one clock should be added into the design. In other words, the input wire B should be divided into clock zone 0 and clock zone 1, as shown in Figure 1.3.

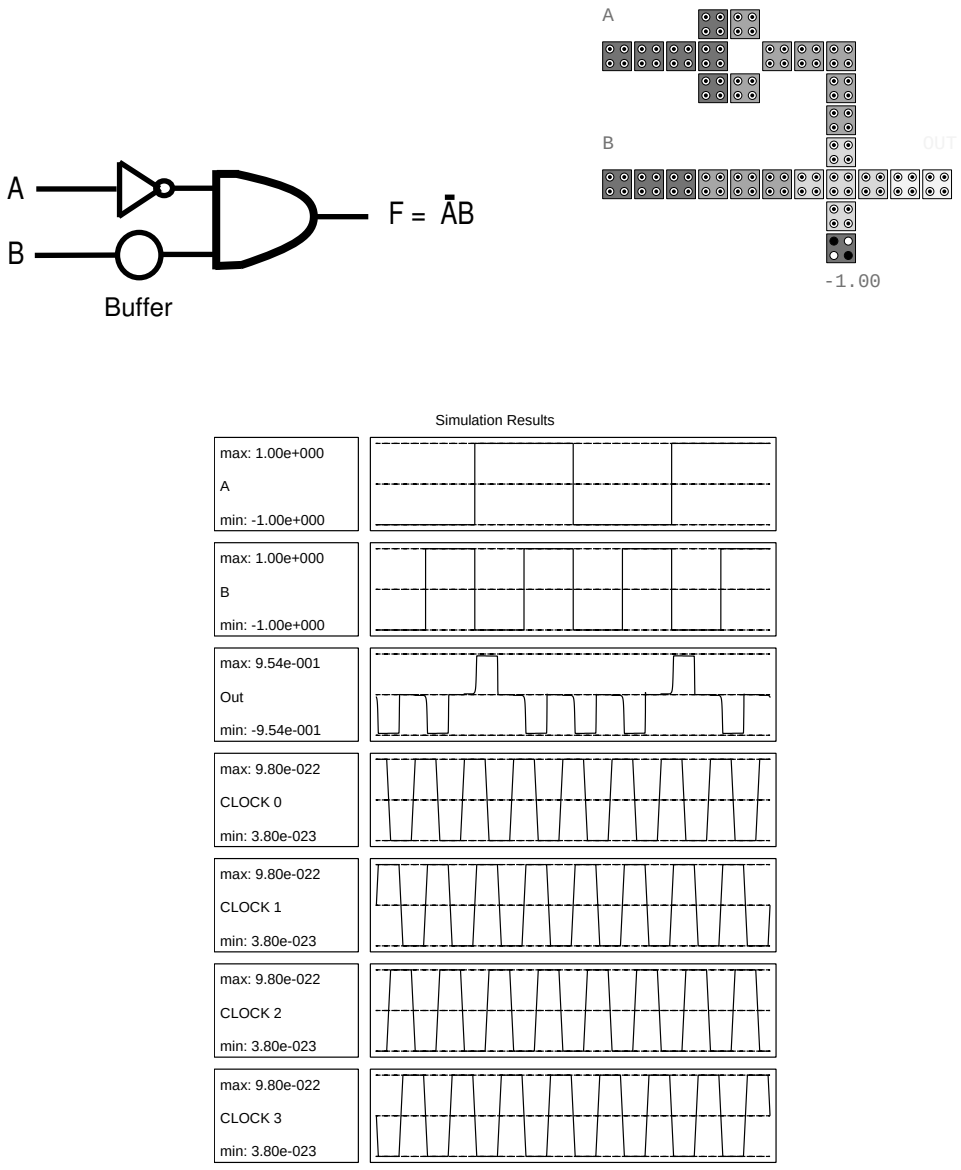


Figure 1.3: Layout and waveform of logic function F = $\bar{A}B$ with proper synchronization.

It is obvious that the buffer consists merely of an array of cells which delays input B by one phase, and this wastes cells as well as space. Basically, in order to synchronize QCA

circuits, many extra buffers are needed, and each buffer has a different number of cells and clocking zones. This fact makes global floorplanning more complex since the number of additional buffer clock zones must be properly detected and considered. In addition, every buffer consists of QCA cells and clocking zones, and they increase the overall cell count and clock zone count. If we can design a new type of AND gate that keeps input B steady until input A comes in, then the buffer can be removed from the design and the gate is made free from ”layout=timing” constraints. These kinds of asynchronous methodologies are well defined in CMOS-based VLSI. The following problems, which are due to ”layout=timing” constraints, will be resolved in this thesis.

- Complex global floorplanning: The synchronization of QCA circuits is an essential factor in running whole circuits correctly. Synchronous QCA circuit design needs a number of buffers to synchronize. Since each buffer may have a different number of clock zones and cells, before designing a QCA circuit, each buffer's clock zone and cells should be completely determined and assigned. This makes global floorplanning more complex.

- Number of clocking zones and cells: It is obvious that an increase in buffers means an increase in clocking zones and cells.

- Wasted Area: Basically, the number of cells in the first clocking zone in QCA circuits may have more cells than the last clocking zone does. If the space of each clocking zone is the same, then the last clocking zone may waste much space. In addition, synchronization buffers are nothing but clocked arrays of QCA cells in a given space. Most buffers, thus, may waste a lot of space.

In order to eliminate such layout=timing constraints from the QCA circuit, NCL (Null Convention Logic), one of the CMOS-based asynchronous circuit design methodologies, will be applied to the QCA paradigm. As one of the CMOS-based asynchronous methodologies, NCL was developed and patented by Karl Fant and Scott Brandt in 1994. NCL is a logic that integrates data transformation and controls it into a single expression (i,e,. DATA or NULL) thus yielding inherently clock-less or delay insensitive circuit and systems.

Because of the separation between control and data representation, there is no invalid data for NCL circuits. On the other hand, the validity of the NCL DATA state depends on data encoding, not time synchronization. After having a valid output, NCL circuits ignore any invalid inputs and they keep its value until all of its inputs become NULL. When they have NULL output, they continue to be NULL until the expected number of inputs has became valid data. Therefore, there is no invalid output between the DATA and the NULL output of NCL circuits. If there is no NULL output of a NCL circuit between two different DATA outputs of the circuit, it means that the circuit is not a valid NCL circuit. NCL uses a combination of multi-wire data representation and control protocol, and NCL circuits only switch between data representation of DATA and a control representation of NULL. So, no clock is needed for NCL circuits [21].

Since the concept of QCA clocking (i.e., 4-phase clocking) brings many critical design problems such as complex floorplanning, circuit complexity, reduced circuit density, wasted space, and increased cell count, complete applying of CMOS-based NCL asynchronous methodology may eliminate strict clocking limitations from QCA circuits. On the other hand, synchronous QCA circuits do not obey the ”layout=timing” problem, because of their delay-insensitivity. Therefore, the NCL provides following advantages for the QCA circuit design:

- Easier global floorplanning: The synchronization of QCA circuits is an essential factor in running the whole circuit correctly. Synchronous QCA circuit design needs a number of buffers to synchronize. Since each buffer may have a different number of clock zones and cells, before designing a QCA circuit each buffer’s clock zone and cells should be completely determined and assigned. QCA-based NCL circuits are free from complex floorplanning since they do not need buffers for synchronization.

- Reduced number of clocking zones and cells: It is obvious that NCL circuits are not needed in order to include extra buffers for synchronization purposes. So, cell count and clock zone count will be reduced for NCL circuits.

- Increased density: Since the buffers are nothing but clocked arrays of QCA cells in

given space, most buffers may waste a lot of space. Elimination of buffers provides better space density for NCL circuits.

CHAPTER 2

PRELIMINARIES AND REVIEW

Since they are quantum mechanical particles, the electrons in QCA cells can tunnel between the dots in a cell. The electrons placed adjacent to each other will interact due to the energy difference between them. This is called kink energy. [9, 8, 13, 20, 10]. Any cell which is not polarized cannot affect its neighboring cells. On the other hand, the polarization of one cell will be directly affected by the polarization of its neighboring cells. The interaction between two neighboring cells is known to show cell-to-cell response. This interaction forces nearby cells to synchronize their polarization. Therefore, the given information (logic 0 or 1) can be transmitted from input cell (or fixed polarization cell) to destination cell, and any array of cells between the input and the destination cell should act as a binary wire. The basic QCA wire is shown in Figure 2.1(a).

2.1 QCA Primitive Gates and clocking

In order to design complete logic circuits with QCA cells, two simple QCA gates are used with a QCA wire. The most fundamental gate in QCA is the majority voter (MV) [11, 20, 6]. The output of the MV gate is equivalent to a logic function F(A,B,C) = AB + AC + BC. A MV gate consists of five QCA cells arranged in a cross as shown in Figure 2.1(b). Both logic AND gate and OR gate are easily created by fixing one input of the MV gate. For example, if the input C is fixed at logic input 1, then logic function can be implemented as F(A,B,1) = AB + A1+ B1 = A + B. The MV gate, which includes one fixed logic input 1, is exactly the same as the logic OR gate. When one of the MV inputs is fixed to logic input 0, then the MV gate is equivalent to logic AND gate. If the input C is fixed to logic input 0, then a logic function can be implemented as F(A,B,0) = AB + A0 + B0 = AB. Since the

MV gate's only logic set is not complete, QCA inverter gate, shown in Figure 2.1(c), is also used along with the MV gate.

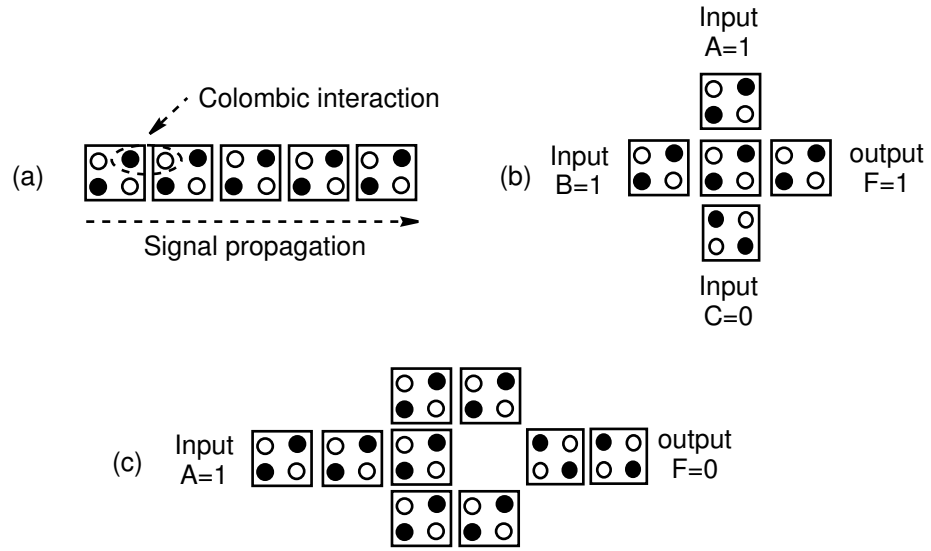


Figure 2.1: QCA primitive devices[7]: (a)Binary wire. (b)Majority Voter. (c)Inverter

The QCA clock is quite different from the standard CMOS clock. In a CMOS-based VLSI, the clock is used as a data signal. Basically, the CMOS clock has a high phase as well as a low phase. For example, clock high means that the data is binary 1 and clock low means that the data is binary 0. The QCA clock is used not only to synchronize and control data flow, but also to provide the power to run the circuit [12, 3]. The QCA clock has switch, hold, release and relax phases [14]. This concept of QCA clocking is the 4-phase clocking. During the first clocking phase, the switch phase, all QCA cells in the first clocking zone become polarized due to their input cells or the neighboring cells of the previous clocking zone. This causes their interdot potential barriers to be raised. This is the essential phase in which the actual computation occurs. The second phase is the hold phase. During this phase, the barriers are held high and they restrain any electron tunneling of cells in this clocking zone. Therefore, nearby cells of the next clocking zone keep computed results and act as inputs for cells of the next clocking zone. In the third phase, the release phase, barriers are lowered and cells are unpolarized. In the last phase, the relax phase, barriers and cells maintain their end status of the release phase. Figure 2.2

shows the 4-phase clocking scheme.

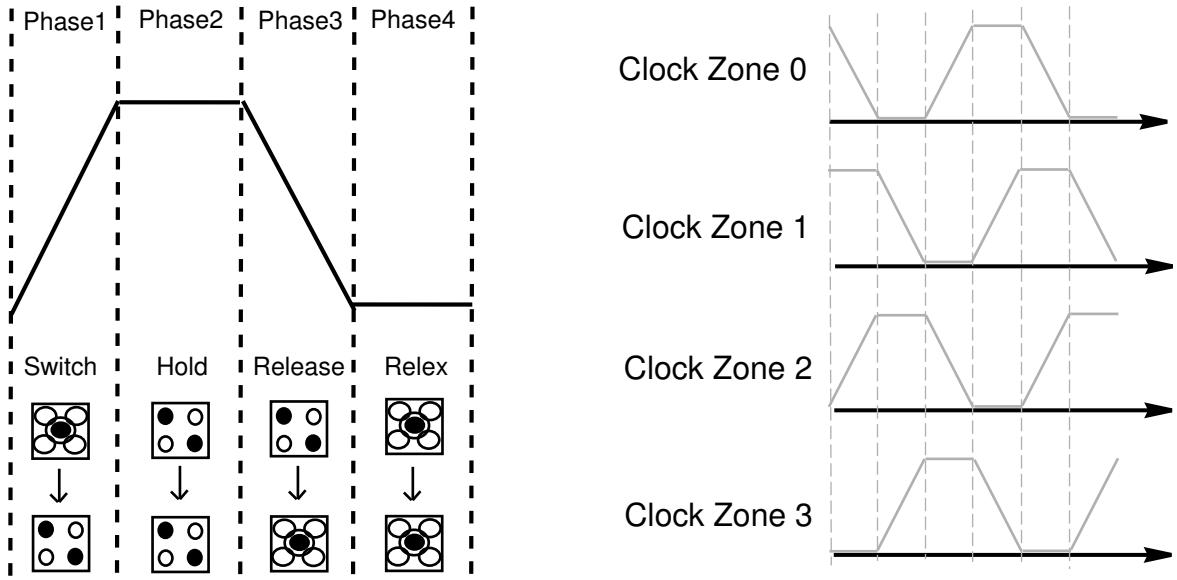


Figure 2.2: 4-phase clocking(left) and QCA clocking zones(right).

Clocking zones grant classical latch behavior to cells. So, since a data can be stored in each cell's latched polarization, the four phase clocking scheme emulates shift register behavior. In order to get a correct data from QCA circuits, proper order of clocking zones should be considered.

2.2 Design Challenges in QCA

With the four phase clocking scheme, it is possible to design QCA circuits while ensuring that each QCA cell is powered, and that the information is processed and forwarded in a timely manner. However, inherited characteristics of QCA, such as the way to hold state, the way to synchronize data flows, and the way to empower QCA cells, make the design of QCA circuits quite different from CMOS-based VLSI and introduce many new design challenges. The most severe challenges are due to the fact that the overall timing of a QCA circuit is mainly dependent on its layout. This fact is commonly referred to as the ”layout=timing” [15, 16]. According to the four-phase clocking scheme, QCA cells placed in one clocking zone require one complete clock cycle (four-phases) to receive the

information from the previous clocking zone and forward the information to the following clocking zone. So, QCA wires and gates in QCA circuits must be placed in the proper clocking zone. For example, let us consider a QCA gate with two input wires. If these two inputs do not come through the same number of clocking zones, one of the inputs could arrive at the gate before the other one does. As a result, incorrect output may be obtained. There are three QCA physical design procedures are described in [15, 25]

- Partitioning: In this step, the given QCA circuit is partitioned so that it can be placed into corresponding clocking zones and fulfill the timing constraints. Each clocking zone and its following clocking zones in the given QCA circuit must be in proper order (i.e., any clocking zone in the ”hold” phase should be followed by neighboring clocking zones in the ”switch” phase). It is important that each clocking zone has a similar width and height so that the buried clocking wire can be uniformly distributed with ease.

- Placement: In a QCA environment, wire crossings are expensive in terms of manufacturing since either the use of large circuits to exchange the position of two signals, or a 45-degree change in the cell orientation is expensive. So the number of wire crossings should be minimized. The devices are arranged within their chosen clocking zones in such a manner that the number of wire crossings is minimized.

- Routing: In order to minimize number of wire crossings, optimal wire routing design is needed.

These three design procedures are mutually dependent. If more clocking zones are provided, it will be much easier to make all the signals reach their destinations in a timely manner. However, providing more clocking zones may increase the latency of the circuit [14]. If we try to make a good placement by minimizing the number of wire crossings, the routing will become difficult and may need larger spaces to route the wires and more cells to lengthen the wires. In addition, if we try to make the clocking zones have a uniform width and height, we will lose some flexibility for the placement. Five significant problems

with designing QCA circuits due to the ”layout=timing” problem have been described in [16]:

- *Wire Length*: When designing QCA circuits, the length of a wire within a given clocking zone should be minimized. Since the lengthening of a wire means that the number of cells in the wire is increased, the probability that a QCA cell will switch decreases in proportion to the distance. Also, wire length determines the clock rate, because every cell within the clocking zone must make the appropriate polarization changes before a given zone can change phase. Since shorter wires ensure a greater probability that all cells will be polarized properly, minimizing the wire length is significant.

- *Clocking Zone Width*: Clocking zone widths should also be minimized in such a manner that wire lengths can be minimized and uniformed to increase the manufacturability of the circuit.

- *Number of QCA cells per Clocking Zone*: If too many cells are included in a single clocking zone, the clock rate could deteriorate simply because the time it takes for all the cells to make the required transition will most likely increase.

- *Wasted Area*: Basically, the number of cells in the very first clocking zone in QCA circuits (i.e., number of cells in input wires) tends to be more than the number of cells in the last clocking zone (i.e., number of cells in output wires). If the space of each clocking zones is the same, then the space efficiency of the last clocking zone is lower than the first clocking zone. Also, inclusion of buffer cells in the design to synchronize data flow increases wasted area.

- *Lack of Feedback*: In order to design sequential circuits, data feedback is essential. Since the 4-phase clocking scheme allows data to flow in one direction, including feedback loops in QCA circuit design may complicate clocking and floor planning.

As methods to solve these problems, the ”trapezoidal clocking” and the ”universal clocking cell” have been proposed in [16]. QCA inherently lends itself to such a ”trapezoidal clocking” structure in which there is initially n inputs and after a certain number of m clocking zones only an output remains. It is possible to stack another trapezoid with opposite data flow to make feedback possible. By allowing data to flow in two directions and by carefully fitting trapezoids together, denser and compact QCA circuits could be generated. Also, clocking zones can be arranged or tiled so that there are multiple wire loops and wire crossings to allow feedback and routing. The universal clocking floorplan is a standardized clocking zone structure in which various functions can be implemented to allow feedback and routing. These two architectures focus on synchronization of QCA circuits. Unlike this approach, an asynchronous methodology such as NCL (Null Convention Logic) [21]can be used to eliminate such ”layout=timing” problems.

2.3 The Null Convention Logic

The Null Convention Logic (NCL) was developed by Theseus Logic, Inc., Orlando, FL and patented as an innovative asynchronous paradigm in 1994. NCL is a clock-free and delay-insensitive logic design for digital systems [22]. Instead of representing a logic state in NCL logic, a wire defines the arrival of data. Wires are combined to form a definition of logic states. Any delay insensitive encoding can be used such as dual rail, quad rail and so on, but in this paper dual rail encoding will be used.

NCL uses symbolic completeness of expression to achieve self-timed behavior. A symbolically complete expression is defined as an expression that only depends on the relationships of the symbols present in the expression without a reference to the time of evaluation. Traditional Boolean logic is not symbolically complete; the output of a Boolean gate is only valid when referenced with time. For example, let us assume it takes $1ns$ for output Z of an AND gate to become valid once its inputs X and Y have arrived. As shown in Figure 2.3, initially suppose $X = 1$, $Y = 0$, and $Z = 0$. If Y changes to 1, Z will change to 1 after $1ns$; so Z is not valid from the time Y changes until $1ns$ later. Therefore output Z not only

depends on the inputs X and Y, but time must also be referenced in order to determine the validity of Z. This can be critical when Z is used as an input to another circuit.

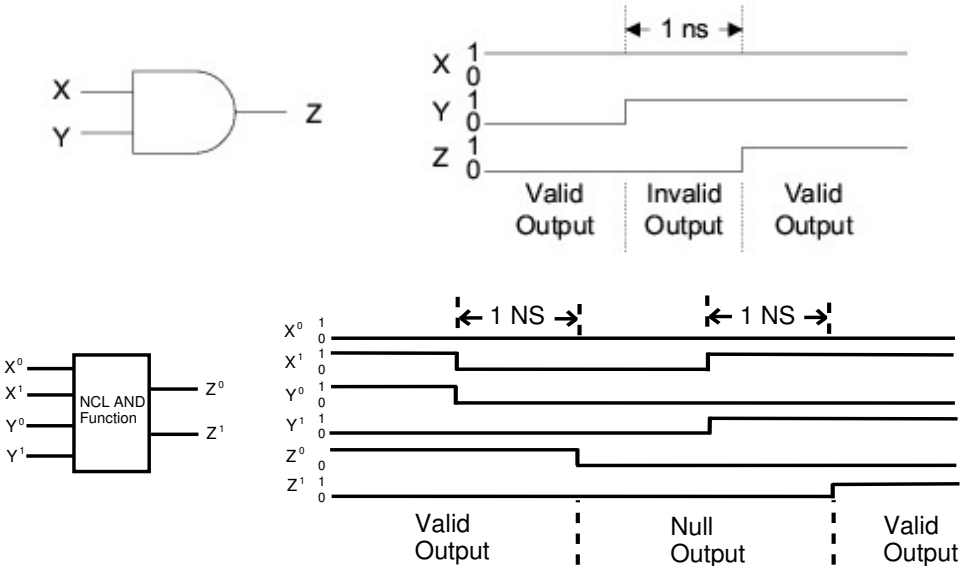


Figure 2.3: (Top): Symbolic incompleteness of Boolean AND gate. Output Z not only depends on the inputs X and Y, but time must also be referenced in order to determine the validity of Z. (Bottom): Dual-rail encoded NCL AND function and associated waveforms. Due to the hysteresis behavior inherent in each threshold gate. Time is never referenced to determine the validity of Z. Figure adopted from [27]

With dual rail encoding, NCL uses a combination of dual wires and NCL circuits switch between a logic based data representation of DATA and a control representation of NULL. On the other hand, combination of dual rail represents one of three states, DATA0, DATA1 and NULL. For example, if there are dual wires D0 and D1 for dual rail signal D of a certain NCL circuit, value D depends on the combination of D0 and D1. The DATA0 state (D0 = 1, D1 = 0) corresponds to a Boolean logic 0, the DATA1 state (D0 = 0, D1 = 1) corresponds to a Boolean logic 1, and the NULL state (D0 = 0, D0 = 0) corresponds to the value of D, which is not yet available. The two rails are mutually exclusive so that both rails can never be asserted simultaneously; this state (D0 = 1, D1 = 1) is defined as an illegal state. The separation between control (i.e., NULL state) and data (i.e., DATA state) representations provide a self-synchronization for NCL designs. Consider the behavior of a symbolically

complete AND function using NCL as shown in Figure 2.3.

CHAPTER 3

PROPOSED QCA-BASED NCL DEVICES AND DESIGN

NCL has been proposed as an asynchronous system design methodology for CMOS [23]. It is difficult to realize NCL in QCA directly since the circuit operation mechanism of QCA is substantially different from the CMOS. In order to realize NCL in QCA, basic QCA-based threshold gates with hysteresis, as basic building blocks of NCL, will be proposed. Then the manner in which it would be possible to eliminate global delay-dependency among logic gates in QCA circuit by utilizing NCL will be investigated. A NCL-based 1-bit full adder which consists of QCA-based threshold gates has to be designed in this work. This will be compared to a traditional QCA full adder in which NCL has not been applied. Finally, design analysis and optimization techniques, and some design metrics for QCA-based NCL circuits, such as area overhead in terms of cell count and clock zone count assignment overhead will be proposed.

In order to design QCA-based threshold gates with hysteresis, an EDA (Electronic Design Automation) tool for QCA "QCADesigner" developed by the University of Calgary ATPTI laboratory [26] will be used. The latest version of "QCADesigner" (version 2.0.4) provides two different simulation engines: coherence vector and bistable. The bistable engine will be used in the "QCADesigner V2.0.4"

3.1 QCA-based NCL Devices and Full Adder

Threshold gates with hysteresis are the basic building blocks of NCL designs. One type of threshold gate with hysteresis is the M-of-N threshold gate with hysteresis. This is denoted as THmn gate, and has n number of inputs and one output signal where $1 \leq m < n$ [23]. THmn gate has two special properties: threshold behavior and hysteresis behavior. The threshold behavior means that the output of the THmn gate becomes 1 when at least m of

n inputs becomes 1. The hysteresis behavior means that the output only changes after a sufficiently complete set of inputs have been established. For example, if the current output of TH23 gate is 0, then the output remains 0 until at least 2 of 3 inputs become 1. In addition, to change the output from 1 to 0, all 3 inputs must be 0. The symbols of THmn gates and TH23 gates are shown in Figure 3.1(a) and (b).

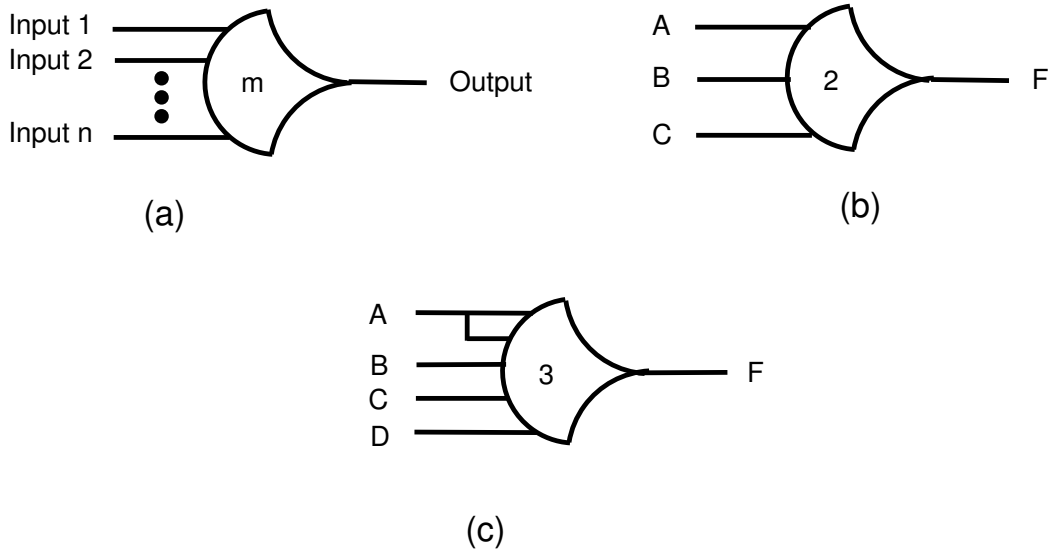


Figure 3.1: (a)The THmn gate symbol, it has been adopted from [23]. (b) TH23 gate symbol, and (c) TH34W2 gate symbol. Note that weight of the input A is 2.

Note that TH23 gate is equivalent to a sequential logic Boolean function $F_{t+1} = (A + B + C)F_t + AB + BC + AC$, where F_{t+1} is the current output, and F_t is the last output of the gate. In order to demonstrate the feasibility of designing NCL gates with QCA cells while preserving the 4-phase clocking for the gates, a QCA-based TH23 gate [**?**] has been designed using QCADesigner V2.0.4. Layout of the gate is shown in Figure 3.2(Top). Five MV gates and one feedback have been used in the design. Since a MV gate can be used as either logic "AND" gate or "OR" gate, four MV gates are used to compute "$(A + B + C)F_t+$" part, and just one MV gate is needed to implement AB+BC+AC. A feed back loop is used to forward the previous output F_t into the circuit. Nine different clock zones have been used in the design. With 4-phase clocking, every fourth clock zone is in the same phase. For example, all inputs of the TH23 and the output F are in the same

phase since the output F is in the eighth clock zone. This means that there are two cycle delays between the inputs and the output as shown in Figure 3.2(Bottom).

Another type of threshold gate is referred to as a weighted threshold gate, denoted as THmnWw1,w2,...,wR, where n is number of inputs, m is threshold, and w1, w2,...,wR are the integer weights of input1, input2,...,inputR, respectively [23]. For example, consider a TH34W2 gate, whose n = 4 inputs are labeled A, B, C, and D. The weight of input A, W(A), is therefore 2. Since the gate's threshold m is 3, this implies that in order for the output to be asserted, either inputs B, C, and D must be asserted together, or input A must be asserted with any one of B, C, or D. The symbol of the TH34W2 gate is shown in Figure 3.1(c). Note that the TH34W2 gate is equivalent to a sequential logic Boolean function $F_{t+1} = (A + B + C + D)F_t + (BC + BD + CD)A + BCD$, where F_{t+1} is the current output, and F_t is the last output of the gate. Likewise, a QCA-based TH34W2 gate [**?**] also has been designed in the exact same manner which was used to design QCA-based TH23 gate. The layout and the waveform of the gate are shown in Figure 3.3. Note that there are 12 different clock zones which have been used in the design. Therefore three cycle delays exist between the inputs and the output. Notably, the local network of QCA cells within both TH23 and TH34W2 gates are fully synchronized and powered by the 4-phase clocking scheme. But the gates themselves are delay-insensitive since it has threshold and hysteresis behaviors.

A synchronous QCA full adder consists of three MV gates, two inverters, and 5 different clock zones and 165 cells are used in the design as shown in Figure 3.4. Although this full adder has logically minimized form, it is not free from the "layout=timing" problem. For example, let us use the full adder as a building block to develop a 2-bit full adder. This is simply done by making a connection between the carry-out of the first bit full adder and the carry-in of the second bit full adder and adding more buffers for synchronization. Since the carry-in value of the second bit full adder will be delayed by 5 clock zones, the other two inputs of the second bit full adder should be delayed by 5 clock zones. In addition, output of the first digit full adder should also be delayed by 5 clock zones. The layout of the synchronous 2-bit full adder is shown in Figure 3.5(A). It is obvious that the buffers

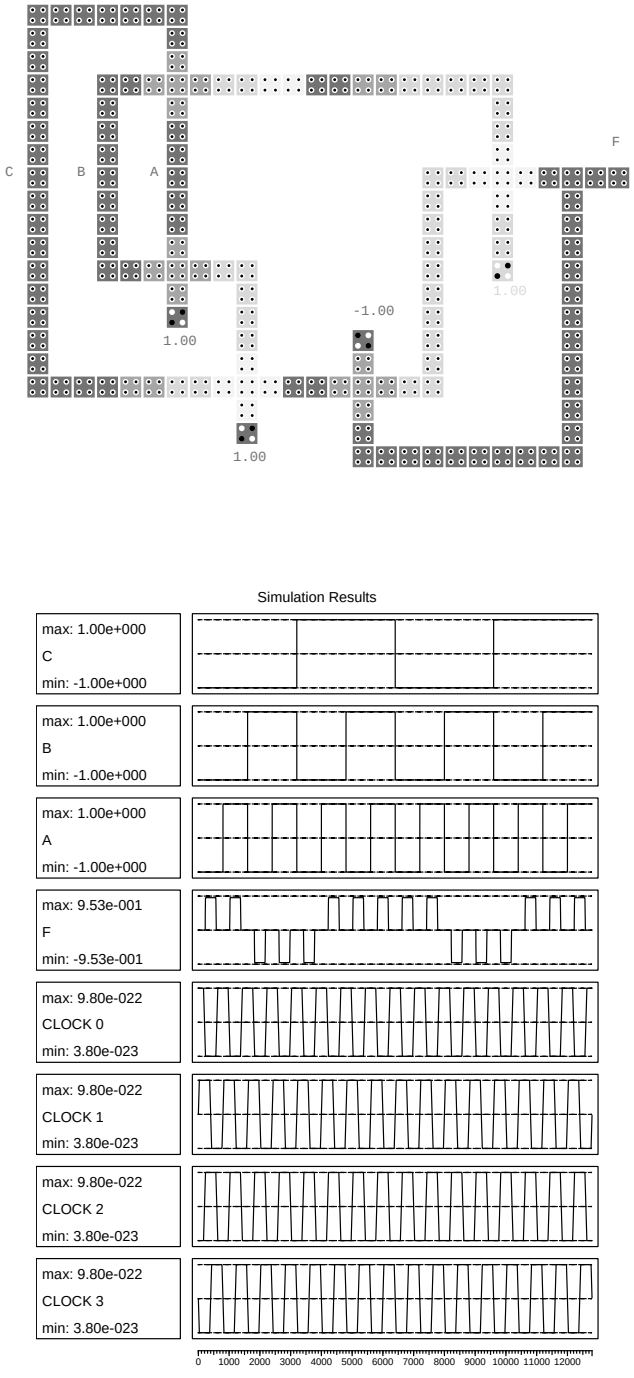


Figure 3.2: QCA-based TH23 gate and its waveform.

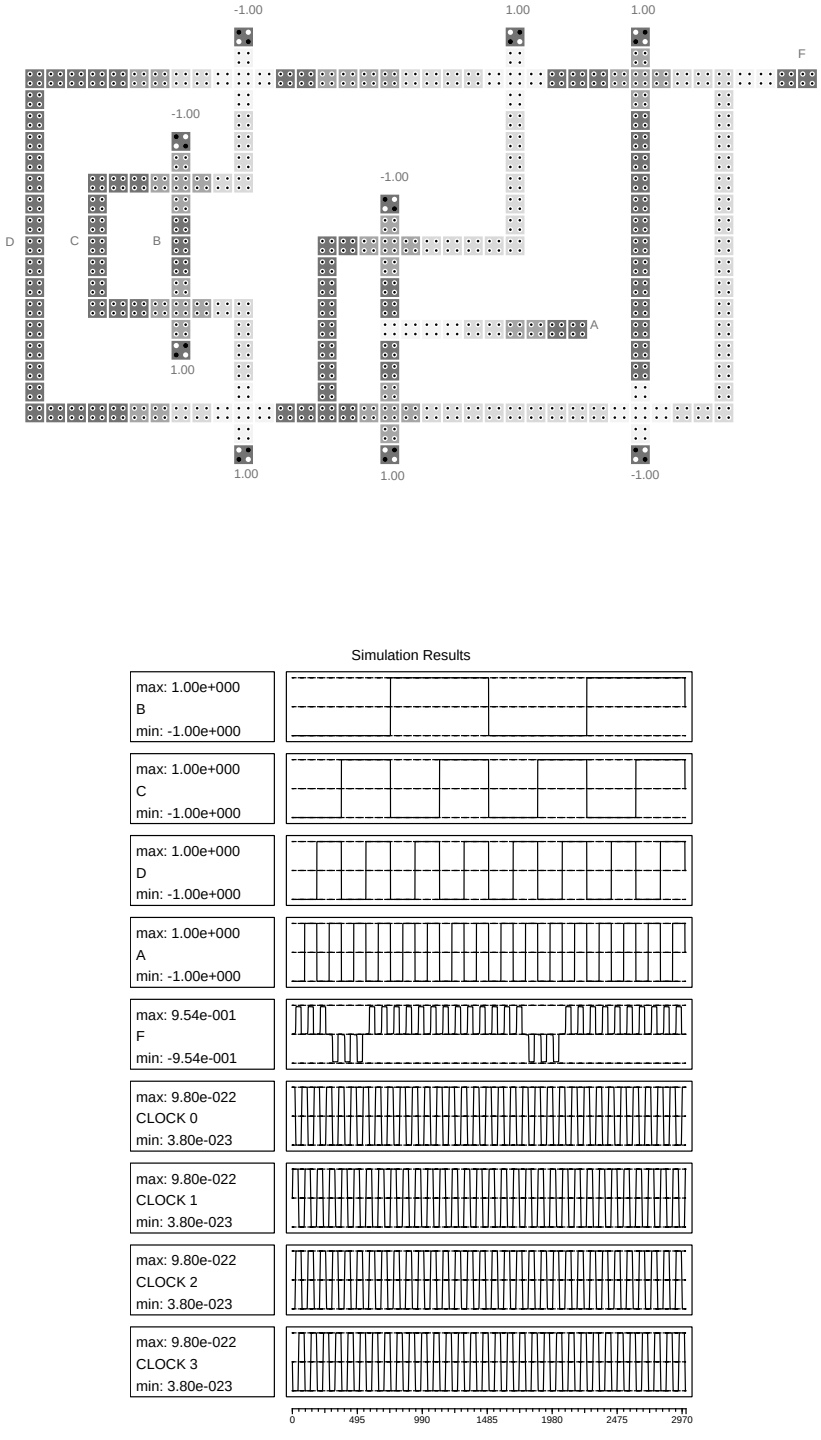


Figure 3.3: QCA-based TH34W2 gate and its waveform.

consist of clocked wires and that they waste a lot of cells and space. This design may cause complications due to the ”layout=timing” problem which was discussed in section 2.3.

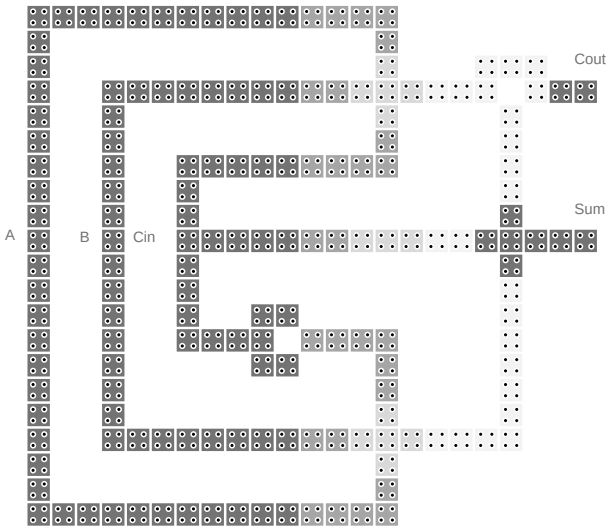


Figure 3.4: The layout of synchronous QCA full adder.

A dual rail full adder consists of two TH23 gates and two TH34W2 gates [23, **?**] as shown in Figure 3.6. Since both TH23 and TH34W2 gates are delay-insensitive, they can be located anywhere in the given area and the corresponding interconnects and fan-ins and fan-outs can be also freely placed, because the global timing dependency is fully removed. The NCL full adder has inherited threshold and hysteresis behavior from TH23 and TH34W2 gates. This NCL full adder as well as any NCL circuit consisting of NCL full adders does not obey the ”layout=timing” constraint. For example, when we design a 2-bit full adder by using a NCL full adder as a building block, a 2-bit full adder is simply obtained by connecting the carry-out of the first digit NCL full adder to the carry-in of the second digit NCL full adder. Each NCL full adder in 2-bit full adders is delay-insensitive; so they can be placed in a same clock zone as shown in Figure 3.5(B). Therefore, any NCL circuit designed with NCL full adders also has the same properties that a NCL full adder does.

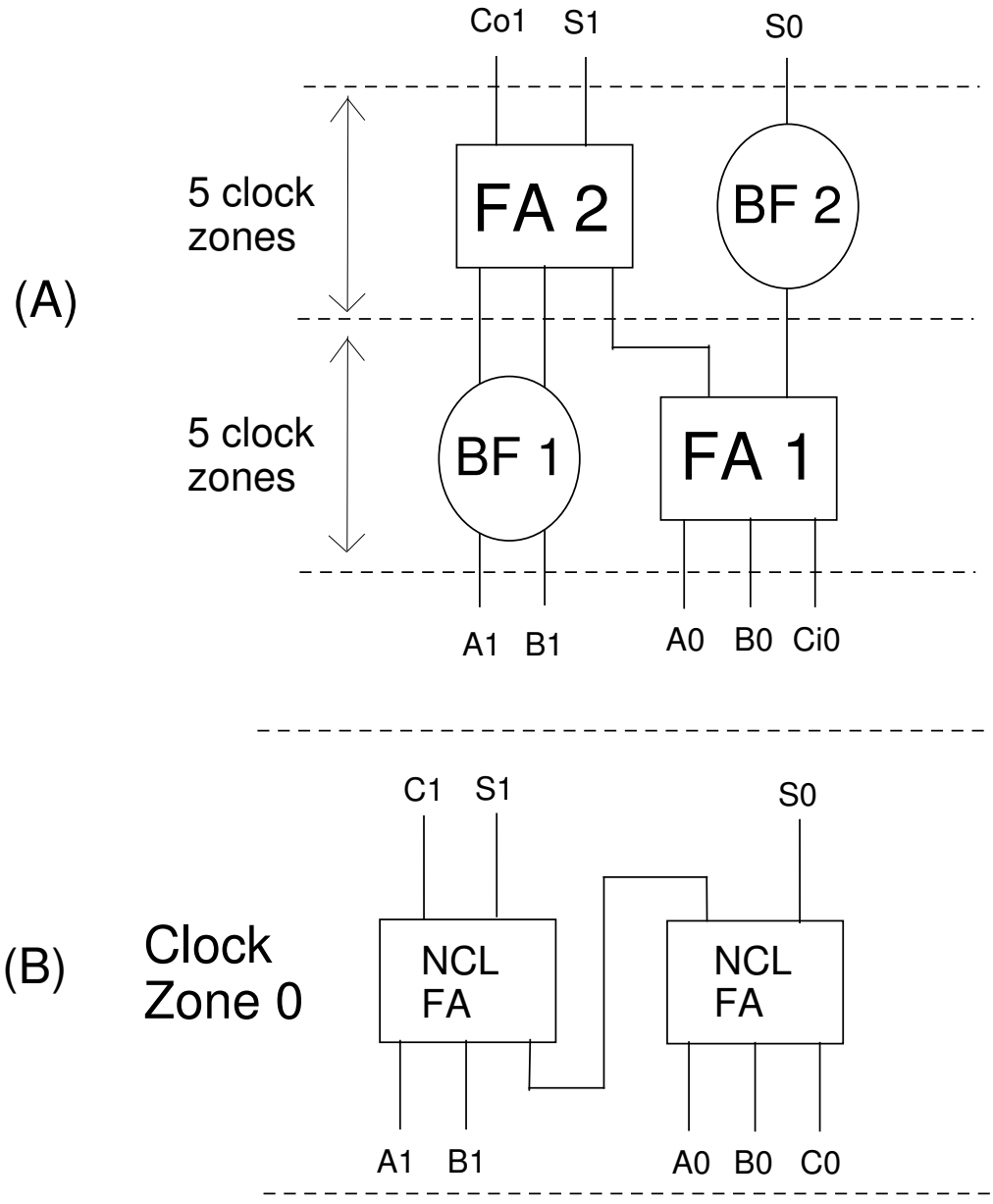


Figure 3.5: (a)The 2-bit full adder using synchronous QCA full adder as building block. In order to synchronize the circuit, two buffers are added. Addition of buffers may cause serious problems such as wire length, number of cells per clock zone, and wasted area problem [16]. (b)NCL 2-bit full adder scheme. Assume that all inputs and outputs of full adders are in a same clocking phase (i,e,. "Switch" phase for the clock zone 0). In other words, every wires outside full adders are nothing but either extended inputs or extended outputs in a same clock zone. Because all wires outside full adders are located in a single clock zone, and NCL full adders are delay-insensitive, the full adders can be placed anywhere in the given area.

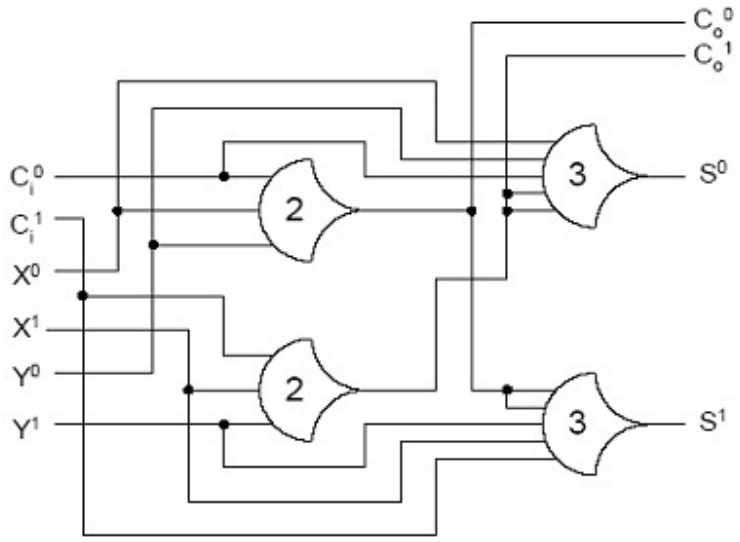


Figure 3.6: Dual-rail full adder design. Two TH23 and two TH34W2 gates are used. Since the design is placement-insensitive when implemented in QCA, since all wires outside THgates are in a single clock zone and THgates are delay-insensitive. The design can be located anywhere in the given area. The Figure adopted from [23].

3.2 Design Integration

Designing a NCL-based QCA 1-bit full adder with QCADesigner can be done simply by combining two TH23 gates, two TH34W2 gates, and wires in a single clock zone. But implementation of the full adder with QCADesigner has complex multiple wire crossing problems. Typical QCA wire crossings can be implemented with an array of 90-degree cells and an array of 45-degree cells (i.e.,inversion chain) since the 90-degree cell cannot have an affect on a 45-degree cell in a direct line, and also since a 45-degree cell has no switching effect on a 90-degree cell in a direct line [28]. In order to complete the QCA wire crossing, there are two special devices called X-to-+ (i.e.,90-cell to 45-cell) and +-to-X (i.e.,45-cell to 90-cell) converters. A simple QCA wire crossing example is shown in Figure 3.7.

Multiple wire crossings should be implemented with this simple wire crossing design. When we implement two wire crossings between arrays of 45-degree cells and arrays of 90-degree cells, it can be simply done by combining two simple wire crossings, shown as 3.8.

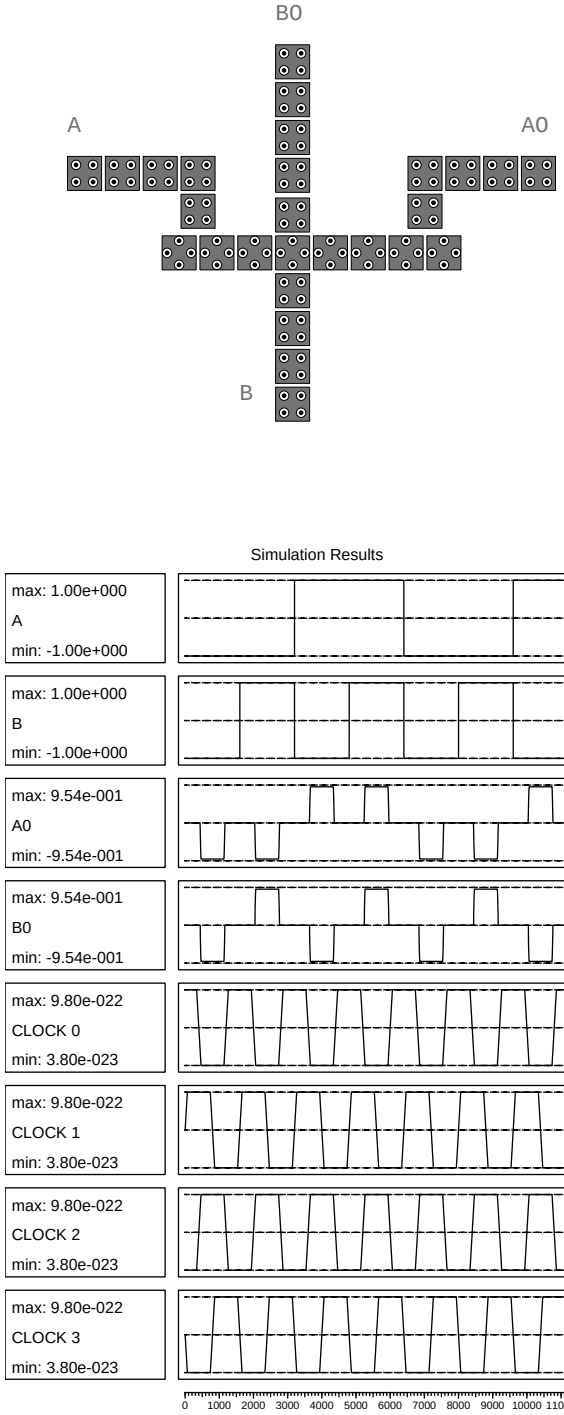
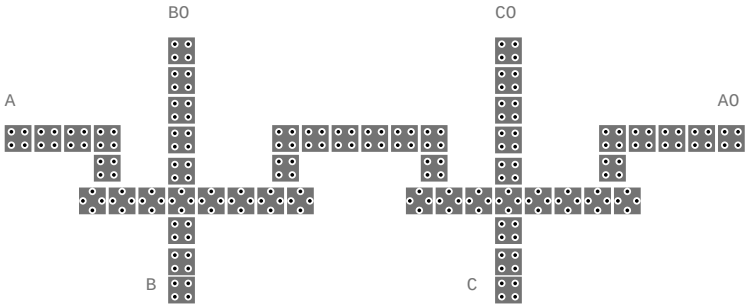


Figure 3.7: Simple QCA wire crossing with X-to-+ converter and +-to-X converter. Even though All cells in a same clock zone, 90-degree cells transmit their state to not 45-degree cells but 90-degree cells and 45-degree cells transmit their state to 45-degree cells . Figure adopted from [28].



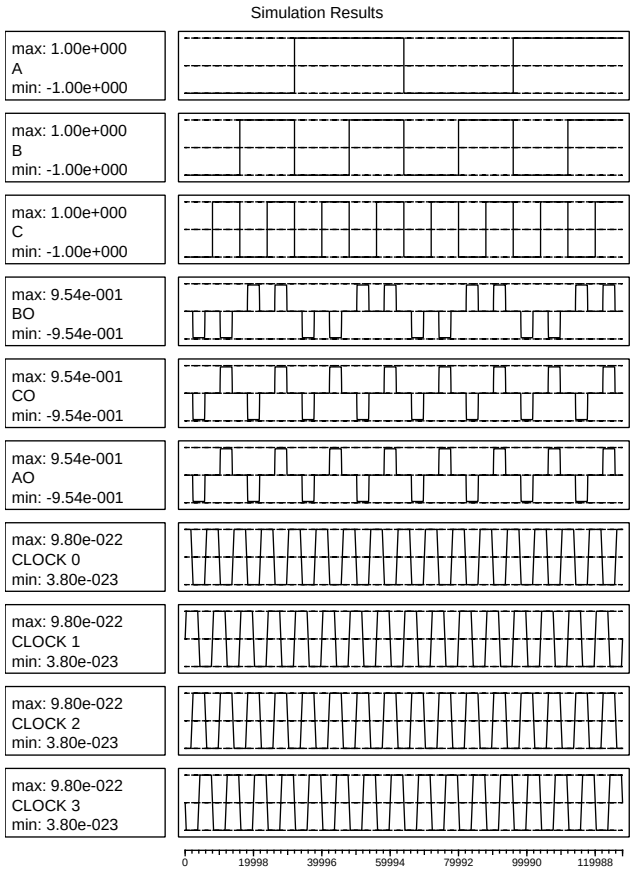


Figure 3.8: A simple example of multiple wire crossings. The Output AO yields an incorrect output. The AO is effected by not input A but input C.

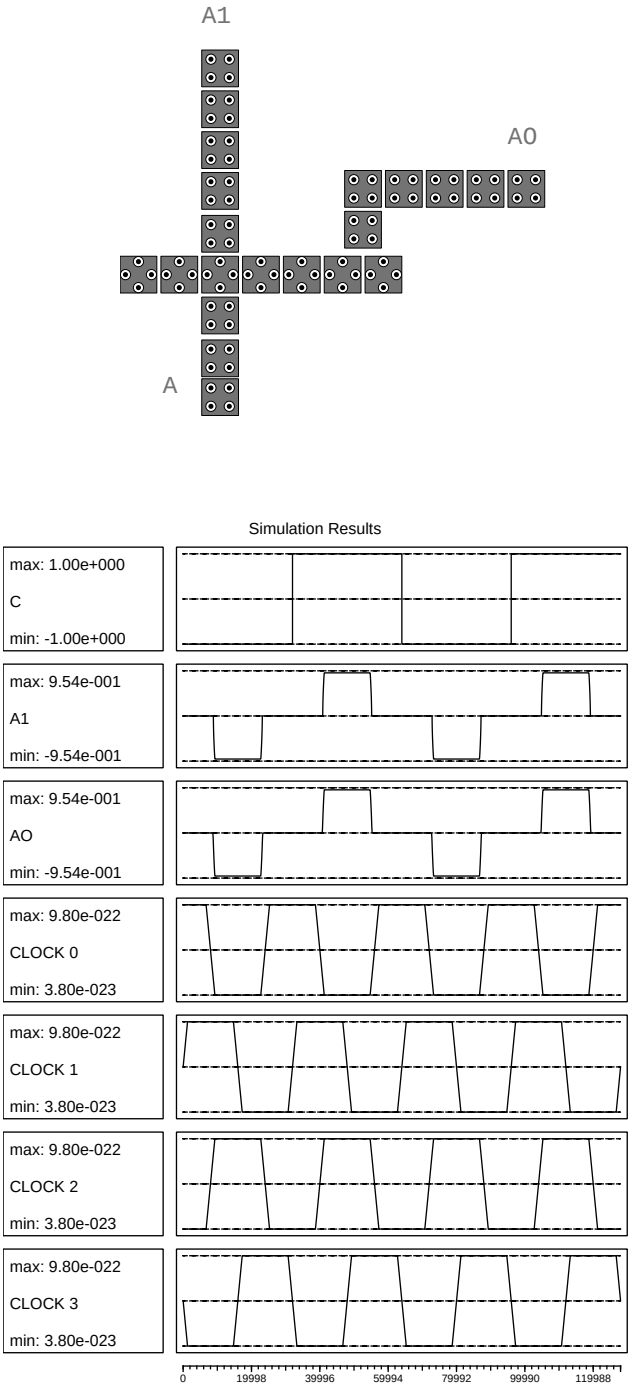


Figure 3.9: This example describes relation between 90-degree cell and 45-degree cell with QCADesigner simulation tool. The Output AO is exactly same as output A1. Even though A0 is output from 45-degree cell, it has fan-out value of 90-degree cell input A.

The output AO in the Figure 3.8 is effected by input A and follows input A's wavefront. But AO is exactly the same as output CO. This means that input C switches output AO. On the other hand, a 90-degree cell can switch a 45-degree cell in a direct line with QCADesigner simulation tool. If a 45-degree cell and a 90-degree cell can affect each other in the direct line, QCA wire crossing by 45-degree cell and 90-degree cell method cannot be used in QCADesigner. Figure 3.9 has been designed in order to illustrate the relations between a 45-degree cell and a 90-degree cell. This design shows us that QCADesigner does not distinguish between 45-degree cells and 90 degree cells in a same line since the 45-degree output AO is affected by the 90-degree input A. Even if we design a complete circuit with the 45-degree and the 90-degree cell wire crossing method, the QCADesigner may yield an incorrect output. So a 45-degree cell with a 90-degree cell wire crossing method cannot be implemented with the QCADesigner. The QCADesigner provides for multiple layers to cross QCA wires. The NCL full adder will be developed with multi-layer wire crossing method.

NCL circuits need a device that monitors the outputs of NCL circuits [27]. The NCL register keeps its data until it receives all of its inputs, before requesting the next either NULL or DATA wavefront from the previous register. This is the hysteresis behavior that NCL circuits usually have. In order to verify the validity of QCA-based NCL full adder, a register stage must be attached to the single full adder. In this case, a simple QCA-based NCL register can be designed with six SR flip-flops, since the register of the single full adder does not request a control signal from the next register. Since a dual-rail NCL full adder has two input rails per input X, Y, and Ci, six total SR flip-flops would be attached to each of the input rails. A SR flip-flop has two inputs (i.e., S and R) and two outputs (i.e., F and $\bar{F}$), where $\bar{F}$ is complement of F. If S and R are both 0, F and $\bar{F}$ retain their last value. A SR flip-flop consists of two nor gates and two feedback loops. In the QCA paradigm, nor gate is merely a combination of the MV gate and the inverter gate. Therefore, a QCA version of SR flip-flop can be designed by two inverters, two MV gates, and two feedback loops. Even though the SR flip-flop has two outputs, only one output is used as an input to the NCL full adder. On the other hand, one of input F and $\bar{F}$ can be removed. Finally, a

modified SR flip-flop consists of one inverter, one MV gate, and one feedback loop. The modified QCA-based SR flip-flop and its waveform are shown in Figure 3.10.

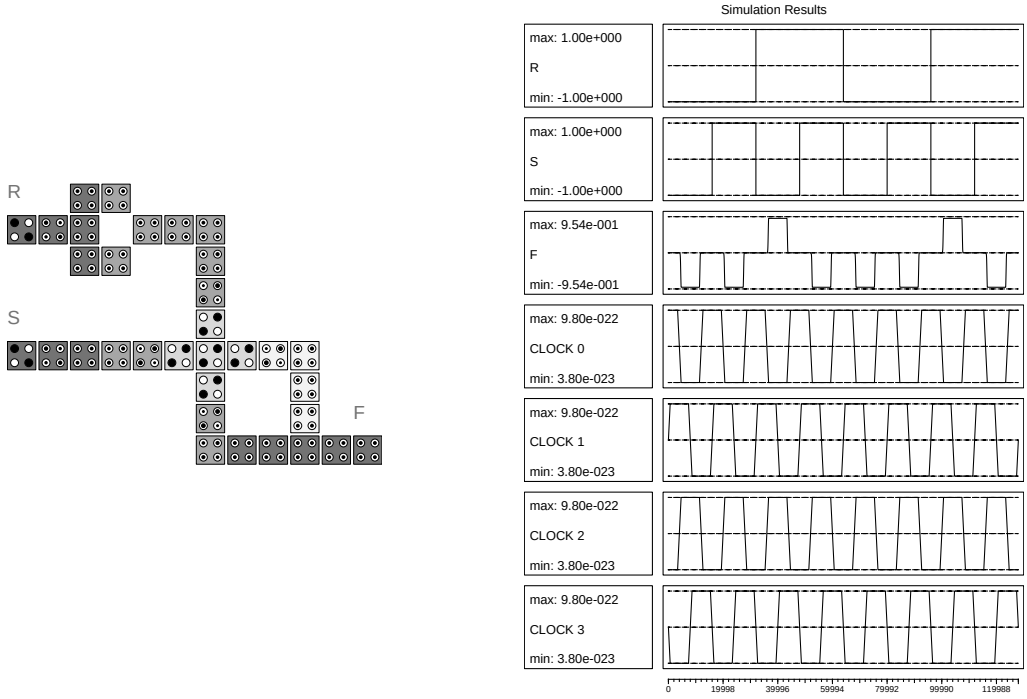


Figure 3.10: Layout of modified QCA-based SR flip-flop and its waveform. A SR flip-flop have two outputs F and $\bar{F}$, but only one output, F will be attached to input of full adder. So, an inverter which yields output $\bar{F}$ has been removed from the design.

The SR flip-flop has two different datas. (R=0, S=1) represents data 1, and (R=1, S=0) represents data 0. Both (R=0, S=0) and (R=1, S=1) are ignored, and the last output of the flip-flop remains as the current output. On the other hand, a SR flip-flop keeps current data until it receives the other data. Even though the SR flip-flop does not need a 'NULL' control signal, it has a hysteresis behavior. Therefore, SR flip-flops may substitute for NCL registers of the single full adder.

QCA-based NCL full adder has been designed by the multi-layer wire crossing method. The design consists of two parts, full adder and register. The Full adder is composed of two TH23 gates and TH34w2 gates, and the register is composed of six modified SR flip-flops. Note that the flip-flops are not part of the NCL full adder. They are being used as simple

substitutes to the NCL register in order to verify validity of the QCA-based NCL full adder. QCA-based NCL full adder and flip-flops are shown in Figure 3.11.

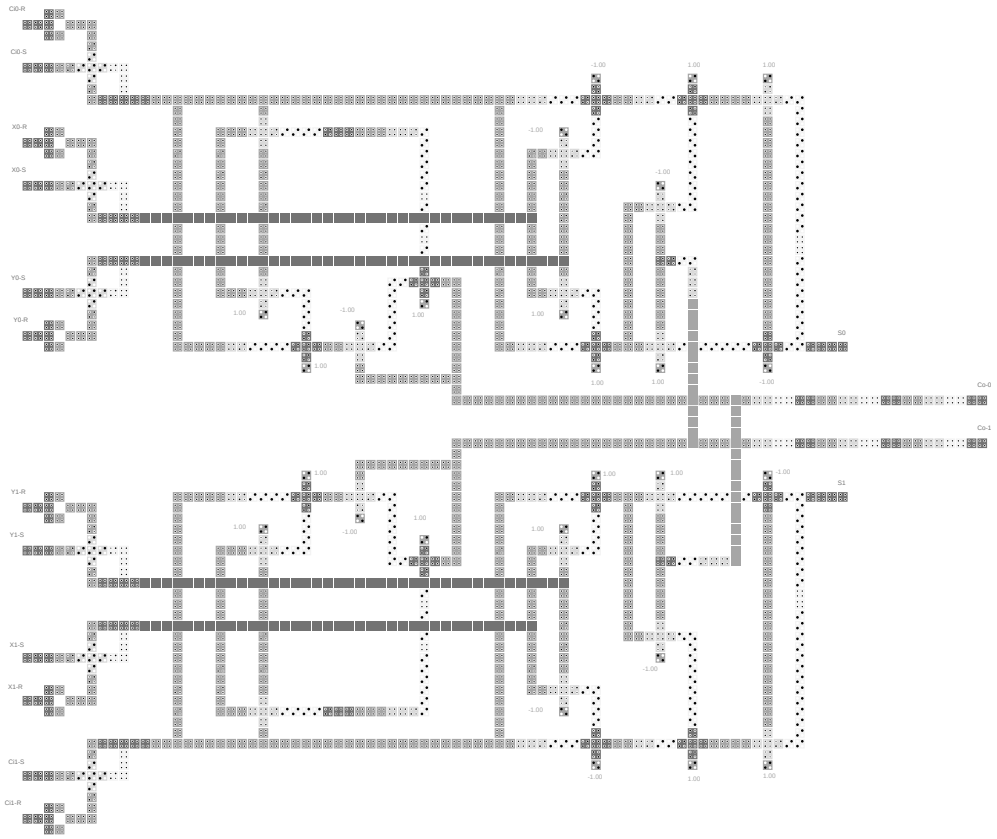


Figure 3.11: Layout of QCA-based NCL full adder and six flip-flops that substitute NCL register. Flip-flops generate input signals which represent hysteresis behavior. 1158 cells of 1350 total cells are used for full adder design within $1.9\ \mu m^2$ space.

Dual rail NCL full adder uses a combination of dual wires and the full adder switch between logic based data representation of DATA and a control representation of NULL. Synchronous full adder has three inputs X, Y, and Ci, but dual rail NCL full adder has six inputs X_0 and X_1 for X, Y_0 and Y_1, for Y and Ci_0 and Ci_1 for Ci. Each input of the NCL full adder is extended from SR flip-flop. This doubles the number of inputs. Therefore, the design in Figure 3.11 has twelve inputs X_0S, X_0R, X_1S, and X_1R for input X, Y_0S, Y_0R, Y_1S, and Y_1R for input Y, and Ci_0S, Ci_0R, Ci_1S, and Ci_1R for input Ci. Value of X depends on the combination of X_0 and X_1. For example, If X's current data is 0, then

X changes to data 1, only when X_0 changes to data 1 and X_1 changes to data 0. This dual rail encoding method had been discussed in Section 2.3. Two inputs of a SR flip-flop act as dual rail inputs for each wire of the dual rail. X_0 is switched to data 1, when X_0S has data 1 and X_0R has 0. Therefore, all inputs with the same capital letter correspond to each other. With the help of the truth table for full adder, corresponding dual rail data table and SR flip-flop data table is shown in Table 3.1.

Input																					Output					
SR Flip-Flop												Dual rail											Dual rail			
X0-S	X0-R	X1-S	X1-R	Y0-S	Y0-R	Y1-S	Y1-R	Ci0-S	Ci0-R	Ci1-S	Ci1-R	X0	X1	Y0	Y1	C0	C1	X	Y	C	Co	S	Co-0	Co-1	S0	S1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0
1	0	0	1	1	0	0	1	0	1	1	0	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1
1	0	0	1	0	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	0	0	1	1	0	0	1
1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1	0	1	0	1	1	1	0	0	1	1	0
0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0	1	0	1	0	0	0	1	1	0	0	1
0	1	1	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0	1	0	0	1	0	1	0	1	1	0	1	1	0	1	0	0	1	1	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	1	1	1	1	1	0	1	0	1
Initial Input																		Truth Table for Full Adder					Final Output			

Table 3.1: Truth table for full adder and its corresponding data table of SR flip-flop and dual rail. Initial input table will be applied to Input vector table of NCL full adder. Final output table will be used as the measure which determines validity of NCL full adder. Note that, not only dual rail, but also SR flip-flop transmits data by dual rail encoding method.

Inputs of the SR flip-flop should not be directly applied to input vector of NCL full adder, since NCL full adder requires NULL control signal to switch current data. On the other hand, even though NCL full adder receives another valid input data, it does not switch its current data without NULL control signal. Therefore, NULL control signal must be inserted between every input data. Figure 3.12 is a screen shot of input vector table for NCL full adder.

NULL control signal is repeatedly placed in every 10th column from column 0, and it remains during 5 complete clock cycles. For example, both (Ci_0R=1, Ci_0R=0) and (Ci_1R=1, Ci_1R=0) represent logic data 0. This means that the two corresponding inputs Ci_0 and Ci_1 have data 0. In the dual rail encoding scheme, (Ci_0=0, Ci_1=0) represents

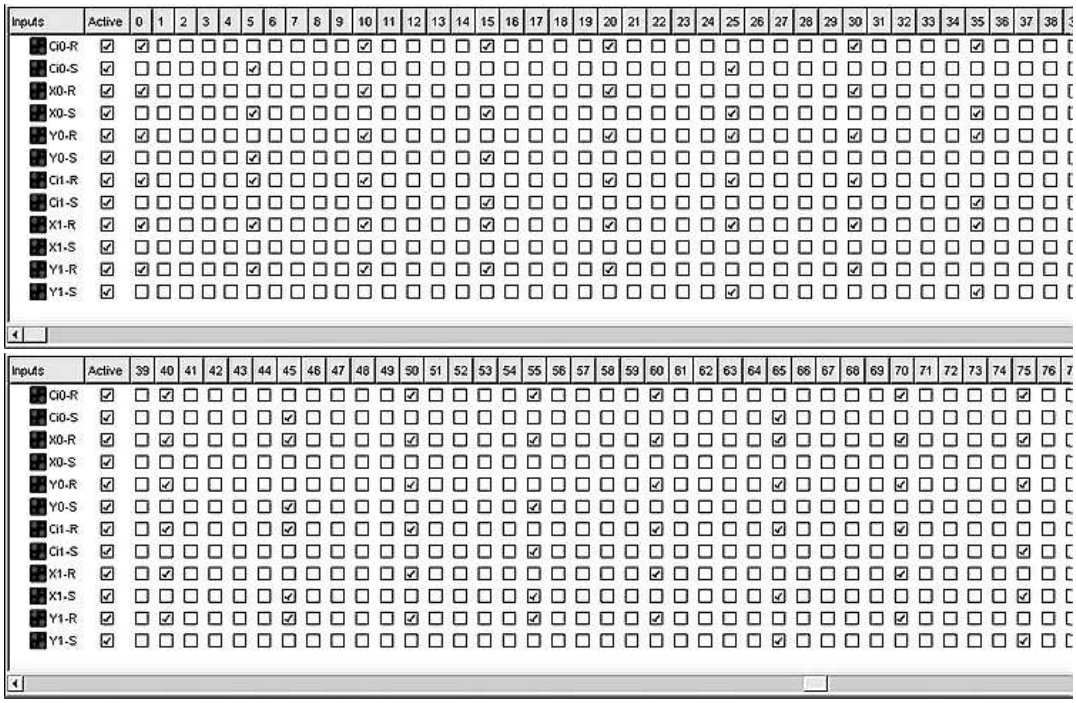


Figure 3.12: Screen shot of input vector table for NCL full adder.

NULL state. DATA signals appears every 10th column from column 5, and they are encoded in the same manner. Even though each blank column has the logic value 1, it does not have an effect on the data, because of NCL circuit's hysteresis behavior. On the other hand, NCL circuit ignores current data and keeps last data, until NULL control signal has been transmitted.

Result waveform is shown in Figure 3.13. NULL control signal is placed between the output signal. With NULL control signal, NCL full adder determines the validity of output without a reference to the time of evaluation. On the other hand, every data followed by NULL control signal is valid data. Therefore, NCL full adder is delay-insensitive. NCL full adder has 20 different clock zones, and SR flip-flop has 4 different clock zones. So, the Output has been delayed by 6 complete clock cycles. Result wave front of output is equivalent to final output table in Table 3.1, except NULL control signal which lies between the output data.

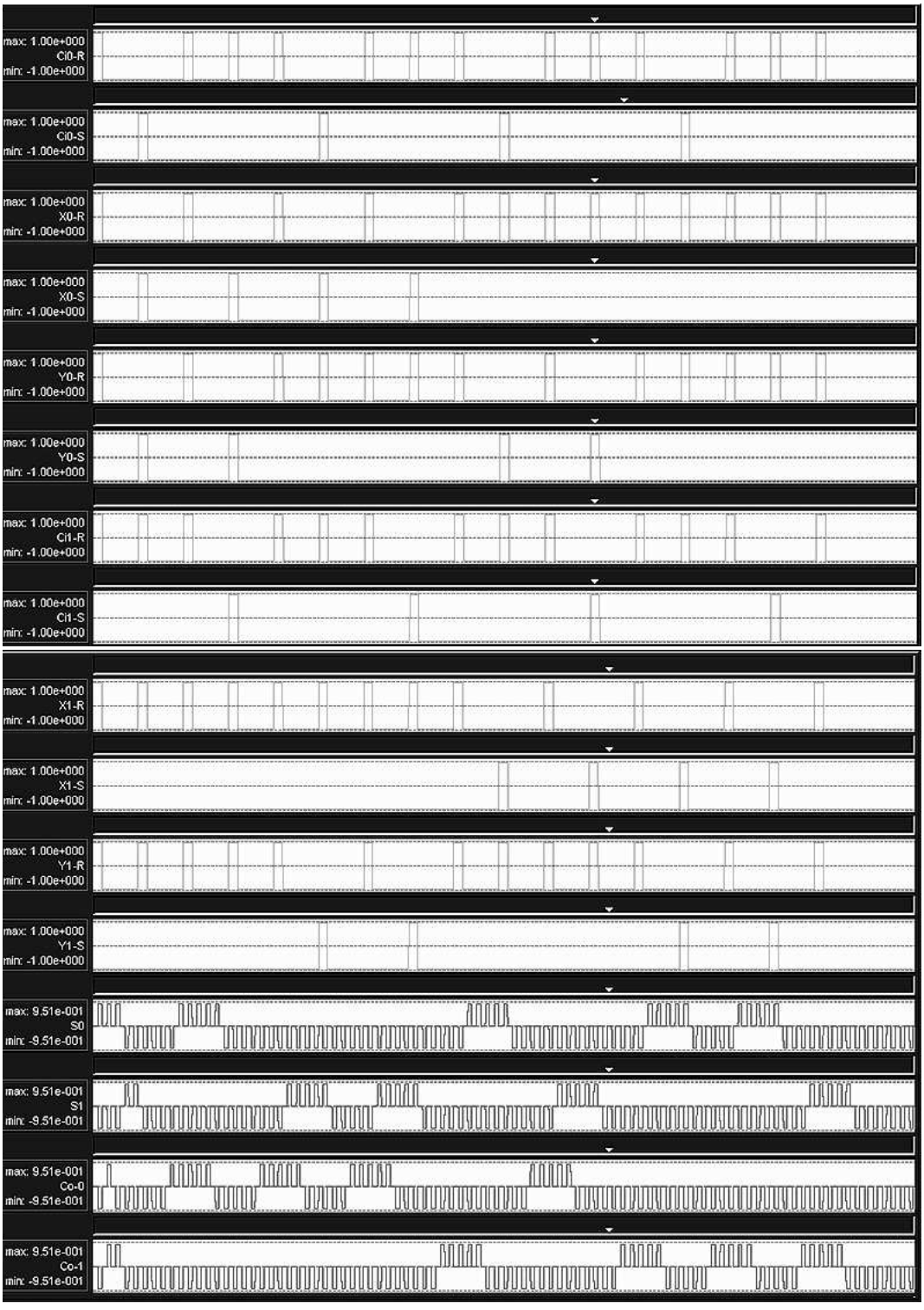


Figure 3.13: Result waveform of QCA-based NCL full adder.

3.3 Design Evaluation

NCL full adder consists of 1158 cells within $1.9\ \mu m^2$ space, but the synchronous full adder consists of just 165 cells within $0.21\ \mu m^2$ space. Approximately, the NCL full adder is 7 times bigger than the synchronous full adder. The NCL seems to be inefficient with a small size circuit. However, the NCL full adder will be quite efficient for a big size circuit. If n number of NCL full adders are used to develop n-bit full adder, NCL full adders can be located into a single clock zone, and they do not need any buffer for synchronization, because of their delay-insensitive property. In the same case, Synchronous n-bit full adder needs $\sum(n-1)$ number of input buffers and $\sum(n-1)$ number of output buffers to synchronize. So, NCL is more suitable to a big size circuit design. For example, in order to design a 4-bit synchronous full adder, additional six input buffers and six output buffers are necessary for synchronization reason, but 4-bit NCL full adder can be designed with four 1-bit full adders. Addition of delay buffers may cause serious loss in both cell count and clock zone space. Figure 3.14 describes 4-bit full adder example.

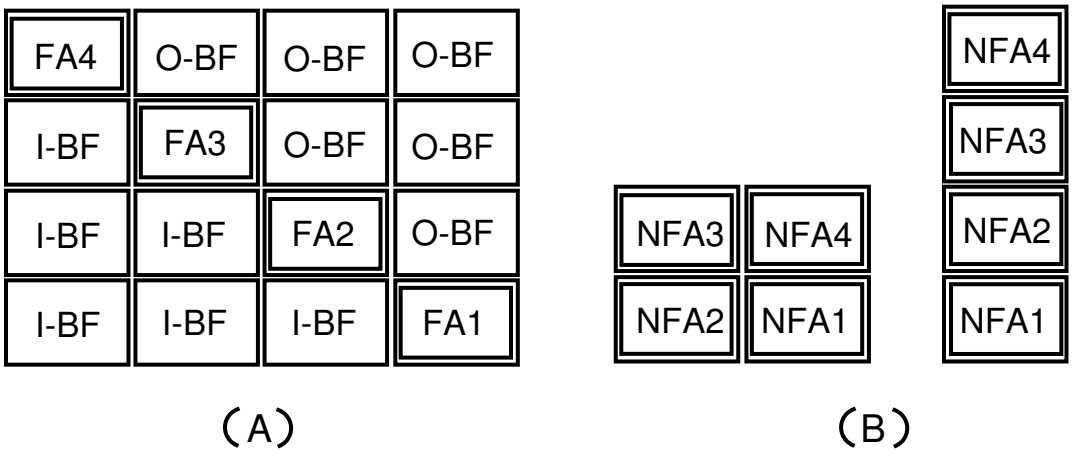


Figure 3.14: 4-bit full adder example. (A) Synchronous 4-bit full adder. Six input delay buffers and six output delay buffers are used with four 1-bit synchronous full adder. (B) NCL 4-bit full adder. Only four NCL 1-bit full adders are used. Shape of 4-bit full adder is either rectangular or linear, since each NCL full adder can be placed anywhere in the given area.

An output buffer is nothing but a clocked array for result output. Since the width be-

tween input and output of synchronous full adder is 23, cell count of an output buffer is also 23. This is minimized cell count for output buffer. An input buffer has two clocked arrays, and length of each array is also 23. Cell count of an input array, therefore 46. That is also the minimized cell count for the input buffer. Obviously, Carry-out wire of current adder is only an extension of the carry-in wire of the following adder. This extension wire is not a delay buffer, but a simple extension wire between neighboring adders. So, there are $n - i$ extension wires in n-bit adder. If the cell count of an extension wire is e, then the total cell count of synchronous n-bit adder can be calculated by :

$$165n + \sum_{i=1}^{n} 23(i-1) + \sum_{i=1}^{n} 46(i-1) + e(n-1)$$

$$= 165n + \sum_{i=1}^{n} 69(i-1) + e(n-1) \tag{3.1}$$

Synchronous 1-bit full adder lies on $0.21\ \mu m^2$ space. Both Adders and buffers occupy the same size space like Figure 3.14(A). Total used space of Synchronous n-bit adder is: :

$$n^2 * 0.21\ \mu m^2 \tag{3.2}$$

Total cell count of NCL n-bit adder is simple. 1158 cells are used in each full adder. NCL n-bit adder consists of n number of NCL full adders. Even though NCL n-bit adder does not require any additional buffers to synchronize, it still needs extension wires between the two neighboring adders. Since the NCL full adder employs a dual rail encoding scheme, two extension wires should be inserted between neighboring adders. So, total cell count of extension wires is shown in terms of $2e' * (n-1)$, where e' is cell count of each extension wire. Total cell count of n-bit full adder is:

$$1158 * n + 2e' * (n-1). \tag{3.3}$$

Total used area can be calculated in same manner. Size of NCL full adder is $1.9\ \mu m^2$.

Therefore, total used area of NCL n-bit adder is:

$$n * 1.9\ \mu m^2. \tag{3.4}$$

Cell count of extension wire e and e' are not fixed. Since the size of e and e' are mainly dependant on the layout of the circuit, they can be reduced and minimized by proper floor planning. For example, if all the carry-out outputs are directly connected to the carry-in input of next digit adder without an extension wire, value of both e and e' will be 0. With this ideal condition, we can remove unknown value e and e' from formula 3.1 and 3.3. Table 3.2 has been developed without both e and e'.

FA: Synchronous n-bit full adder
NFA: NCL n-bit full adder

Cell Count: cells
Used Area: μm ^{2}

		n=4	n=8	n=16	n=32	n=64	n=128
Cell Count	FA	1,074	3,252	10,920	39,524	149,664	581,952
	NFA	4,632	9,264	18,528	37,056	74,112	148,224
Used Area	FA	3.36	13.44	53.76	215.04	860.16	3440.64
	NFA	7.6	15.2	30.4	60.8	121.6	243.2

Table 3.2: Total used cell count and used space for two different versions of n-bit full adder.

Synchronous full adder seems to be more efficient when n is below 16. However, synchronous full adder wastes more space which is occupied by delay buffers. Insertion of a delay buffer reduces circuit density. For example, when n=32, total cell count of both synchronous full adder and NCL full adder are almost same, but the used space of synchronous full adder is 3.5 times larger than used space of NCL full adder. When n goes over 30, NCL full adder is more efficient in both cell count and used space aspects. NCL

seems to be more suitable for a big size system. Assume that there is a possible way to reduce the size of a buffer. However, this may not reduce cell count, because the cell count of both input and output buffers is already minimized (i.e., 23 cell for output buffer and 46 cell for input buffer). It makes global floorplanning more complex, since space reduction may change each buffer's shape. No complex floorplanning is required for the NCL full adder. Since NCL full adder is delay insensitive, each can be located within any spot in a given area. The overall timing of QCA circuits is mainly dependent upon its layout [16]. Asynchronous n-bit adder is an extreme case of this timing constraint that is referred to as the "Layout=Timing" problem which has been discussed in the previous chapter. QCA-based NCL devices are free from this constraint.

CHAPTER 4

CONCLUSION

The concept of clocking for QCA, referred to as the four-phase clocking, is widely used and it is possible to design QCA circuits while ensuring each QCA cell is powered and the information is processed and forwarded in a timely manner. However, inherited characteristics of QCA, such as the way to hold state, the way to synchronize data flow, and the way to power QCA cells, make the design of QCA circuits quite different from VLSI and introduces a variety of new design challenges, and the most severe challenges are due to the fact that the overall timing of a QCA circuit is mainly dependent upon its lay out. QCA suffers from this strict timing constraints referred to as the ”layout=timing” problem. Applying NCL asynchronous methodology to QCA paradigm eliminates this problem, and it also provides many advantages such as easier floorplanning, increased circuit density, reduced complexity, decreased cell count, and reduced space.

In order to realize CMOS-based asynchronous methodology in QCA, NCL, one of the CMOS-based asynchronous methodologies has been used. THmn gates are basic building units for NCL circuit designs, and they have threshold and hysteresis behaviors. These behaviors are significant since they provide delay-insensitivity for the NCL circuits. In order to synchronize, NCL circuits refer to control data represent NULL rather than time. Therefore, they do not need delay buffers to synchronize. QCA-based TH23 and TH34W2 gates have been developed to optimize QCA asynchronous methodology, and they also have the same properties that CMOS based THmn gates usually have. Basically, ”Layout=Timing” constraint forces synchronous QCA circuits to waste many buffers for synchronization purpose. Applying NCL to QCA eliminates the ”Layout=Timing” problem as well as the delay buffers from QCA circuits. QCA-based NCL 1-bit full adder has been developed with two TH23 gates and two TH34W2 gates. In order to verify validity of the QCA-based NCL

full adder, six modified SR flip-flops that substitute the NCL register are added into the NCL full adder design. Simulation result of the QCA-based NCL full adder indicates that NCL can be realized in the QCA paradigm. QCA-based NCL full adder has succeeded two important behaviors such as hesteresis behavior and threshold behavior from TH23 and TH34W2 gates. NCL circuits do not need buffers to synchronize, since they are delay-insensitive. Large scale synchronous QCA circuits may have more number of buffers inside. An extreme example is shown in TABLE 3.2. From the table, we realize the NCL is more suitable for a large scale circuit design.

REFERENCES

[1] International Technology Roadmap for Semiconductors, "International Technology Roadmap for Semiconductors (ITRS) 2004," *http://public.itrs.net,* 2004.

[2] C.S. Lent, D. Tougaw and W. Porod, "Quantum Cellular Automata," Nanotechnology, Vol. 4, No. 49, 1993

[3] J. Timler and C. Lent, "Power Gain and Dissipation in Quantum-Dot Cellular Automata," Journal of Applied Physics, Vol 91, pp 823-831, 2002.

[4] A. Orlov, R. Kummamur, R. Ramasubramaniam, C. Lent, G. Bernstein and G. Snider, "Clocked Quantum-Dot Cellular Automata Shift Register," Surface Science, Vol 532, pp.1193-1198, 2003.

[5] A. Orlov, R. Kummamuru, R. Ramasubramaniam, C. Lent, G. Bernstein and G. Snider, "A Two-Stage Shift Register for Clocked Quantum-Dot Cellular Automata," Journal of Nanoscience and Nanotechnology, Vol 2, pp 351-355, 2002.

[6] G. Snider , A. Orlov, I. Amlani, G. Bernstein, C. Lent, J. Merz and W. Porod, "Quantum-Dot Cellular Automata: Line and Majority Logic Gate," Japanese Journal of Applied Physics Part 1- Regular Papers Short Notes & Review Paters, Vol 38, pp 7227-7229, 1999.

[7] G. Snider , A. Orlov, I. Amlani, G. Bernstein, C. Lent, J. Merz and W. Porod, "Quantum-Dot Cellular Automata," Microelectronic Engineering, Vol 47, pp 261-263, 1999.

[8] G. Bernstein, I. Amlani, A. Orlov, C. Lent and G. Snider, "Observation of Switching in a Quantum-Dot Cellular Automata Cell," Nanotechnology, Vol 10, pp 166-173, 1999.

[9] G. Toth and C. Lent, "Quantum Computing with Quantum-Dot Cellular Automata," Physical Review-A, Vol 63, Art, No 052315-, 2001.

[10] G. Toth and C. Lent, "Quasiadiabatic Switching for Metal-Island Quantum-Dot Cellular Automata," Journal of Applied Physics, Vol 85, pp 2977-2984, 1999.

[11] I. Amlani, A. Orlov, G. Toth, G. Bernstein, C. Lent and G. Snider, "Digital Logic Gate Using Quantum-Dot Cellular Automata," Science, Vol 284, pp 289-291, 1999.

[12] R. Kummamuru, J. Timler, G. Toth, C. Lent, R. Ramasubramaniam, A. Orlov, G. Bernstein and G. Snider, "Power Gain in a Quantum-Dot Cellular Automata Latch," Applied Physics Letters, Vol 81, pp 1332-1334, 2002.

[13] A. Orlov, I. Amlani, G. Toth, G. Bernstein and G. Snider, "Experimental Demonstration of a Binary Wire for Quantum-Dot Cellular Automata," Applied Physics Letters, Vol 74, pp 2875-2877, 1999.

[14] M.T. Niemier, P.M. Kogge, "Exploring and Exploiting Wire-Level Pipelining in Emerging Technologies,", ISCA, June 2001.

[15] D.A. Antonelli, D.Z. Chen, T.J. Dysart, X.S. Hu, A.B. Kahng, P.M. Kogge, R.C. Murphy and M.T. Niemier, "Quantum-Dot Cellular Automata (QCA) Circuit Partitioning: Problem Modeling and Solutions," The 41st Design Automation Conference (DAC), June 2004.

[16] M. Niemier and P. Kogge, "Problems in Designing with QCAs: Layout Equals Timing," International Journal of Circuit Theory and Applications, Vol 29, pp 49-62, 2001.

[17] QCADesigner, http://www.qcadesigner.ca

[18] W. Porod, "Quantum-Dot Devices and Quantum-Dot Cellular Automata," International Journal of Bifurcation and Chaos, Vol 7, pp 2199-2218, 1997.

[19] W. Porod, "Quantum-Dot Devices and Quantum-Dot Cellular Automata," Journal of the Franklin Institute-Engineering and Applied Mathematics, Vol 334B, pp 1147-1175, 1997.

[20] W. Porod, C. Lent, G. Bernstein, A. Orlov, I. Amlani, G. Snider and J. Merz, "Quantum-Dot Cellular Automata: Computing with Coupled Quantum Dots," International Journal of Electronics, Vol 86, pp 549-590, 1999.

[21] K.M.Fant and S.A.Brandt, "NULL Convention Logic: A complete and consistent logic for asynchronous digit circuit synthesis," in Int.Conf.Application Specific Systems,Architectures,Processors, pp 261-273. 1996.

[22] S.K. Bandapati, S.C. Smith and M. Choi, "Design and Characterization of Null Convention Self-Timed Multipliers," IEEE Design and Test of Computers, Nov-Dec 2003.

[23] J.S.Yuan and W.Kuang, "Teaching Asynchronous Design in Digital Integrated Circuits," IEEE Transactions on Education, Vol. 47, No.3, pp397-404 2004.

[24] M. Choi and N. Park, "Locally Synchronous, Grobally Asynchronous Design for Quantum-Dot Cellular Automata (LSGA QCA)," IEEE Conference on Nanotechnology, pp 275-278, July 11-15th, 2005.

[25] M.T. Niemier, "Designing Digital System in Quantum Cellular Automata," MS CSE Thesis, Univ of Notre Dame, Apr 2000.

[26] K. Walus, T. Dysart, G. Jullien and R. Budiman, "QCADesigner: A Rapid Design and Simulation Tool for Quantum-Dot Cellular Automata," IEEE Transactions on Nanotechnology, Vol 3, pp. 31-, 2004.

[27] S.C.Smith, "Gate and Throughput Optimizations for NULL Convetion Self-Timed Digital Circuits" Ph.D.Dissertation, School of Electrical Engineering and Computer Science, University of Central Florida, 2001.

[28] C.S. Lent and D. Tougaw, ”Logical Devices implemented Using Quantum Cellualar Automata,” J.Appl.Phys, Vol. 75, No. 3, 1994

VITA

Myungsu Choi

Candidate for the Degree of

Master of Science

Thesis: A STUDY ON A QUANTUM-DOT AUTOMATA BASED ASYNCHRONOUS CIRCUIT DESIGN

Major Field: Computer Science

Biographical:

Personal Data: Born in Seoul, Korea, on March 25, 1974, the son of Jinsung and Daeja Choi.

Education: Graduated from Daesung High School, Seoul, Korea in January 1993; received Bachelor of Science degree in Computer Science from Oklahoma State University, Stillwater, Oklahoma in August 2003, respectively. Completed the requirements for the Master of Science degree at Oklahoma State University in December 2005.

Experience: Employed as a Sales Representative; Carrefour Korea, Ilsan, Korea, 1995-1997.

Professional Memberships: IEEE, Golden Key National Honor Society.

Name: Myungsu Choi Date of Degree: December, 2005

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: A STUDY ON A QUANTUM-DOT AUTOMATA BASED ASYNCHRONOUS CIRCUIT DESIGN

Pages in Study: 42 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study: Quantum-Dot Cellular Automata (QCA) is one of the most emerging technologies. It has been introduced as a possible alternative to CMOS (Complementary Metal Oxide Semiconductor). The main objective of this thesis work is to apply CMOS based asynchronous methodology to QCA paradigm. The concept of QCA clocking is quite different from that of CMOS clocking. In the QCA, the clock is used as a control signal in stead of a data signal. Inherited characteristics of QCA, such as the way to hold state, the way to synchronize data flows, and the way to power QCA cells make the design of QCA circuits quite different from VLSI. This also introduces a variety of new challenges to the design and the greatest challenges are due to the fact that the overall timing of a QCA circuit is mainly dependent on its layout. In order to eliminates "Layout=Timing" constraint from QCA circuits, delay-insensitive data encoding scheme such as NCL(Null Convention Logic) are applied to the QCA paradigm. Proposed QCA-based NCL full adder illustrates how NCL prevents timing constraint from QCA circuits.

Findings and Conclusions:Basically, "Layout=Timing" constraint forces synchronous QCA circuits to waste many buffers for synchronization purpose. Applying NCL to QCA eliminates the "Layout=Timing" problem as well as the delay buffers from QCA circuits. The NCL provides numerous advantages such as easier floorplanning, increased circuit density, decreased QCA cell count, and reduced circuit complexity. Especially, NCL is more suitable for a large scale circuit design.

ADVISOR'S APPROVAL: Nohpill Park