A Thesis

entitled

Modeling and Simulation of Altera Logic Array Block using Quantum-Dot Cellular Automata

by

Rohan Kapkar

Submitted to Graduate Faculty as partial fulfillment of the requirements of the Master of Science Degree in Electrical Engineering

The University of Toledo
August 2011

An Abstract of

Modeling and Simulation of Altera Logic Array Block using Quantum-Dot Cellular Automata

by

Rohan Kapkar

Submitted to the Graduate Faculty as partial fulfillment of the requirements for

The Master of Science Degree in Electrical Engineering

The University of Toledo

August 2011

According to Moore's law, the number of transistors that can be placed on an integrated circuit doubles approximately every 18 months. Recent advancements in CMOS technology have led to the implementation of new computational designs with extremely small size and high device density. However, CMOS technology is reaching its limits. Currently, scientists and researchers are exploring various new technologies that may replace CMOS when its limit is reached in the future. Quantum-dot Cellular Automata (QCA) is one of the novel nanotechnologies that is being considered as a possible replacement for CMOS. It has great potential for very dense memory and low power logic based on single electron effects in quantum dots and molecules. QCA relies on novel design concepts to exploit new physical phenomena such as coulombic interactions and implement unique paradigms such as memory-in-motion and processing-by-wire. This thesis presents a first Altera implementation of Stratix Logic Array Block (LAB) architecture using QCA technology along with simulation results. The design is modeled and simulated using QCADesigner software. A novel Logic Array Block in QCA is developed by implementing Look-Up Tables (LUTs). In the design of the LUT, QCADesigner software is utilized to design and simulate a four-to-sixteen decoder, 16-bit memory, and an output circuit to implement a LUT. The LUT design comprises of QCA memory cells with low read latency. The proposed LAB includes Look-up Tables, D flip-flops, multiplexers and various logic gates that are designed and tested using QCADesigner. The LAB designed in this work comprises of 25,000 QCA cells approximately and has a latency of 17 clock cycles. The LAB design is compared with the previous designs of Configurable Logic Blocks. The results show that the proposed LAB provides high degree of performance, simplicity, and optimization of design area compared to the other designs.

This thesis also presents an implementation of a novel Programmable Switch Matrix using QCA technology.

Acknowledgements

I would like to thank Dr. Niamat for giving me an opportunity to work under his leadership and guidance. I could not have completed this work without the mentoring of my thesis advisor. I selected this research topic because what I learnt from him about the topic. He is persuasive in new technologies and has a wonderful fascination about the nano-technology. I am truly indebted to him for fostering the same pursuit and fascinations in me and, of course his guidance and his advice during the years as his student. I would also like to extend my regards to Dr. Junghwan Kim and Dr. Hong Wang for being the committee members.

This thesis would not have been possible without love, support, and encouragement I received from my parents, brother, and sister. I whole heartedly thank my parents who were very supportive in me pursuing higher education. I do not have words to adequately describe my deep gratitude for all they have provided me; I hope to show them my appreciation in the years to come. I would also like to thank Priyanka Potnis who has been a good friend, without whom this work would have been impossible. I would also thank all my friends Aloke, Sanket, and Amogh who stood by me all the time and being helpful whenever needed.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction to Research

Moore's law dictates that the number of devices integrated on a single die doubles every 18 months, and has, since 1960s, governed the manufacturing of integrated circuits. Studies indicate that as early as 2015, the transistor may hit physical scaling limits that inhibit this aggressive packing of devices. Although recent advances have given significantly more life to this device model, and some in the field contend that molecular transistors may be feasible in the near future, other devices may be better suited to the computing environment of the future. CMOS (Complementary Metal Oxide Semiconductor) technology has managed advancement of microelectronics to provide high densities, high speed and low power dissipation. The reduction of size has increased speed and density but it reduced the circuit performance. This is because of the fundamental physical limits of CMOS and more because of the leakage current. Recent advancements in CMOS technology have led to the implementation of new computational designs with extremely small size and high device density. However, CMOS technology is reaching its limits. Currently, scientists and researchers are exploring various new technologies that may replace CMOS when its limit is reached in the future. There has been substantial research in need of an alternative technology to overcome the roadblock in the future of scaling of CMOS devices.

Quantum-dot cellular automata (QCA) is an alternative to silicon/CMOS based designs. It was first proposed in 1993 by Lent, and fabricated in 1997. In QCA, the logic states are represented depending upon the position of electron rather than in voltage levels as in CMOS technology. This characteristic of QCA eliminates the problem of leakage current in the CMOS technology. Quantum dots are nanostructures created from semi conductive materials. These structures can be fabricated as quantum wells. They exhibit energy effects. A dot can be visualized as well. Once electrons are trapped inside the dot, the electrons require higher energy to escape from one dot to another. A quantum dot cellular automaton is a Novel technology that attempts to create general computational functionality at the nanoscale by controlling the position of single electron [1], [2]. An idealized QCA cell/device can be viewed as a set of four change containers or "dots", positioned at the corners of a square [3],[4].

Field-programmable gate arrays (FPGAs) [5], [6] has provided a great the trade-off point between the flexibility and low cost of software and the performance of hardware. FPGAs can be programmed by two ways: programmable logic and programmable interconnect. FPGAs represent a good application for QCA technology [7]. Their homogeneous structure is well suited to fabrication at the nanoscale. Also the general-purpose nature of FPGAs could allow many applications to be implemented using QCA. An FPGA could be made with programmable interconnect and fixed logic blocks. Alternatively, an FPGA could be made with fixed interconnect and configurable logic blocks. In both cases, the fixed/configurable blocks are placed in a two-dimensional grid. In the former type (Figure 1-1(a)), the fixed logic blocks are implemented with fixed combinational circuits such as NAND-based circuits or special hardware blocks. Vertical and horizontal I/O wires are placed between the fixed logic blocks. The inputs and outputs of a block (not shown in Figure 1-1) are connected to the wires surrounding the block. At intersections of vertical and horizontal wires, the wires can be connected together through programmable interconnect to perform the desired functions. In the latter type, the configurable logic blocks are implemented with LUT-based circuits or other configurable circuits. Figure 1-1(b) shows one example of the FPGA arrangements; the input(s) and output(s) on each side of the block are connected between the block and the nearest block. The purpose of a LUT is to look-up the corresponding output value from user-configured truth table based on address selection (inputs). Previous work on QCA-based FPGAs [7], [8] has focused on programmable interconnect. In contrast, this work focuses on programmable logic. The goal of this work is to design and evaluate programmable logic using QCA. In particular, this work reviews the design and layout of a QCA-based Look-up table. The design has been simulated using QCADesigner [9], [10]. Many copies of this LUT can be tiled together to form a complete QCA-based LAB which can be used to build FPGA device.

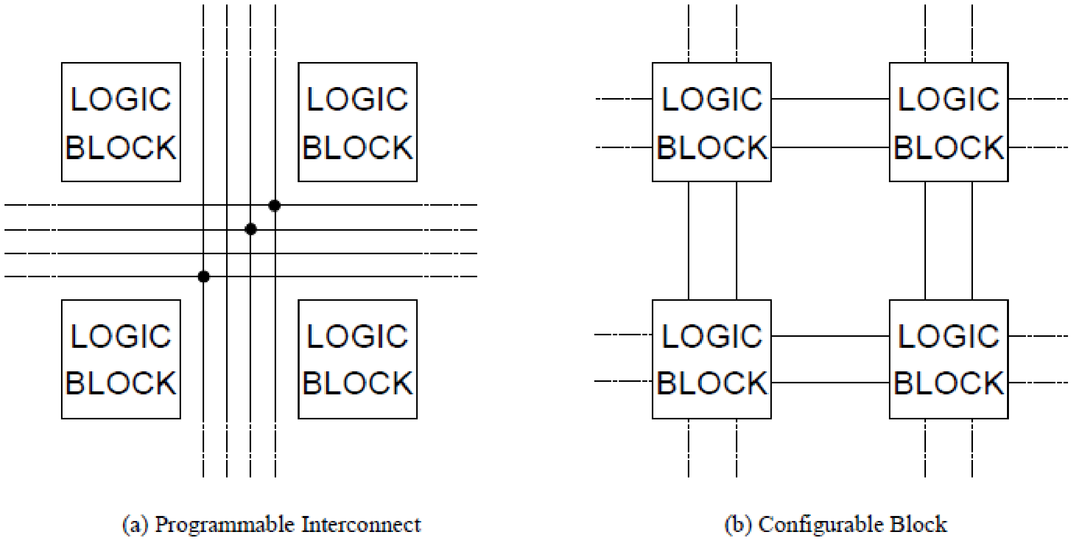

Figure 1-1: FPGA Architecture

An overview of Quantum-dot Cellular Automata (QCA) is given in Section 1.1.1.

1.1.1 Quantum-Dot Cellular Automata

Quantum-dot cellular automata technology is an emerging technology and possibly the replacement for transistor circuits in digital systems. QCA has gained significant popularity in recent years. This is mainly due to rising interest in creating computing devices at the nanoscale. At such scales, QCA has an inherent advantage over conventional integrated circuit technologies (such as CMOS) in that its performance increases as feature sizes are reduced, whereas CMOS exhibits decreasing performance with decreasing feature sizes. At this time, it is unclear whether or not this technology will replace such a firmly embedded technology as CMOS, but investigations into modeling and design have demonstrated that QCA has many powerful features some of which are not available in CMOS. Unlike complementary metal oxide semiconductor (CMOS) technology, QCA uses the positions of electrons in quantum dots to represent binary values '0' and '1'. QCA technology uses the Coulombic interaction to determine the positions of electrons, while CMOS uses the current to determine the potential (voltage) levels. The advantages of using QCA technology are smaller circuit size, higher clock frequency, and less power consumption. The basic building block of a QCA circuit is a QCA cell consisting of four quantum dots, containing two mobile electrons. These electrons tunnel within the cell and occupy diagonal positions of the cell due to coulombic interaction. The basic logic element made using the QCA cell is majority gate. This majority gate can be used as an AND gate or an OR gate. QCADesigner is the software which is used to design QCA circuits. It allows the QCA circuit designs to be implemented and simulated. Despite of the advantages of QCA, there are few limitations to this technology. Quantum-dot cellular automata experiments are carried out at cryogenic temperatures. For molecular QCA, the material should be in its purest form. There is a need of technology to implement QCA in molecular form at room temperatures. Chapter 2 will give more detailed information about the QCA.

1.2 Motivation

Quantum-dot cellular automata are novel devices with characteristics that differ substantially from CMOS. QCA takes advantage of the characteristics of the device to achieve memory densities that exceed the SIA roadmap for CMOS [11]. QCA architectures have low power dissipation. QCA takes less area for design than CMOS. FPGAs represent a good application for QCA technology [7]. Field-programmable gate arrays (FPGAs) have provided a great trade-off point between the flexibility and low cost of software and the performance of hardware. Their homogeneous structure is well suited to fabrication at the nanoscale level. Also, the reconfigurable nature of FPGAs could allow many applications to be implemented using these nano QCA devices.

In the past, different QCA architectures based on programmable interconnect and programmable logic has been presented in [7], [8], [12], [13], [14]. The goal of this work is to design and evaluate programmable logic using QCA. In particular, this research is about the design and implementation of a QCA-based Altera Stratix Logic Array Block. The design has been simulated using QCADesigner [9]. Previous work on designing Configurable Logic Block using QCA was focused on designing Xilinx Virtex family FPGAs. This is the first attempt to design a logic array block based on Altera Stratix family FPGA.

1.3 Proposed work

This work presents the design and simulation of a LUT-based Logic Array Block (LAB) that uses QCA technology. Each LUT has a 4-to-16 decoder, sixteen-bit memory and the output circuit. It has four inputs and one output. QCADesigner [10] is used for layout and simulation of the entire design. For simplicity, the focus is on a single LAB component rather than the whole FPGA architecture (a cellular array of LABs and interconnects together). The design has not been fabricated. However, the simulation results show that the LAB is working properly to meet expected results. QCADesigner is used to implement Stratix LAB in QCA technology using QCA basic blocks. This research presents a first Altera implementation of Stratix Logic Array Block (LAB) architecture using QCA technology along with simulation results. The design is modeled and simulated using QCADesigner software. This design provides high degree of performance, simplicity, and optimization of design area. The research also presents a design of interconnect switch matrix using the QCA technology.

1.4 Previous Work

Previous work on QCA-based FPGAs has focused on programmable interconnect [7], [8]. Design in [7] discusses a development of interconnect with flexible routing. The design uses pre-made circuit for all possible routing at interconnects. At the site of interconnects, QCA cells in the unclocked zone are assumed to be an open circuit between point A and point B. For instance, there is no connection between horizontal and vertical wires at interconnects. Similarly, QCA cells in the clocked zone are operating in the normal mode (connecting wires). The programmable interconnect in [8] uses a QCA 4-Diamond circuit which has a similar function to that of a pass transistor in CMOS technology. The interconnect design has four diamond-shaped circuits in a two-by-two grid. Upper-left, upper-right, and lower-left quadrants have the interconnect circuit, while lower-right quadrant has the logic block which consists of one fixed NAND gate. The interconnection design mainly uses unidirectional wire routing. This uses similar clocking methods (unclocked and clocked zones) to those in [7].

The authors in these papers mentioned above use programmable interconnect but fixed logic, which is the first type of FPGA architecture represented in Figure 1-1(a). In contrast, the QCA-based FPGA design presented here is believed to be the first design using configurable logic blocks (LUT-Based with memory) with fixed interconnect. This design targets the type of FPGA architecture represented in Figure 1-1(b). Notice that [7] and [9] do not use memory elements for configuration. There are numerous sources for

QCA-based memory designs. Only a few of them are discussed here. The authors in [15] present a memory with parallel write and parallel read with Row Select, Input, Write/Read, and Output signals. Because QCA is a pipelined technology, it uses an internal loop to preserve the value of the memory. An array of four one-bit memory cells is constructed to create 1x4 memory with a two-to-four decoder, a top-level shared Write/Read signal, a shared Input Signal, and a serial output. This design is shown in Figure 1-2. The design is scalable to create a larger array memory or a two-dimensional grid memory. This design is good for a small amount of memory. However, it may incur a larger delay for a large memory design (e.g., a 16-bit memory). The outputs are combined in a serial chain of OR gates, which can produce a long latency at the outputs. In contrast, the memory presented in this work uses a tree structure of OR gates. In another memory design, a memory with two or more bits can be developed by using a longer internal memory loop to hold the contents of the memory. Thus, it has higher memory density than [15], which uses a separate loop for each bit. The authors in [16], [17], [18], [19] use this memory design as a fundamental memory design, but each of them has different memory architecture for read and write operations. The authors in [19] use primitive QCA elements, each of which occupies a tile of 5x5 QCA cells. These include such primitives as a binary wire, a fan-out element, a majority gate, and an AND gate. The primitives are then used to make up logical devices such as a counter, comparator, register, latch, and so on. The memory in the design has 16-bit long loop to hold the contents of the memory. In order to read or write certain bit of the memory, the counter and comparator are used to compare between the current address and the desired address. The loop rotates until it has arrived at the desired address. At that point, read or write operation occurs. A long memory loop is good for compact memory design, but at the expense of the counter and comparator. Another disadvantage is that the loop takes a time to rotate in order to arrive at the desired address.

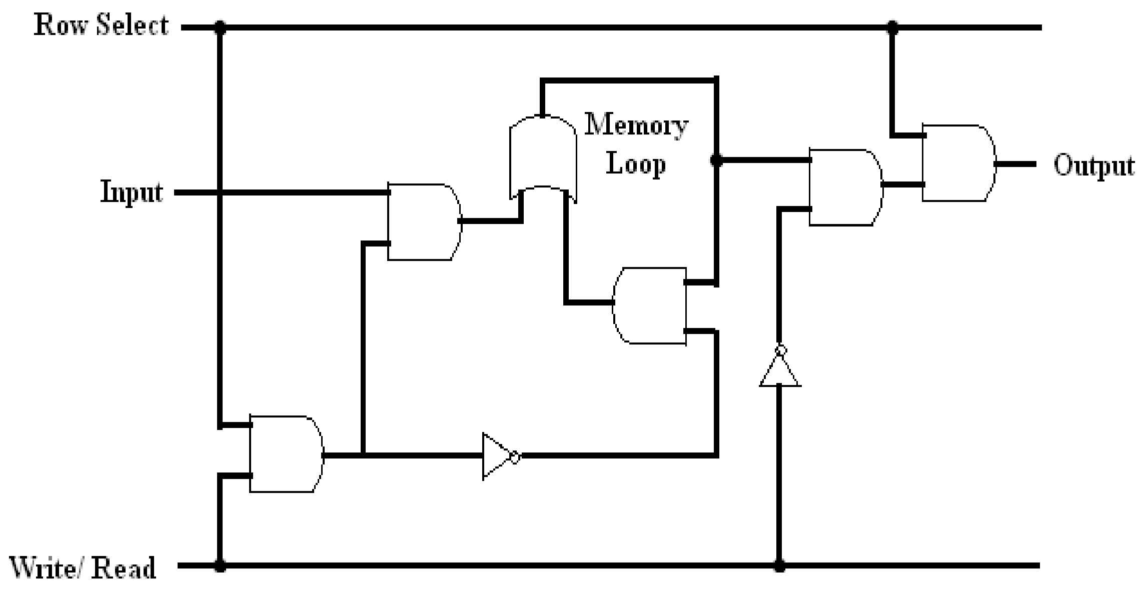

Figure 1-2: QCA Memory Cell

There is a spiral 12-bit memory loop with different write/read circuitry in [18]. The authors presented an H-Memory design with four Memory Macro blocks in a two-dimensional grid with a central routing node at the center of the design. The design is scalable to create bigger memory by using a tree of H-Memory. H-Memory routes a data packet (8-bit address, 1-bit op code, and 12-bit data, in that order) to go through a sequence of routing nodes until it reaches desired memory location to write or read the value of that memory. The data packet technique is very similar to the conventional computer architecture for memory accessing. This design is not well adapted for the LUT-based design presented here. The output of a LUT must continuously respond to the value at the LUT inputs rather than to arriving packets. The memory design in [17] describes tile-based serial memory. It uses three types of tiles: an input tile, an internal memory tile and an output tile. A number of internal tiles (variable N) can be stacked together vertically to implement M-word by N-bit memory. This memory design is almost equivalent to a long memory loop. However, it uses unconventional QCA clocking scheme for faster memory accessing. This design is a serial for both reading and writing. A LUT requires parallel read. The main disadvantage in [19] and [17] is that, in the read mode, it takes time to rotate the memory loop to arrive at the desired address to read the value in that memory location. The maximum time for read latency is N-1 cycles, where N is a memory width in bits. For example, consider a 16-bit memory constructed as a single loop with output available only at one end of the loop. Suppose that bit 2 is currently available for reading but the desired memory location to be read is bit 15. It takes 13 clock cycles to rotate the memory loop in order to reach bit 15. The authors in [16] overcome this problem by using parallel read/serial write memory and a multiplexer device. The memory architecture is roughly similar to the architecture in [19] with an additional device, 2n-to-1 multiplexer at the output. The multiplexer selects one bit from the memory loop. Since QCA memory is always shifting in a pipelined wire, the use of the counter, adder, and multiplexer components helps to track the current memory location. The adder is used to offset the difference between current and desired memory locations. The output of the adder controls the select signals at the multiplexer. This eliminates the need to wait for the values in the memory loop to rotate into the current position. However, each memory address must pass through the adder before it reaches the select lines of the multiplexer. This adds to the latency of each read operation. This is a significant concern when using the memory for as a look-up table for an FPGA. In addition, the adder contributes to the area of the memory.

The memory design presented here uses serial write and parallel read (a similar memory interface to that in [16]) but replaces the 16-bit memory loop with sixteen one-bit memory cells, each of which is a modified version of one-bit memory cell in [15]. The new memory cell has two rails: DI (Data Input) rail and RW (Read/Write) rail. During write operations, signals flow in one direction on the DI rail, but in the opposite direction on the RW rail. This memory configuration allows N cells to be stacked shoulder-to-shoulder to create N-bit memory. A decoder and an output circuit replace the multiplexer and adder of [16]. However, the output circuit does not need counter and adder components to track the current memory location because each bit of memory is localized to its own loop. Therefore, the location of each bit is known in advance. The desired address directly drives the select lines of the decoder. As in [16], write operations are performed serially. This is appropriate for a LUT, because write operations are only used for configuration. Afterwards, the memory circuit is in read mode. Because of the heavily pipelined nature of QCA, each read incurs multiple clock cycles of latency. However, the throughput is one read operation per clock cycle. Memory is very important component of the LUT design. Various researchers have developed numerous LUT architectures. The above work provides a basic understanding for the LUT design.

Many researchers have developed Xilinx family architecture CLBs using QCA technology. They have used programmable interconnect strategy to develop the architecture.

1.5 Organization of Thesis

Chapter 1 provides introduction to research. It outlines the whole thesis. Chapter 2 provides the QCA background which gives fundamental understanding of the QCA Technology. Chapter 3 describes hierarchical levels of the FPGA design. Chapter 4 describes the design, modeling and implementation of LUT in QCA. Chapter 5 describes the implementation of Stratix logic array block. Chapter 6 describes the implementation of interconnect switch matrix. Chapter 7 discusses about the analysis of design and comparison of the design. Chapter 8 concludes the thesis work and discusses about the future research.

Chapter 2

Quantum-Dot Cellular Automata

2.1 Background

This section describes basics of QCA and its components such as a cell, wire, majority gate, and inverter. A combination of these components is needed to develop a digital circuit at the nanotechnology level. Quantum-dot cellular automata are novel devices with characteristics that differ substantially from CMOS. QCA takes advantage of the characteristics of the device to achieve memory densities that exceed the SIA roadmap for CMOS [11]. QCA architectures have low power dissipation. QCA takes less area for design than CMOS. FPGAs represent a good application for QCA technology [7].

Quantum-dot cellular automata (QCA) are an alternative to CMOS based designs. Quantum dots are nanostructures created from semi-conductive materials. These structures exhibit energy effects. A dot can be visualized as a quantum well. Once electrons are trapped inside the dot, the electrons require higher energy to escape from one dot to another. An idealized QCA cell can be viewed as a set of four charge containers or dots, positioned at the corners of a square [3], [4]. Each cell contains two electrons which can quantum mechanically tunnel between dots. The dots can be realized as electro-statically formed quantum dots in a semiconductor or reduction-oxidation centers in a molecule. There is a barrier between dots so that charge can move by tunneling. The configuration of charge within the cell is defined by cell polarization, which can vary between P = -1, representing a binary “0”, and P = +1, representing a binary “1”. This is illustrated in Figure 2-1.

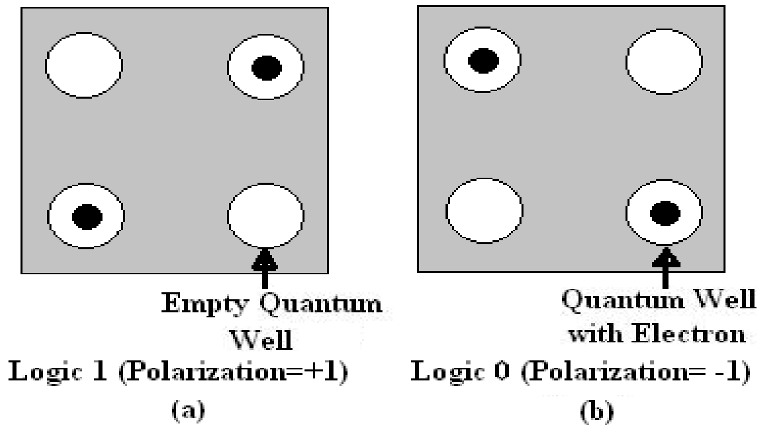
Empty Quantum Well
Logic 1 (Polarization=+1)
(a)
Quantum Well with Electron
Logic 0 (Polarization= -1)
(b)

Figure 2-1: QCA Cell Polarizations

QCA [20] uses the positions of electrons in quantum dots to represent binary values ‘0’ and ‘1’. Two electrons occupy each cell. Each electron is free to tunnel between dots within one cell, but cannot leave the cell. The two electrons within each cell repel each other to diagonally opposite corners of the cell. This leaves only two stable states for each cell. These two states are used to represent logic values 1 and 0 [4], [21].

2.2 QCA Circuits

2.2.1 QCA Wire

The Coulombic interaction is used to determine the positions of electrons in a cell. Consider a few QCA cells placed together as in Figure 2-2. Assume that the cell on the left hand side is the input cell and the cell on the right hand side is the output cell. The Coulombic interaction which is the magnitude of electron repulsion from neighboring cells determines the positions of the electrons in the normal cell. In case of logic '0', the first cell on the left responds to the input cell and outputs logic '0'. The cell is polarized to "-1" because of Coulombic interaction (electron repulsion). The second cell reproduces the logical value in the first cell. This process repeats for the remaining cells in the wire until it has reached the output cell. In other words, the wire exhibits an effect in which all cells adopt the same polarization.

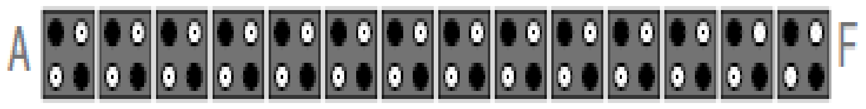

Figure 2-2: QCA Wire

In Figure 2-2, A is the input to the wire and F is the output from the wire. Figure 2-3 shows the simulation of a QCA wire. The wire shows the correct output depending upon the input to the wire which is shown in Figure 2-3.

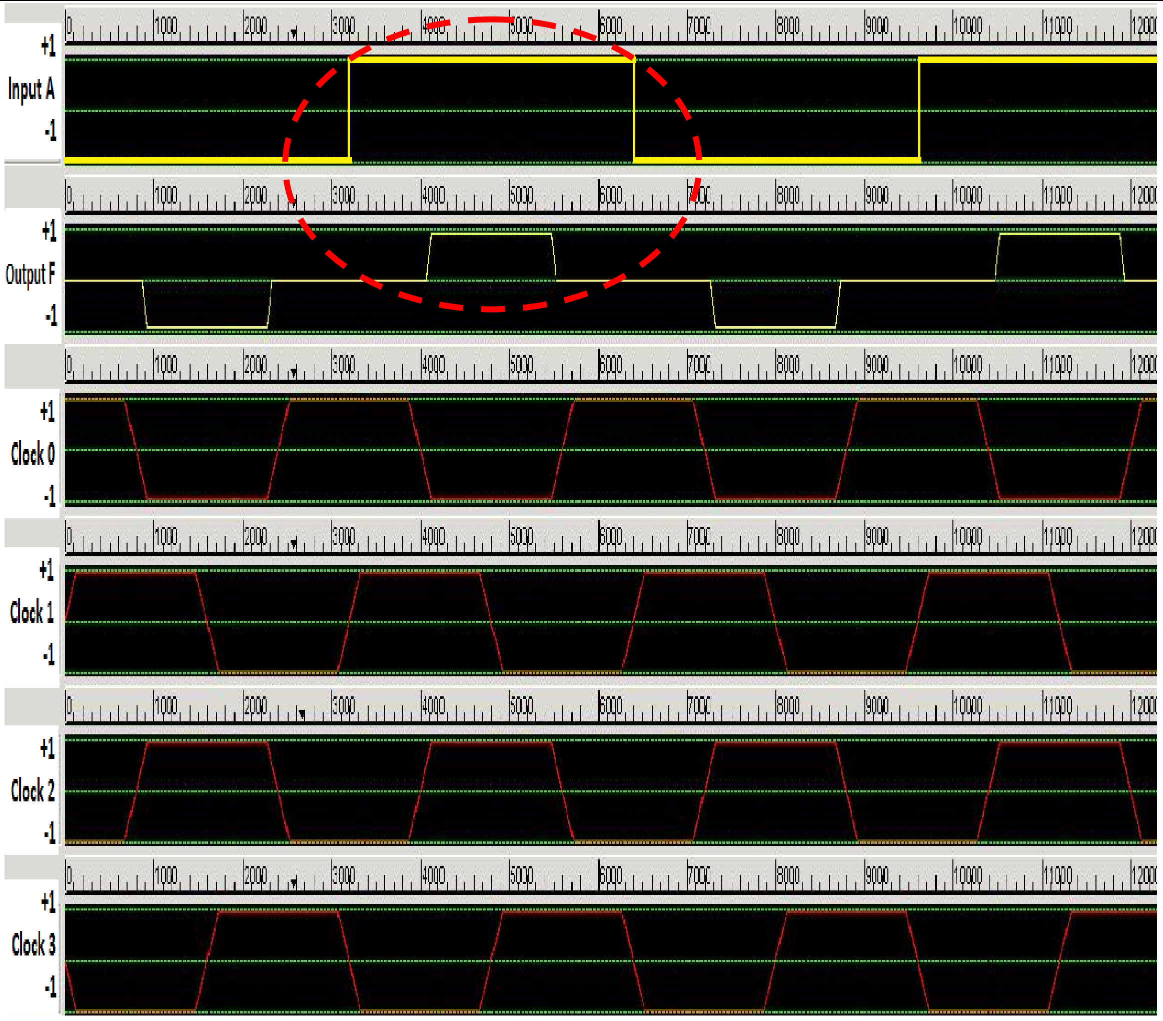

Figure 2-3: Simulation of a QCA Wire

2.2.2 Clocking in QCA

QCA technology requires the need of clock mechanism to operate properly. Unlike the standard CMOS clock, the QCA clock is not a signal with a high or low phase. Rather, the clock changes phases when potential barriers that affect a group of QCA cells (a clocking zone) passes through a series of four clock phases [21], [22]. The low level, rising edge, high level, and falling edge of a clock waveform represent the four phases [23] called hold, release, relax, and latch, respectively. During the switch phase of the clock, the unpolarized QCA cells polarize in accordance with the driver cell. In the hold phase, a QCA cell holds the current position of the electrons. During the rising edge, the electrons tunnel down to the bottom of the QCA cell and release its state of polarization because of the electron's attraction to the clock plane (release state). When the clock is high, the QCA cell is considered to have null polarization where electrons sit on the bottom of the cell (relax state). The net effect of this scheme is that even on a simple wire, data is latched as it moves from one clocking zone to the next. It is very much important that QCA cells should be pipelined according to clock phases [24].

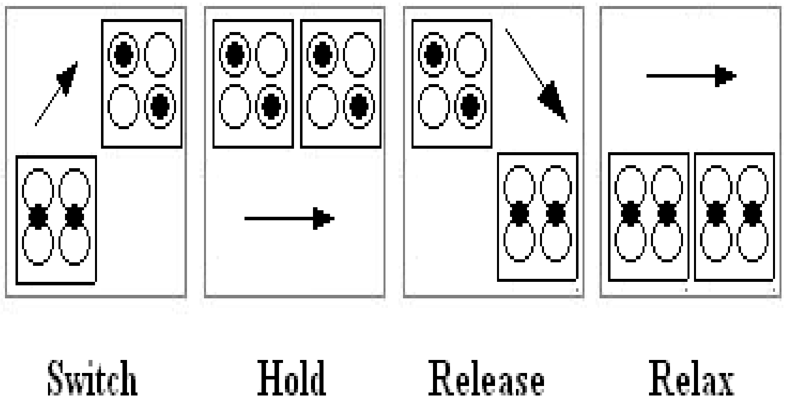

Figure 2-4: Clock Phases in QCA

There are two types of switching for the operation in QCA. These are abrupt switching and adiabatic switching:

- **Abrupt Switching:** When the inputs to the QCA circuit change suddenly, the circuit would be in some randomly excited state which is unpredictable. Therefore, the QCA circuit has to be relaxed to ground state by dissipating energy. This inelastic relaxation is uncontrolled and the QCA circuit might enter a meta-stable state that may be determined by a ground state.

- **Adiabatic Switching:** In this kind of switching, the system is always kept in its instantaneous ground state. A clock signal is introduced to ensure adiabatic switching. This is the preferred method of switching.

2.2.3 Inversion Chain

There is another kind of the QCA wire called an inversion chain as shown in Figure 2-5. The position of quantum dots in a normal cell is rotated 45º clockwise or counterclockwise to become an inverted cell. One quantum dot is placed halfway between two corners on each side of the QCA cell. The inversion chain inverts the logical value from the previous cell due to Coulombic interaction. If the number of inverted cells between the input and output cells is odd, the output cell presents a buffered version of the input cell. Similarly, if the number of inverted cells between the input and output cells is even, the output cell presents an inverted version of the input cell.

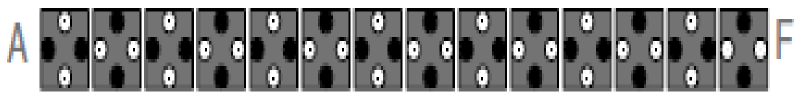

Figure 2-5: Inversion chain

The inversion chains are useful for developing bus rails with perpendicularly tapped normal wires. This can be done by placing a normal cell below or above the inversion chains and aligning the cell halfway between inverted cells at a certain location to obtain a buffered value or inverted value.

2.2.4 Majority Gate

Now the signals can be passed along wires but the gates need to be constructed to compute various digital logic functions. The basic QCA logic device is called the majority gate. The majority gate implements the logic function F = AB + BC + AC, where A, B, and C are inputs and F is the single output. By fixing a single input to 0, an AND gate can be implemented (Figure 2-6 (a)) whereas fixing an input to 1 implements an OR gate (Figure 2-6 (b)). Fortunately, creating a fixed cell can be done within the manufacturing process and constant signals do not need to be routed within the circuit. The sums of Coulombic interaction (signals *A*, *B*, and *C*) determine the value at the output where the dominant interaction wins the vote.

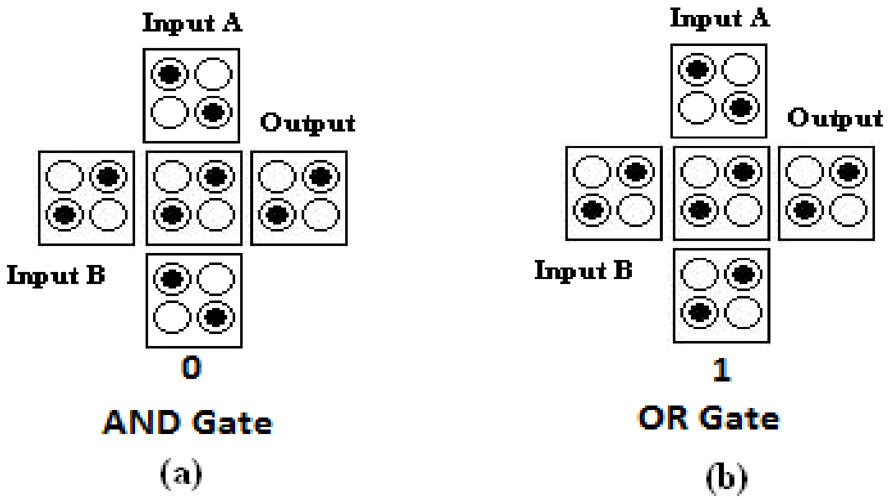

Figure 2-6: QCA Devices (a) AND Gate (b) OR Gate

Table 2.1: Truth Table for Majority Gate

Assigned Input	Input A	Input B	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1

(a) AND Gate

Assigned Input	Input A	Input B	Output
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(b) OR Gate

Figure 2-7 shows the simulation of the AND gate in QCA and Figure 2-8 shows the simulation of OR gate in QCA. From these figures, Table 2.1 can be verified to determine the expected output. In Figure 2-7, output ‘1’ is marked for AB = ‘11’ to show the expected result. In Figure 2-8, output ‘1’ is marked for AB = ‘10’ to show the expected result.

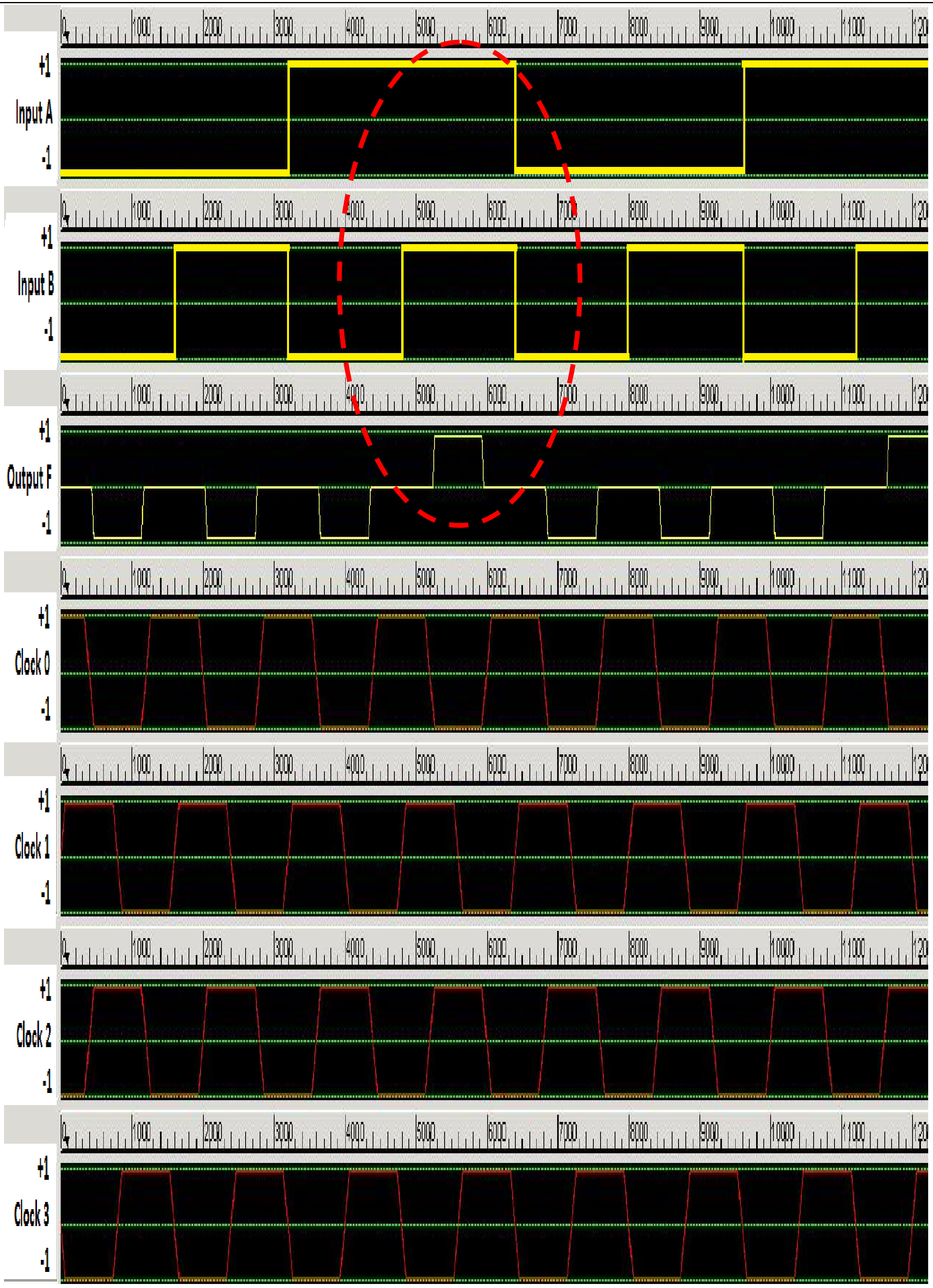

Figure 2-7: Simulation of AND Gate

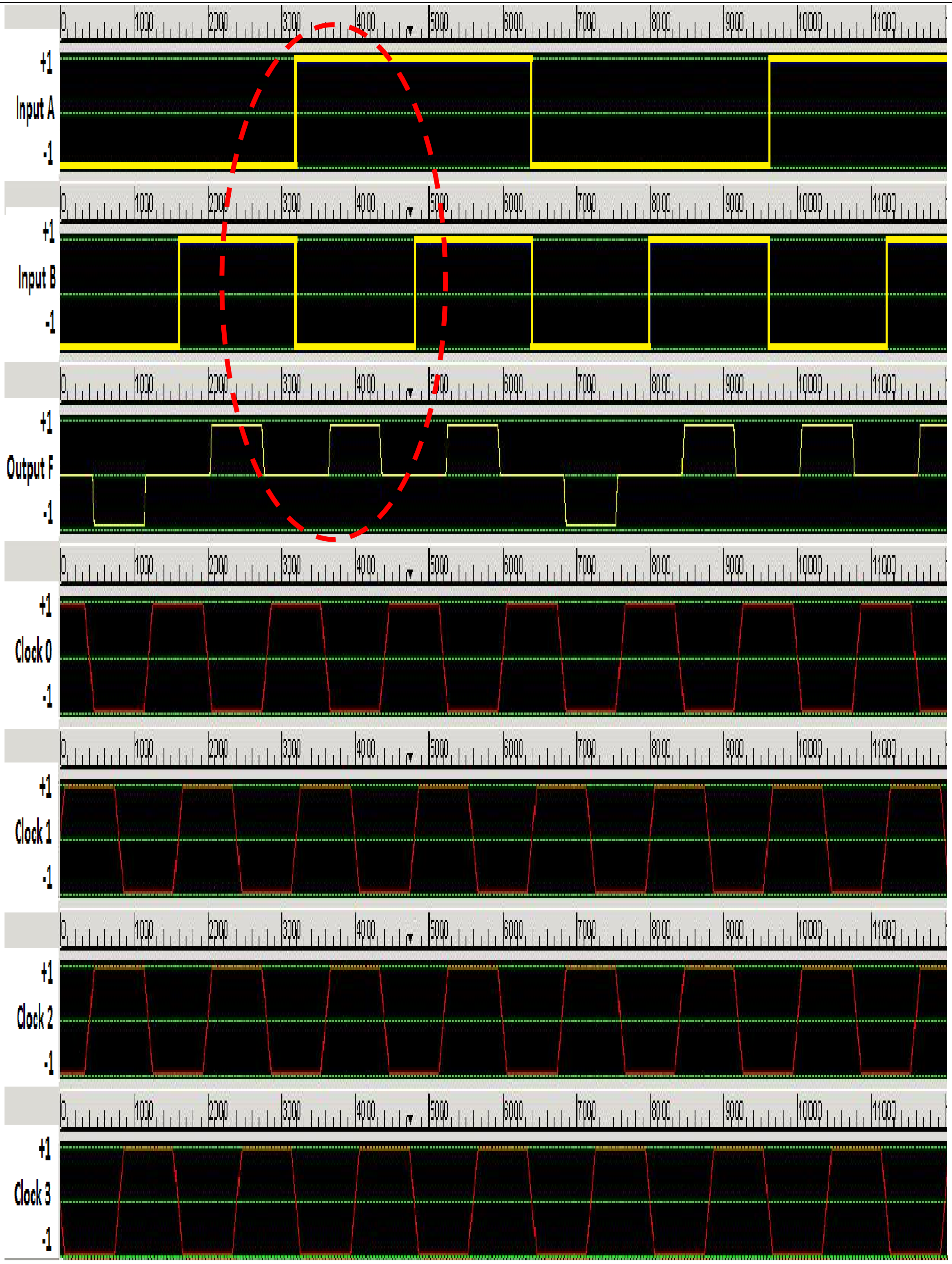

Figure 2-8: Simulation of OR Gate

2.2.5 Inverter Gate

In an inverter gate, the input wire is constructed like a tuning fork. The left cell of the output wire is aligned vertically between the two prongs of the fork and is abutted horizontally on the edge sides of the prongs. The logical value in the left cell of the output wire is determined by the Coulombic interaction between it and the cells at the ends of the prongs. An inverter gate uses either one clock zone or two clock zones. In the former arrangement, both input wire and output wire use the same clock zone. In the latter arrangement, the input wire is assigned to one clock zone, while the output wire is assigned to the next clock zone.

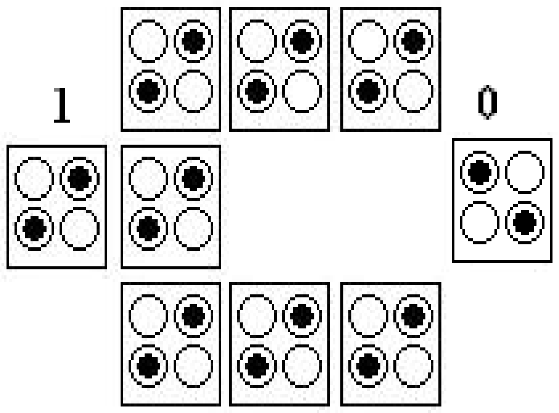

Figure 2-9: QCA Inverter

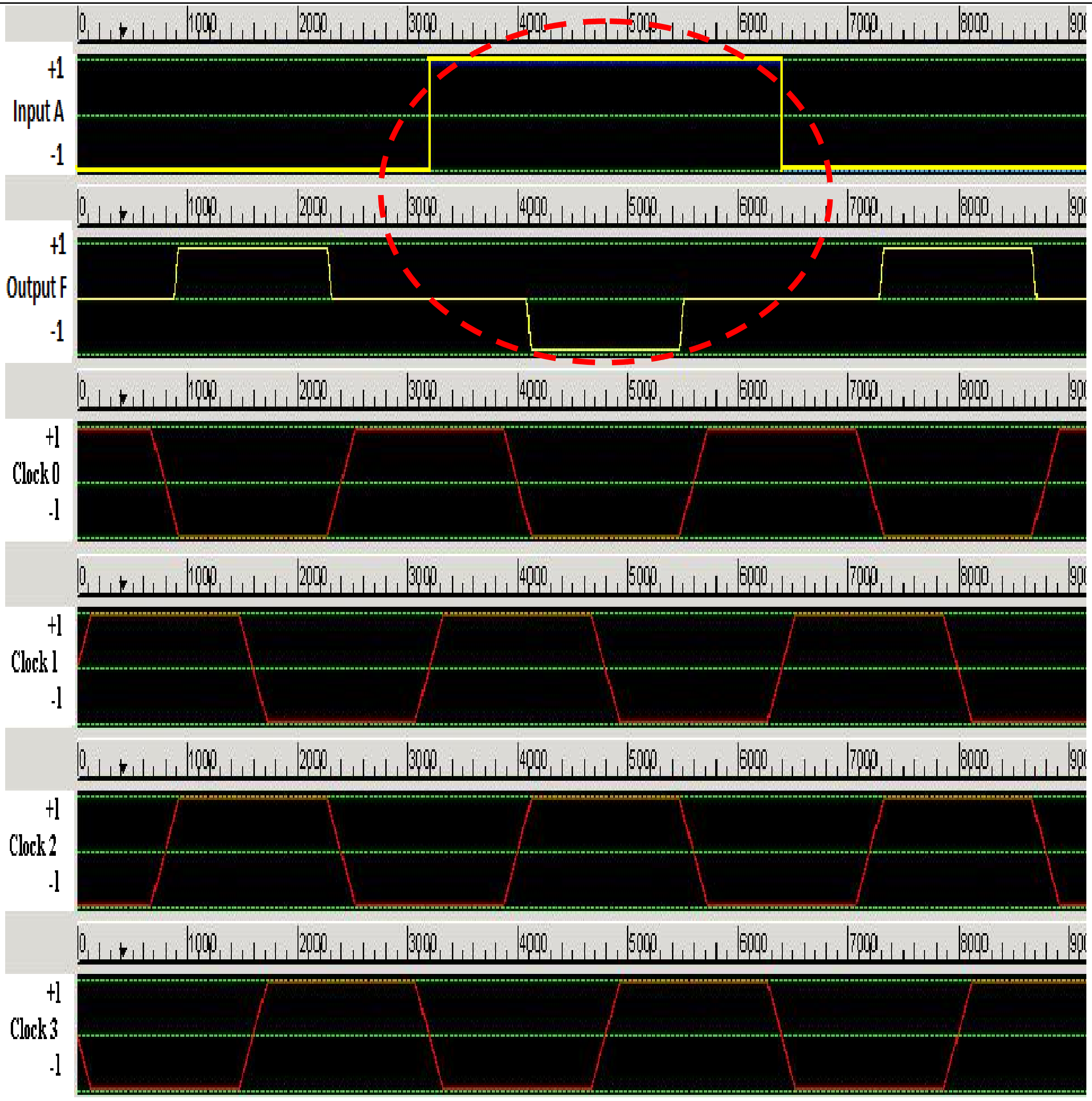

Figure 2-10: Simulation of Inverter Gate

So by arranging these components, we can implement any digital circuit using the QCA technology [24]. The aim of this work is to design a potentially implementable architecture for the digital logic applications.

2.3 Wire Crossings in QCA:

2.3.1 Coplanar Crossover:

In a circuit, two wires are crossing each other by physically passing two wires one over the other with some insulation in between them. This type of arrangement can also be done in QCA using a wire with normal cells and another wire the orientation of cells rotated by 45 degrees. This type of arrangement is called coplanar crossover. This is based on the fact that regular cells with dots arranged along the diagonal and cells with orientation rotated by 45 degrees don't interact with each other. Figure 2-11 shows the coplanar crossover in QCA. The horizontal and vertical wire in the figure doesn't interact with each other and signal is carried from one end to the other. However, this wire crossing would be sensitive to fabrication issues [25]. Detailed analysis has to be carried out to resolve this problem. Hence, the variations in the position of the cells can introduce crosstalk between interconnects. For this wire crossing, the cells have to be properly aligned. Any misalignment of the cells leads to a non-zero value of kink energy resulting in the interaction of the cells in the region of crossover. In Figure 2-11, QCA wire with input A and output Z is modeled with the normal QCA cells. QCA wire with input B and output Y is modeled with the rotated QCA cells. Figure 2-12 shows the simulation for the coplanar crossover. The dotted lines in Figure 2-12 show the corresponding output depending upon the input.

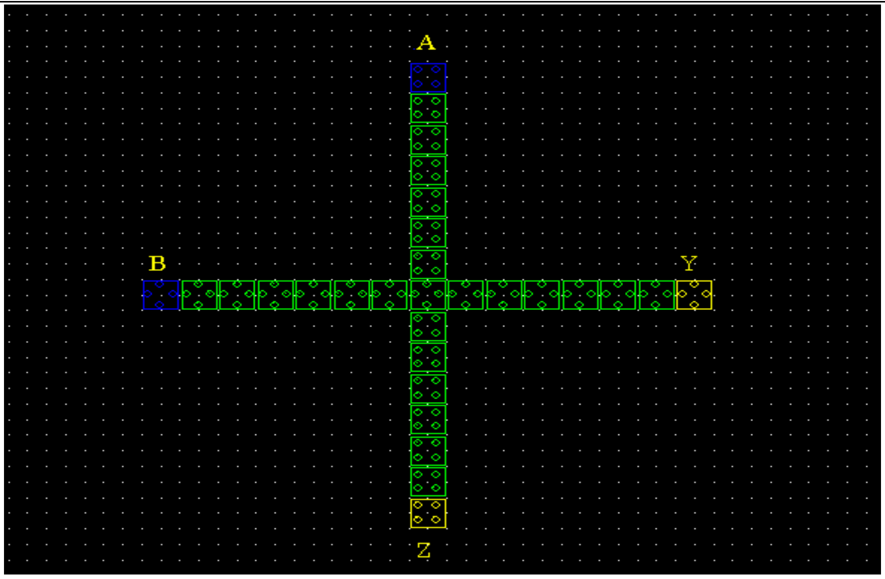

Figure 2-11: Coplanar Crossovers in QCA

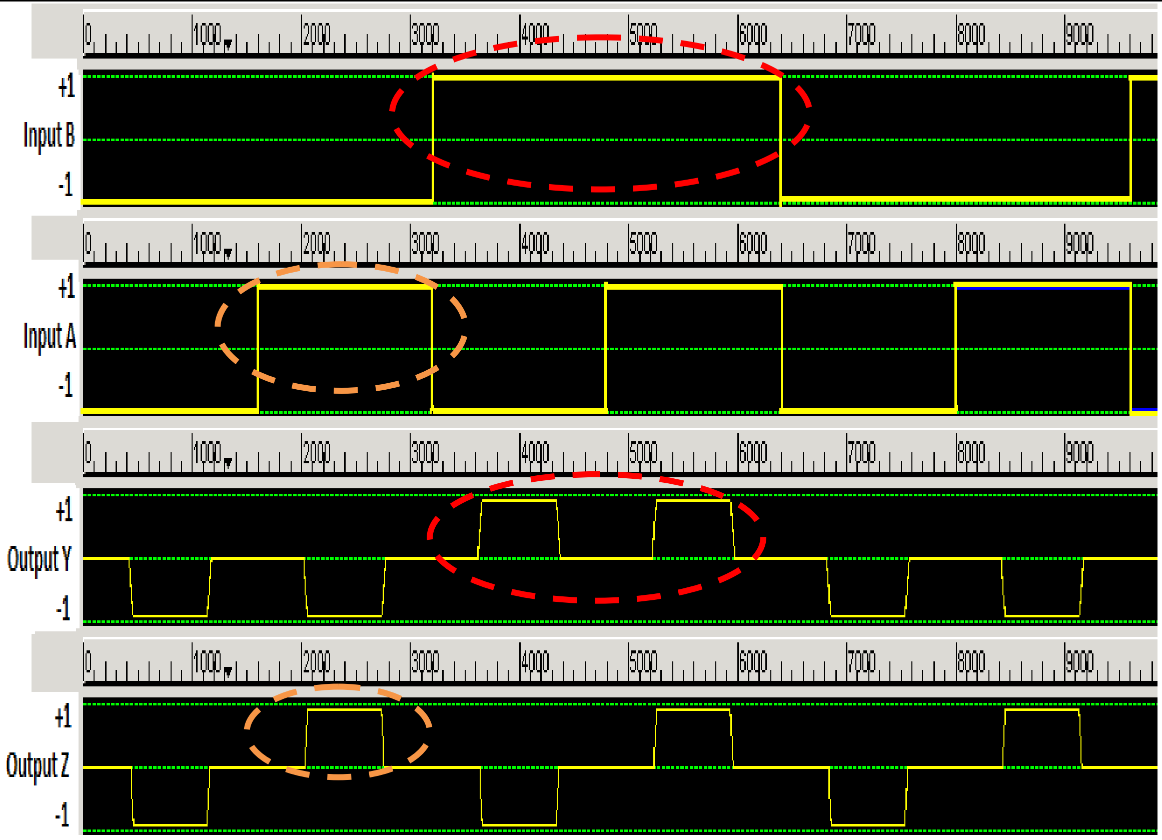

Figure 2-12: Simulation of a Coplanar Crossover

2.3.2 Multi-layer Crossover

To implement the digital circuits in QCA, it is required that signals being able to cross each other. As we have seen that the coplanar crossing is sensitive to various facts, it is essential to find an alternative method for the crossing of signals. Multi-layer QCA is the solution for the wire crossings in QCA. This is very efficient method for the crossing of signals. This is done by keeping the cells one over the other. In this technique, the cells are arranged in layers and after the crossing, the signal travels in a horizontal line. The layers created for the crossover can also be used as active components of the circuit [25] which gives optimization of area. This technique performs well and it eliminates the problem in coplanar crossover. Figure 2-13 shows the multilayer crossover.

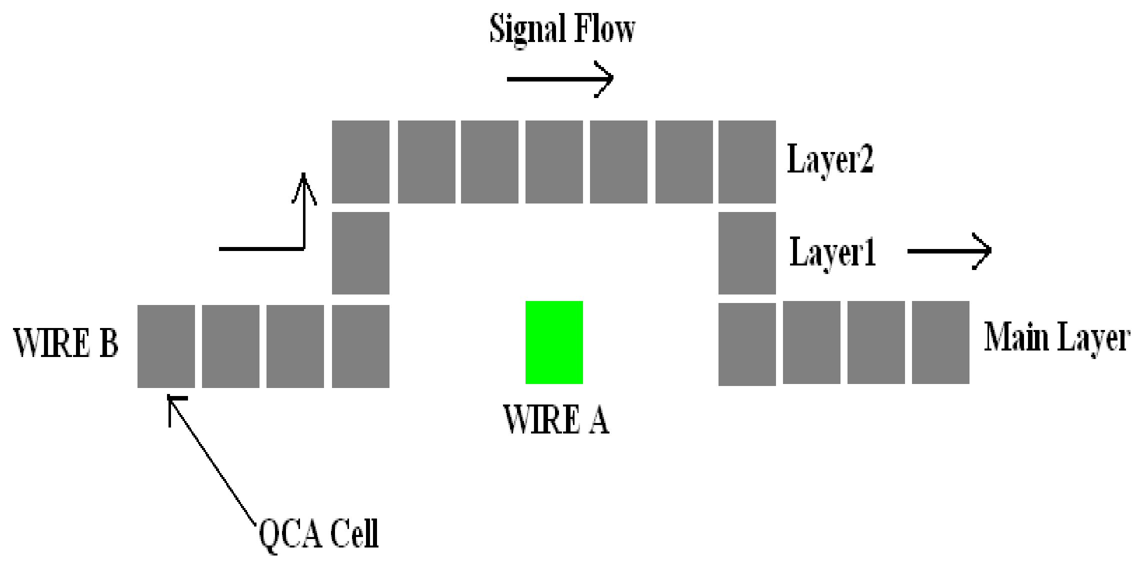

Figure 2-13: Multi-layer Crossovers in QCA

Chapter 3

FPGA Hierarchical Architectures

This Chapter describes hierarchical architectures for the design presented in this work. The design uses bottom to top method for the implementation from three sub-components of a LUT to a LAB.

3.1 Field Programmable Gate Array (FPGA)

FPGA presents a good application in QCA technology. In hierarchy of digital systems, FPGA is the topmost design entity. Several Logic Array Blocks or Configurable Logic Blocks are put together in such a combination to form FPGA architecture. Although, FPGA design has not been implemented, but it should be a simple task to put together a cellular array of identical copies of the LAB presented in this work to form FPGA with some modification in routing wires between LABs.

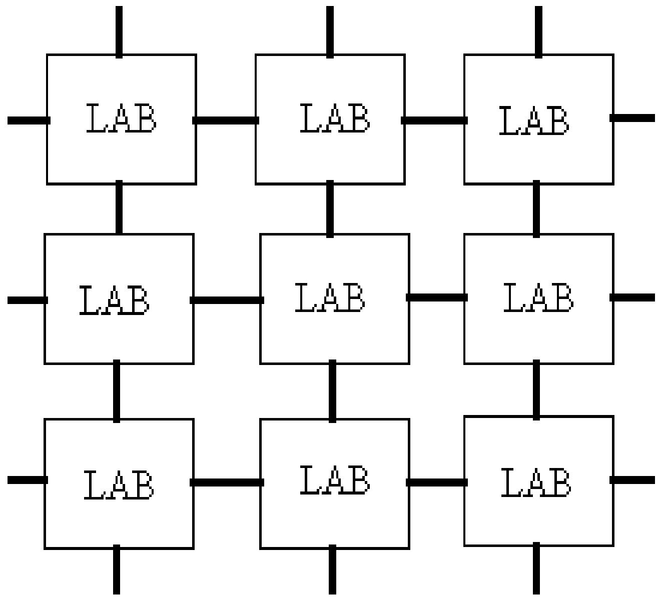

Figure 3-1: FPGA Architecture showing LABs

3.2 Logic Array Block (LAB)

The proposed LAB implements four LUTs where each of them is placed in such a manner to implement the design. Each LUT computes an independent Boolean function of four inputs. At the upper most level of the LAB, all corresponding inputs of the four LUTs are connected as inputs to LAB. Thus, each output from the LAB can be any function of the inputs from its neighbor LUTs. The LUT can be programmed for any Boolean function by using an external FPGA programmer to shift values into the LUT memories.

3.3 Interconnects

FPGA architecture contains logic array blocks and programmable interconnects. LAB is used to implement the logic in FPGA. A hierarchy of programmable interconnects called programmable switch matrix (PSM) allows logic blocks in FPGA to be interconnected. Logic blocks and interconnects can be programmed as per the needs of the application. The detailed explanation about interconnects is given in Chapter 6. Also, a QCA design of interconnects is modeled and shown in Chapter 6.

3.4 Look-Up Table (LUT)

Each LUT consists of a four-to-sixteen decoder, a memory, and an output circuit. The purpose of the decoder is to enable one of the sixteen lines and to select the memory cell at the address specified by the four inputs. The implementation of the decoder uses a tree structure of two-to-four decoders. Each output line of the four-to-sixteen decoder connects to an enable signal of a QCA memory cell. The outputs of sixteen memory cells are combined together to form a single output called the output circuit. Since three-state buffers do not exist in QCA technology, a tree structure of OR-gates is used. When an enable signal in a memory cell is not enabled, it asserts logic '0' at the output to avoid a data collision with an enabled memory cell. The details are given in Chapter 4.

Chapter 4

Implementation of Look-Up Table in QCA

This chapter covers the implementation of a LUT with three sub-components: a 4-to-16 decoder, a 16-bit memory, and an output circuit. Each component is explained in detail in the following sections.

4.1 Decoder

The basic 4-to-16 decoder block diagram is shown in Figure 4-1. There are five two-to-four decoders in a tree structure configuration. However, while designing a decoder using QCA circuits, we need to take care of wire crossings within the circuit. So, the five two-to-four decoders are arranged in a tree structure configuration to build a 4-to-16 decoder as shown.

A two stage method is used to design this decoder which makes the design simple to model it in QCA. A combination of 2-to-4 decoders is used to design a 4-to-16 decoder. This is shown in Figure 4-1. It is possible to design a four-to-sixteen decoder in a single stage. However, this configuration is not effective in QCA technology due to the increased number of inputs in a single clock zone and complicated coplanar wire crossings and timing issues. For instance, the inputs (signal A, B, C, and D) are located on the left side of a single-stage decoder. Assume that it takes one clock cycle from one bit to the next bit in the input rails. It takes one clock cycle from the inputs to output bit F, while it takes sixteen clock cycles from the inputs to output bit 0. This results in an unequal delay between inputs and outputs. One can compensate for the unequal delay by adding a delay at each output (15 clock cycles for bit 15, 14 clock cycles for bit 14 and so on). However, this is not a good solution because of lengthy delay-compensated wires. In addition, four signal rails with coplanar interconnects in the same clock zone do not provide reliable signals due to cross-talk interference from perpendicular tapped wires.

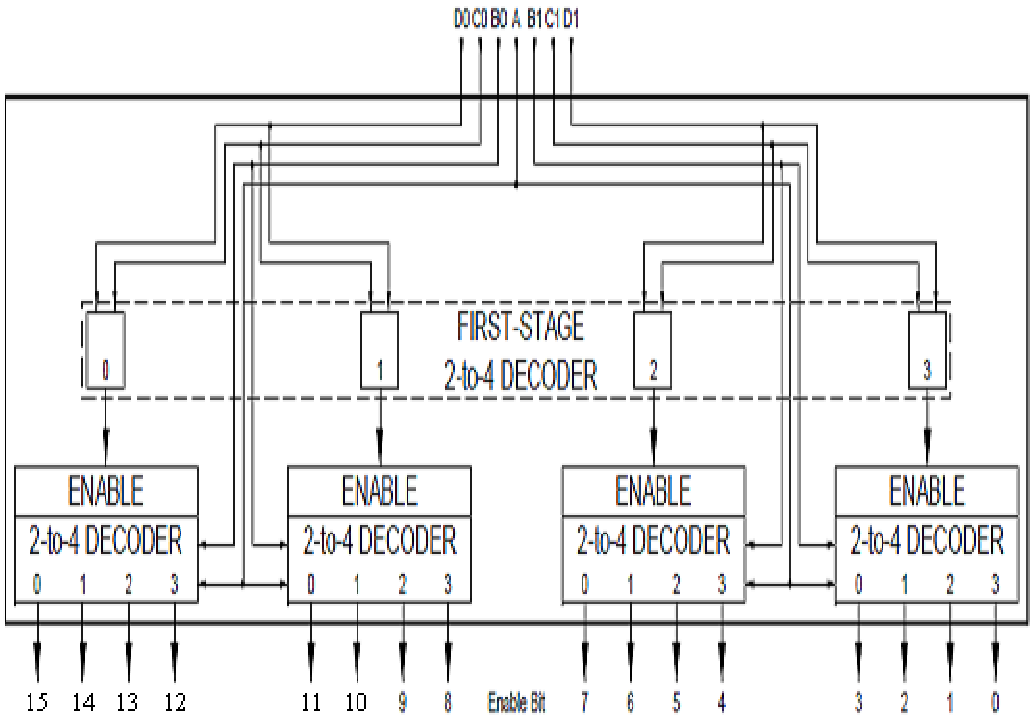

Figure 4-1: Block Diagram of 4-to-16 Decoder using Two-stage Decoder Approach

Each of four primary input signals (A, B, C, D) must fan-out to four different locations to connect to the 4 decoders. C and D are distributed to the first-stage decoder. Inputs A and B are distributed to the four decoders in the second stage. However, there is a technical issue with the fan-out in QCA technology. The fan-out has complicated coplanar wire crossings. Different signals are flowing in different directions all in a small space. It is difficult to set tracks for direction because in QCA clock zones are used to control the direction of information flow. Therefore a symmetrical fan-out gives the solution to the problem where there are no wire crossings. So, we have approached this problem using B = B0 = B1, C = C0 = C1, and D = D0 = D1. A secondary fan-out (after the primary fan-out) uses the elbow fan-out with a few coplanar wire crossings. Now these 7 variables are the inputs to the decoder. The outputs of decoder shown in Figure 4-1 are labeled with enable lines. Memory cells are enabled by these enable lines from the decoder. A tree structure is beneficial because its symmetry for QCA pipelined circuits where input to output has the same delay. The decoder is designed in two stages. In the first stage, there are four 2-input AND gates. The inputs of AND gates are connected to the input wire C (C = C0 = C1) and the input wire D (D = D0 = D1) through inverters where needed. In the second stage, each decoder has eight 2-input AND gates. Four of these AND gates controls the enabling or disabling the output. The circuit configuration is the same as in the first-stage decoder with three additional A, B, and EN rails and four additional 2-input AND gates. If the enable rail (EN) is high (logic '1'), it activates the two-to-four decoder and the decoder perform its normal operation; otherwise, it deactivates the decoder and forces all outputs to logic '0'. A combination of one first-stage decoder and four two-stage decoder results the 4-to-16 decoder. Figure 4-2 shows the basic 2-to-4 decoder QCA layout.

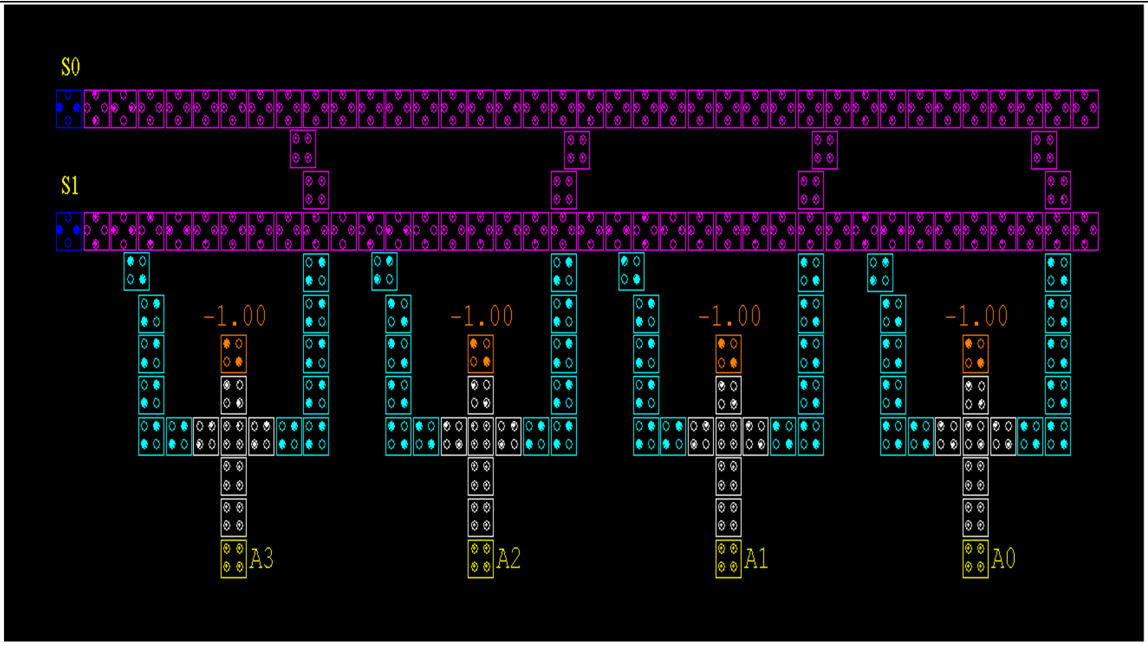

Figure 4-2: QCA Layout of 2-to-4 Decoder

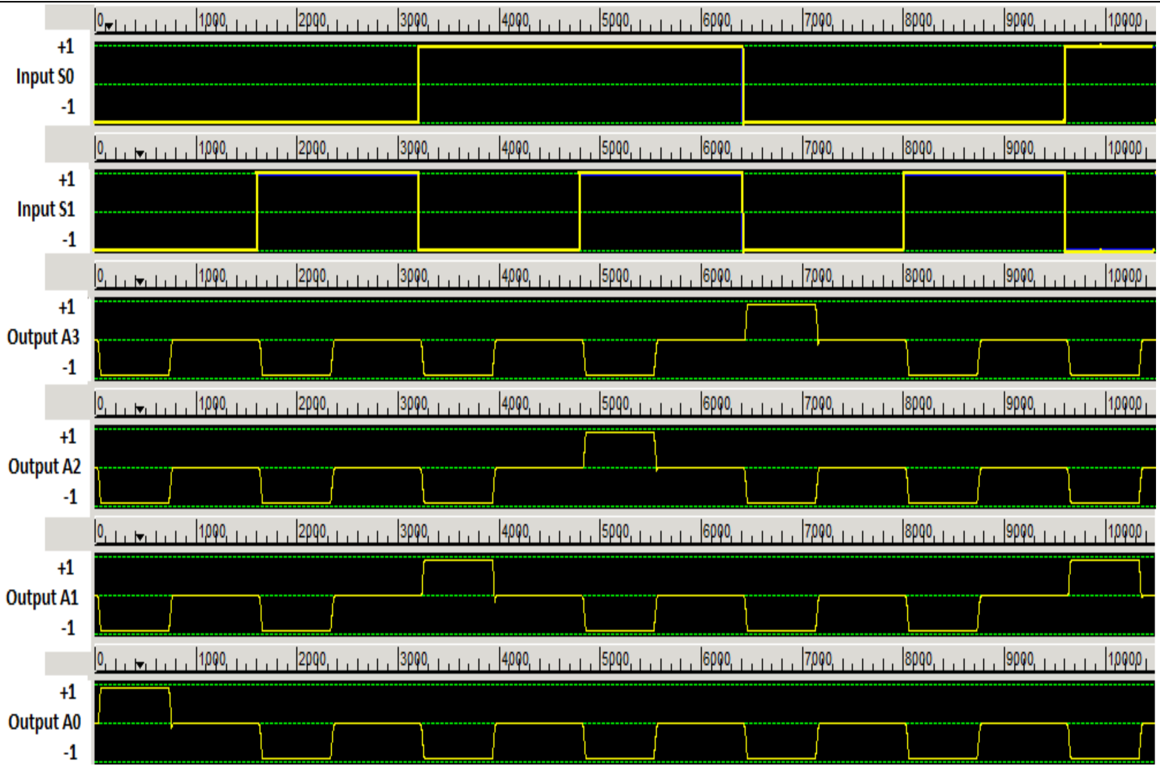

Figure 4-3: Simulation of 2-to-4 Decoder

Four second-stage decoders are placed in such a way to implement sixteen enable lines. The first-stage decoder is placed above the second-stage decoders. The second-stage decoders receive input signals A and B, while the first-stage decoder receives input signals C and D, as in Figure 4-1. It takes five clock cycles to travel from the inputs to the outputs. Figure 4-4 shows the QCA layout of 4-to-16 decoder.

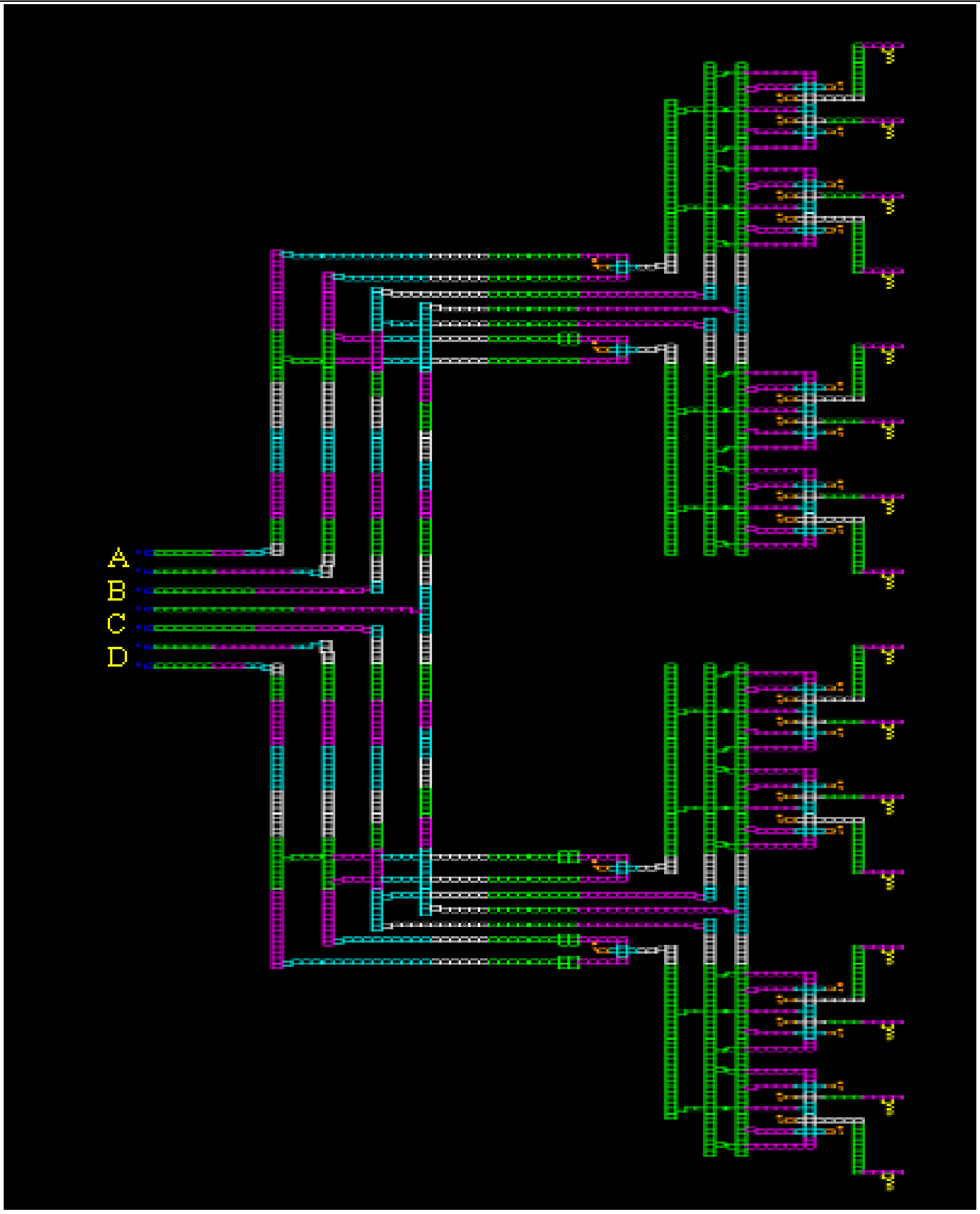

Figure 4-4: QCA Layout of 4-to-16 Decoder

4.2 Memory Architecture

The QCA memory is at its best use over the standard CMOS RAM when memory accesses are entering the memory at very high rate, i.e. as soon as one access fully enters the memory; another access is ready to be processed. In today's von Neumann type architectures, processors are much faster than memories and they generate requests at a rate that cannot be satisfied by the memory. However, memory cells are the essential components in any digital circuit. They are needed to store programmed values to perform Boolean functions based on address lines. Memory cells presented here are the modified version of the memory architecture from [15], [12]. The original design has five AND gates, one OR gate, and two inverters with Row Select, Write/Read, Input, and Output signals. Decoder selects a memory cell to accept a new value during the write mode using a parallel configuration. Similarly, during the read mode, an enabled memory gives the stored value [15]. The memory cells in this design have a serial Read/Write (RW) signal, a serial Data Input (DI) signal, a parallel Enable (EN) signal, and a parallel Output signal. Compared to the memory cell from [15], the design presented here eliminates a Row Select signal, two AND gates and one inverter. The RW and DI rails in the modified memory cells use inversion chains. The memory cells are stacked side-by-side with RW and DI rails connected in series to create an N-bit memory. This allows the entire memory to be configured as a shift register for writing. The enable line in each memory cell is connected to an output of the four-to-sixteen decoder. This allows random access for parallel reading. Figure 4-5 shows the diagram of a 1-bit memory cell presented in this work.

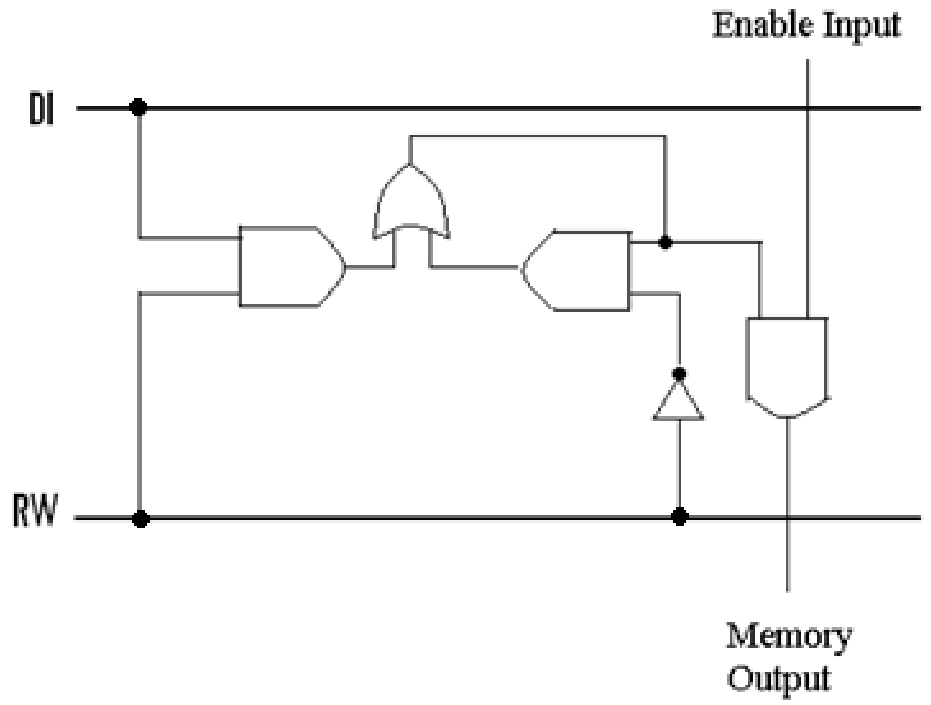

Figure 4-5: Gate Level Representation for Memory

During the write mode (RW = '1'), the left AND gate outputs the new value from DI. At the same time, the middle AND gate erases the stored value by outputting '0'. After 1/4th of a clock, the OR gate gives the output with inputs as the outputs of left and middle AND gates. Similarly, during the read mode (RW = '0'), the output of the left AND gate is forced to output '0' where it rejects the value from the DI. The lower input of the middle AND gate is always '1'. Thus, the gate outputs the current value in the memory loop. Because the output of left AND is '0', the OR gate transports from right input to the output. The output of the OR gate is looped back to the upper input of the middle AND gate. In a QCA system, this allows the stored value to be preserved in the loop until the write mode occurs. The right AND gate is called an enable gate and operates independently from the rest of the circuit. Regardless of whether one is in the read or write mode, the enable gate outputs the stored value when EN is '1'. This shows that the memory cell is selected to be read. Otherwise, when EN is '0', the output is '0' which means that the memory cell is not selected to be read. The summary of the memory operation for read and write modes is shown in Table 4.1. The RW and DI rails in each memory cell are connected back-to-back to create the sixteen-bit memory. The RW and DI input ports are at the left side of the RW rail and the right side of the DI rail, respectively. Since QCA is a micro pipelined system, the memory design presented here employs two important rules for writing to an N-bit memory. First, the RW (left to right) and DI (right to left) signals must propagate in opposite directions. Second, during the write mode, the RW signal must use a one-cycle pulse and pass it down from bit F (the leftmost bit) to bit 0 (the rightmost bit). During clock cycle 0, the memory is initialized. In the next cycle, the DI port starts receiving the binary string 0101 with the rightmost bit first. At the same time, bit 3 of the memory is in write mode because there is a one-cycle pulse on the RW rail at that memory bit. The logic '1' is loaded at the DI input in bit 3. During clock cycle 2, the one-cycle pulse is shifted to bit 2. The rightmost bit of the value is also shifted to bit 2. The previous memory cell (bit 3) updates the logic '1' in the memory loop and changes to in the read mode. The process repeats for remaining memory cells.

Table 4.1: Truth Table for the Memory Cell Operation

Mode	**RW**	**EN**	**DI**	**Memory Loop**	**OUT**
Read	0	0	X	No change	0
Read	0	1	X	No change	Stored value
Write	1	X	0	0	Don’t care
Write	1	X	1	1	Don’t care

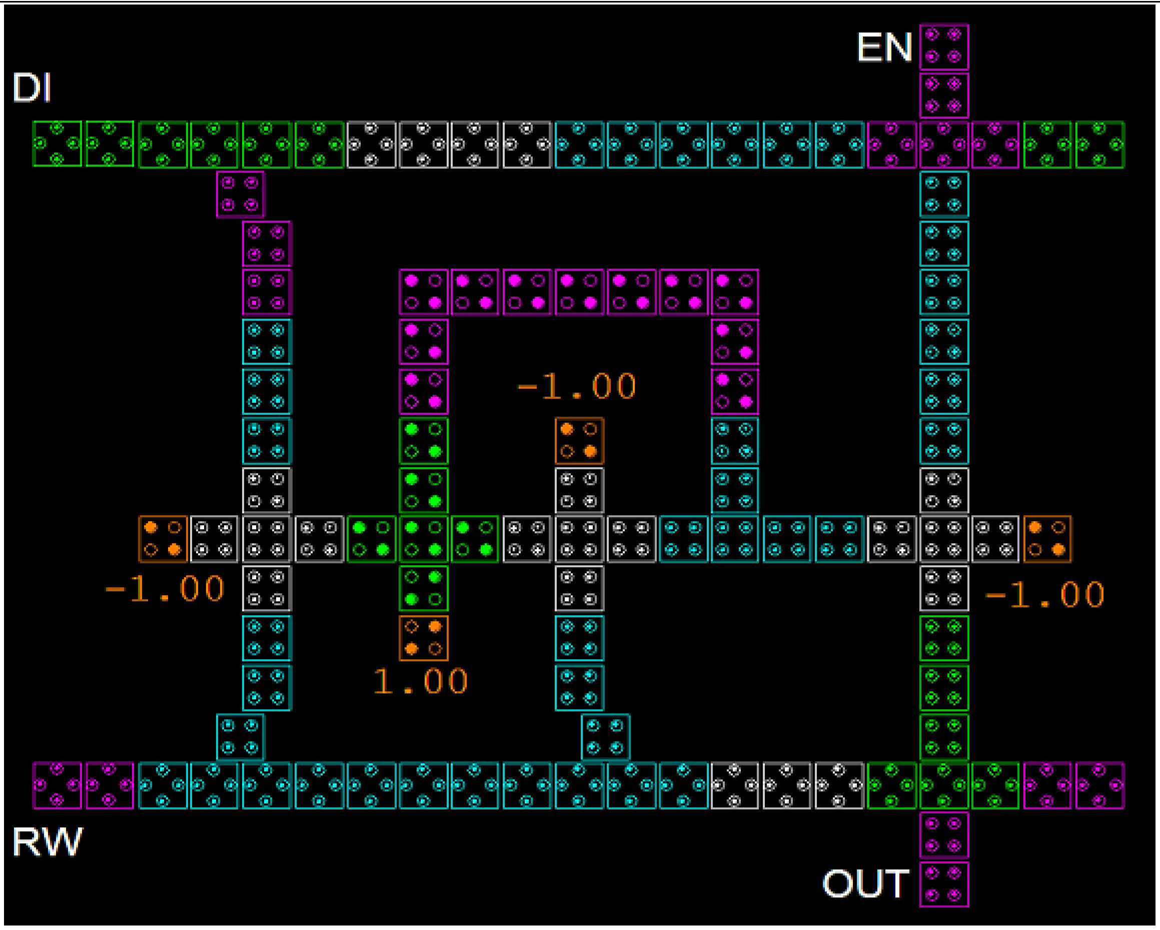

Figure 4-6: QCA Layout of 1-bit Memory Cell

4.3 Output Circuit of LUT

The purpose of the output circuit is to give a final output by using a tree structure of OR gates by narrowing down the outputs from memory cells. Because QCA cannot take three inputs at a time in case of an OR gate, that is why the shared busses must be handled with logic. The order of values that appear at the output corresponds to the order in which the memory cells are enabled. The disabled memory cells are forced to output logic '0' to avoid a data collision with an enabled memory cell. Each memory cell takes a turn (one per clock cycle) depending on the address selection.

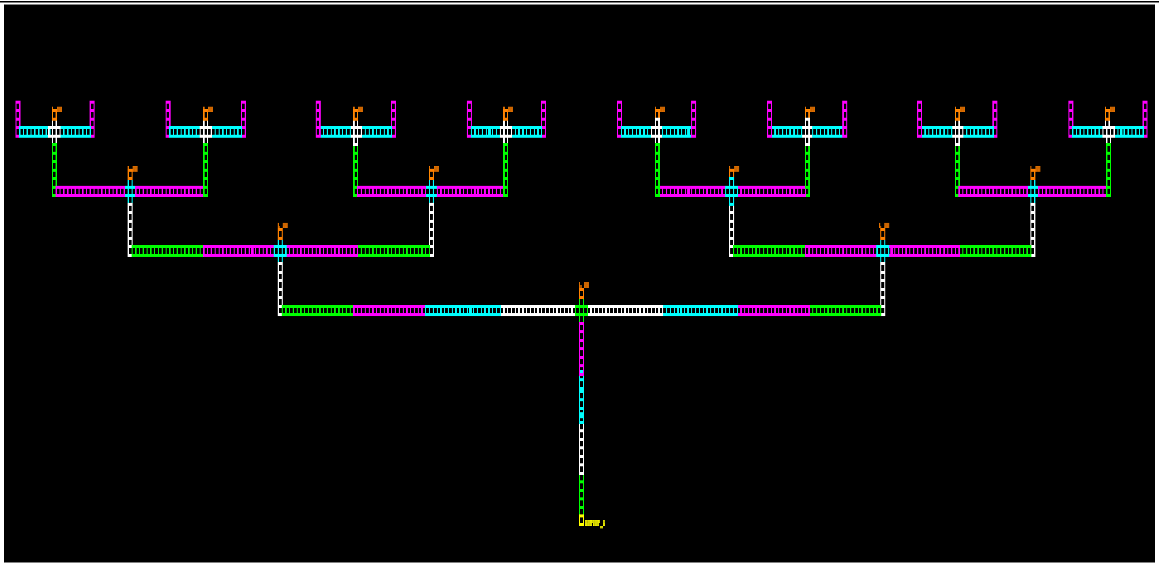

Figure 4-7: QCA Layout of Output Circuit of LUT

4.4 Entire Assembly of LUT

The entire LUT block diagram is shown in Figure 4-8. The figure shows three components of LUT: a four-to-sixteen decoder, a sixteen-bit memory, and an output circuit. They are combined to implement a look-up table. The enable line of a memory cell is connected to the corresponding enable line in the decoder. Similarly, the output of a memory cell is connected to the corresponding input of the output circuit (the tree structure of OR gates).

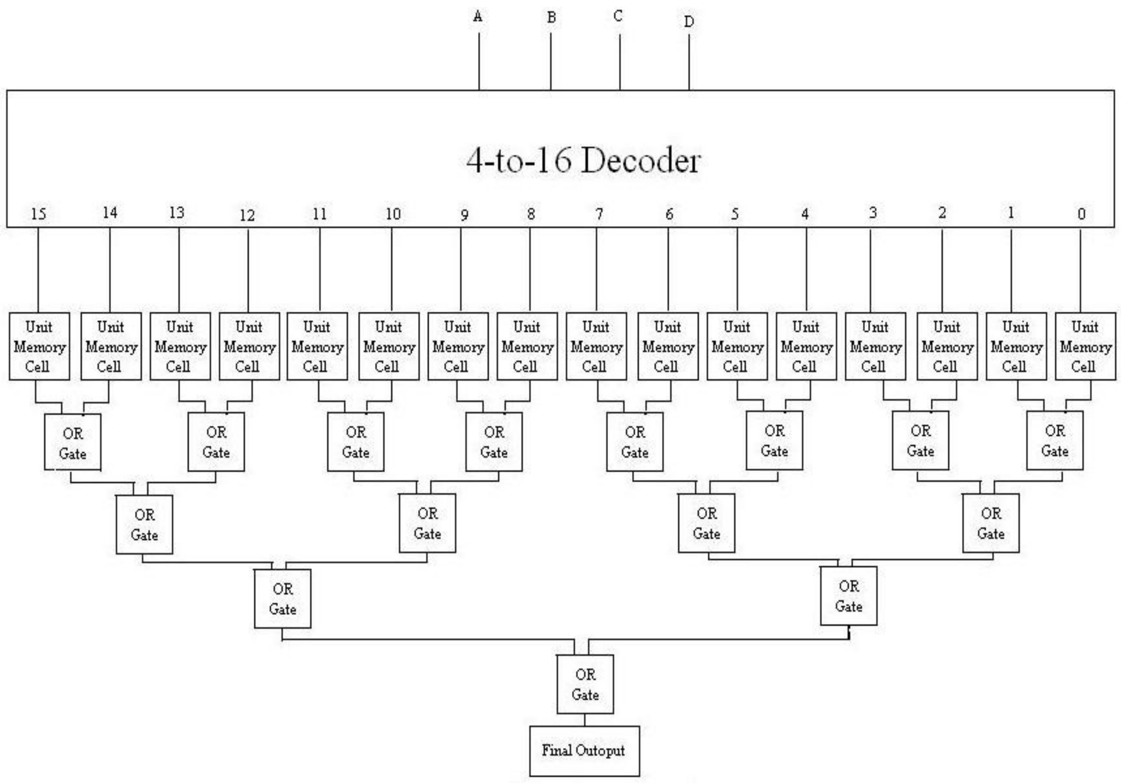

Figure 4-8: Block Diagram of LUT

The QCA layout of the LUT is shown in Figure 4-9. The latency is eleven clock cycles long between the inputs of the decoder and the final output. We will compare the LUT design with other LUTs in Chapter 7. LUT functions according to the truth table. Appropriate memory cell is enabled by the decoder and the output of the LUT shows correct result.

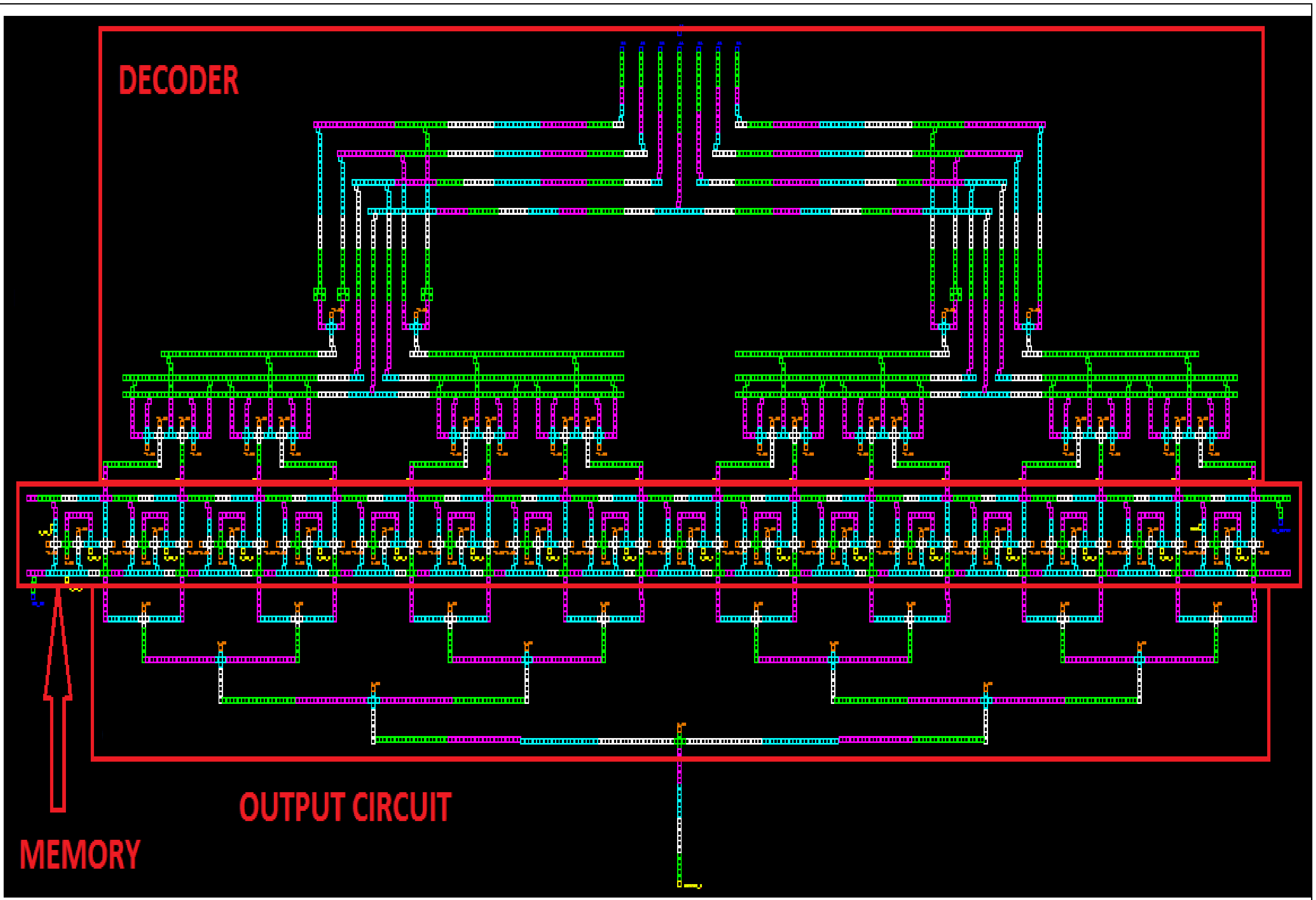

Figure 4.9 QCA Layout of Entire LUT

We have implemented this LUT design in QCADesigner. Depending upon the input to the decoder, appropriate memory cell is enabled and the output is obtained. The following table shows memory cell is enabled according to the input to the decoder.

Table 4.2: Table showing Enabled Memory Cell depending upon the Input to Decoder

Input				**Enabled Memory Cell**															
A	**B**	**C**	**D**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	0	0	1															
0	0	0	1		1														
0	0	1	0			1													
0	0	1	1				1												
0	1	0	0					1											
0	1	0	1						1										
0	1	1	0							1									
0	1	1	1								1								
1	0	0	0									1							
1	0	0	1										1						
1	0	1	0											1					
1	0	1	1												1				
1	1	0	0													1			
1	1	0	1														1		
1	1	1	0															1	
1	1	1	1																1

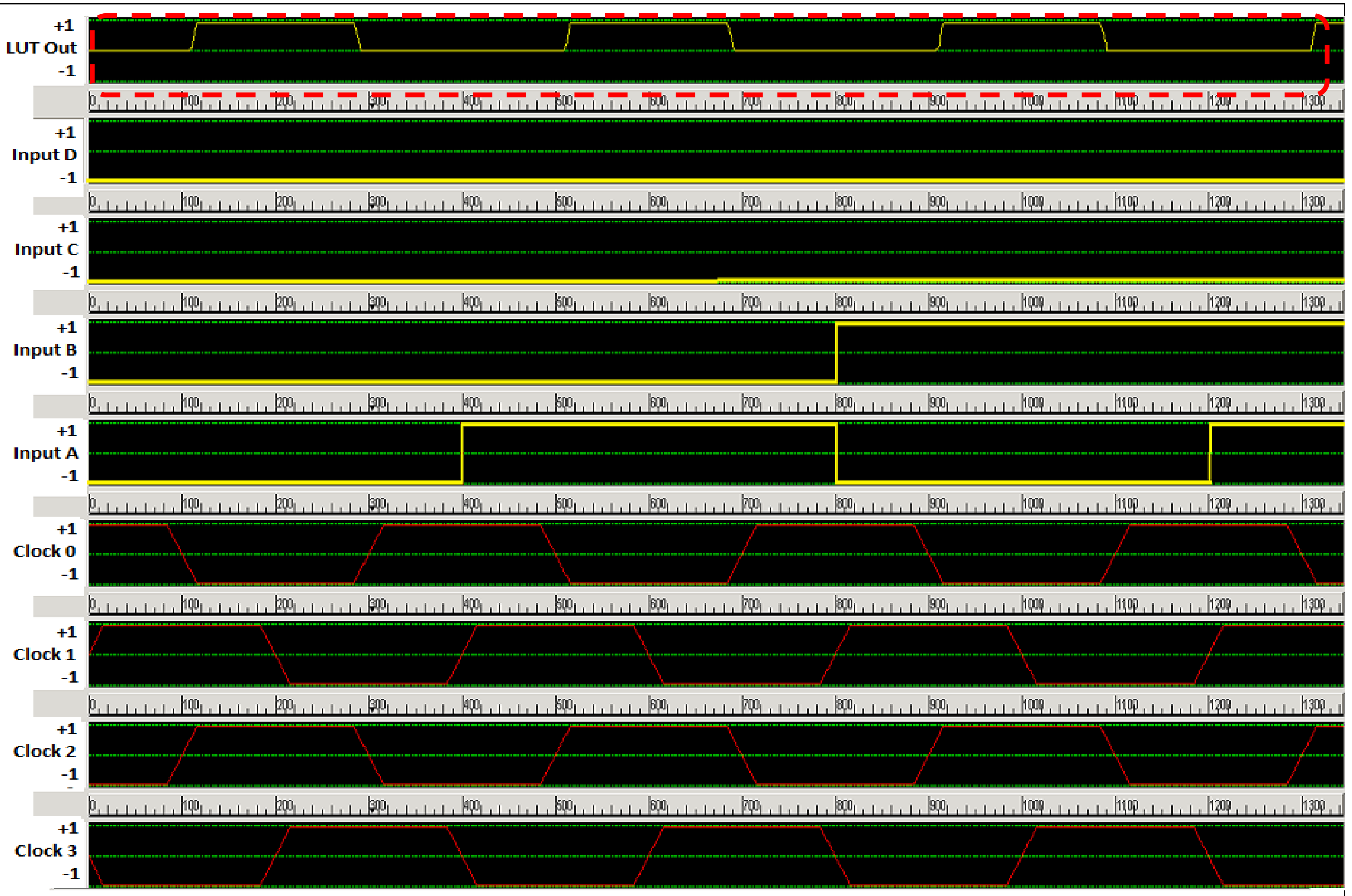

Figure 4-10: Simulation of 4-Input LUT

Thus, we have successfully completed the main block of the design. The latency is eleven clock cycles long between the inputs of the decoder and the final output. The LUT used here gives an advantage to the LUT design in [13] in that it uses less number of QCA cells.

Chapter 5

Design and Implementation of Altera Stratix Logic Array Block in QCA

5.1 Introduction to Altera Stratix LAB

The main aim of our research is to design the Altera Stratix LAB and implement it in QCADesigner. Logic Array Block (LAB) is the basic logic structure in Altera Stratix architecture and is shown in Figure 5-1. The LAB consists of combinational logic block, adders, multiplexers and registers. A carry chain is used to connect multiple LABs in the structure. The Altera Stratix architecture in QCA uses less number of QCA cells and provides more functionality as compared to other implementations [12], [13], [14]. There are various modes in which the Stratix LAB can be operated [26]. These are as follows:

- **Normal mode**: It is suitable for general logic applications and combinational functions. In this mode, up to eight data inputs from the LAB local interconnect are inputs to the combinational logic. The normal mode allows two functions to be implemented in one Stratix LAB, or LAB to implement a single function of up to six inputs. The LAB can support certain combinations of completely independent functions and various combinations of functions which have common inputs. The normal mode provides complete backward compatibility with four input LUT architectures. Two independent functions of four inputs or less can be implemented in one Stratix LAB. In addition, a five-input function and an independent three-input function can be implemented without sharing inputs.

- **Extended LUT mode**: It is used to implement a specific set of seven-input functions.
- **Arithmetic mode**: It is ideal for implementing adders, counters, accumulators, wide parity functions, and comparators. A LAB in arithmetic mode uses two sets of two four-input LUTs along with two dedicated full adders. The dedicated adders allow the LUTs to be available to perform pre-adder logic; therefore, each adder can add the output of two four-input functions. The four LUTs share the inputs. As shown in Figure 5-1, the carry-in signal feeds to adder0, and the carry-out from adder0 feeds to carry-in of adder1. The carry-out from adder1 drives to adder0 of the next LAB in arithmetic mode. While operating in arithmetic mode, the LAB can support simultaneous use of the adder's carry output along with combinational logic outputs. In this operation, the adder output is ignored. This usage of the adder with the combinational logic output provides resource savings of up to 50% for functions that can use this ability. The carry chain provides a fast carry function between the dedicated adders in arithmetic or shared arithmetic mode. Carry chains can begin in either the first LAB or the fifth LAB.
- **Shared arithmetic mode**: It is an enhanced version of arithmetic mode. It is used for addition of numbers having more bits.

In our research we are going to implement a LAB in arithmetic mode as this mode gives more functionality than other modes. Figure 5-1 shows the block diagram of the LAB in the arithmetic mode. The details of this diagram can be found in [26].

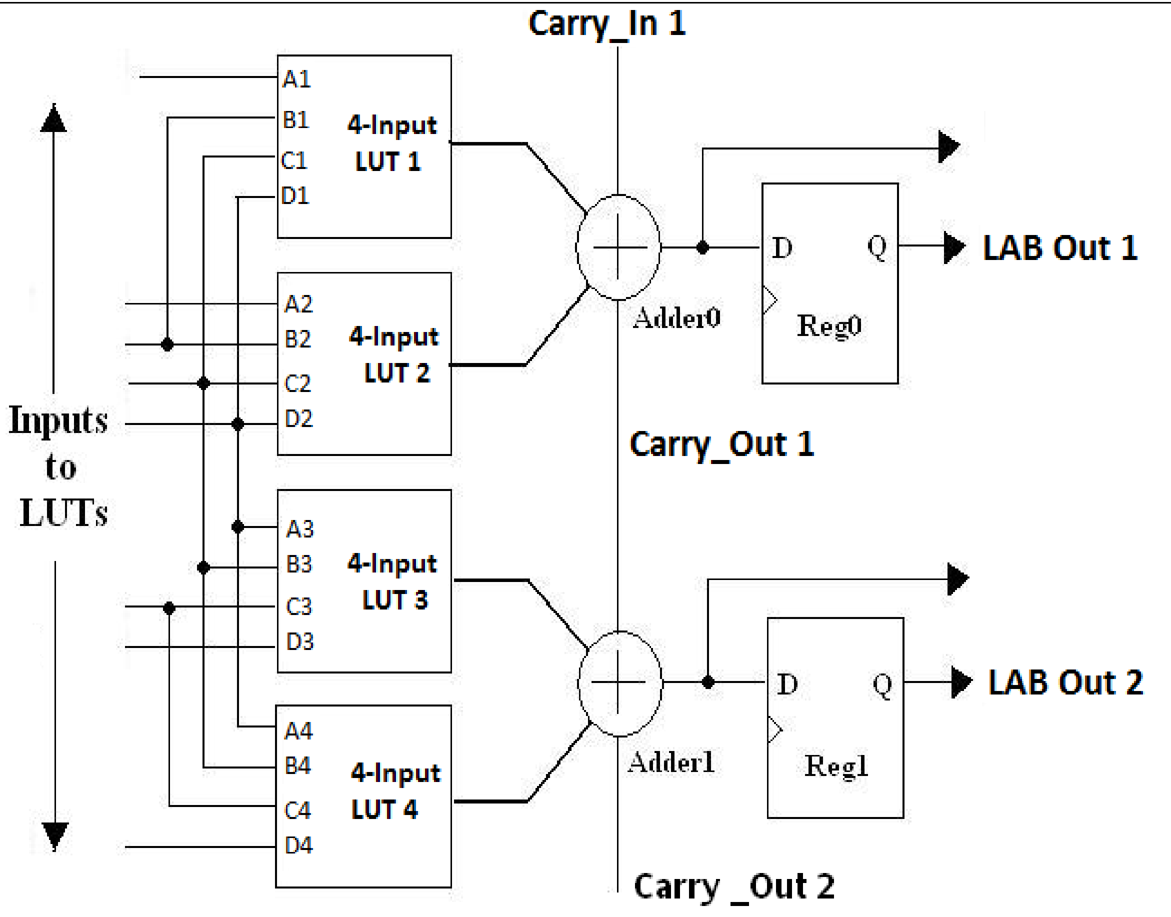

Figure 5-1: Block Diagram of Altera Stratix LAB in Arithmetic Mode

Briefly, an LAB in arithmetic mode uses two sets of two four-input LUTs along with two dedicated full adders. The four LUTs share the inputs. The carry-in signal feeds to adder0, and the carry-out from adder0 feeds to carry-in of adder1. This usage of the adder with the combinational logic output provides resource savings of up to 50% for functions that can use this ability [26]. The Altera Stratix LAB in arithmetic mode is discussed in detail in Section 5.3.

Before going ahead with the design of Stratix LAB, we need to model the basic components required for the implementation. As we can see that the LAB requires 2 adders i.e. full adders and 2 flip-flops. This is shown in Figure 5-1. We will have to implement these circuits in QCA to complete the design of Stratix LAB.

5.2 Components used in Design of Stratix LAB

5.2.1 Full Adder

There are various ways for adding three bits in the digital systems. For addition of three bits, it requires a full adder circuit. Full adder is a digital circuit which is used for addition of three bits (Inputs). We are familiar with the design of full adder using basic CMOS logic gates. In the CMOS technology, it requires more gates to model a full adder circuit. While in QCA the full adder circuit only requires three majority gates. This is one example where the QCA technology has advantage over the CMOS technology. In QCA, full adder is designed in a different manner. We have seen in Section 2.2.3, majority gate has three inputs. We set one of the input to either 0 or 1 to implement it as AND or OR gate respectively. While designing the full adder, we take the advantage of property of QCA to select the majority of inputs when the cells are arranged in majority function. Figure 5-2 shows the gate level diagram of full adder.

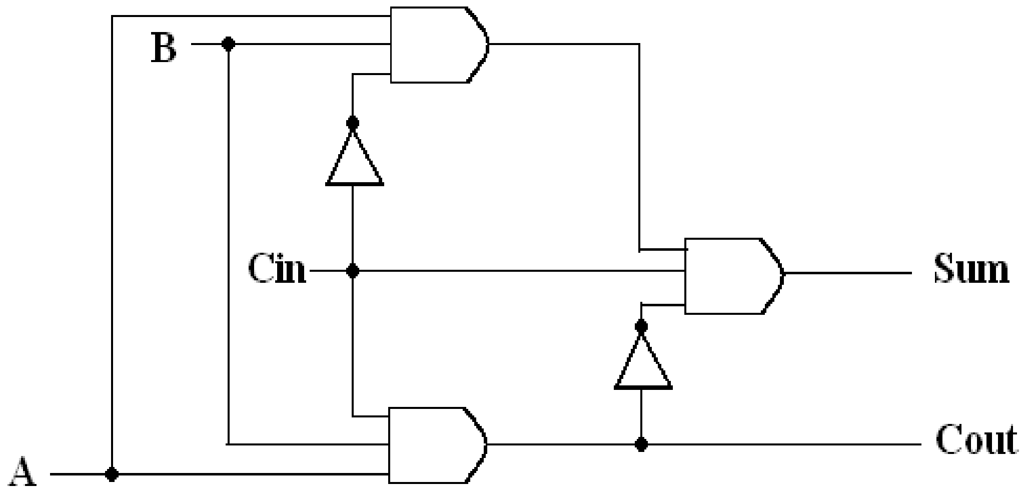

Figure 5-2: Majority Gate Level Logic Diagram of Full Adder

Figure 5-3 shows the QCA design of full adder using the design shown in Figure 5-2. The majority voter concept in QCA is very useful for the design of full adder. It is not required to model the basic gates like AND, OR. This concept simplifies the circuits and gives correct Sum and Cout values. The simulated output of the full adder circuit implemented here is shown in Figure 5-4. It is observed that full adder circuit becomes very simple with the QCA technology. Correct output can be observed in Figure 5-4. Thus, full adder circuit is implemented in QCA. The dotted lines in Figure 5-4 shows that for A, B, Cin = ‘1’, ‘0’, ‘1’ respectively, the Sum of outputs is ‘0’ and the Carry Out output is ‘1’.

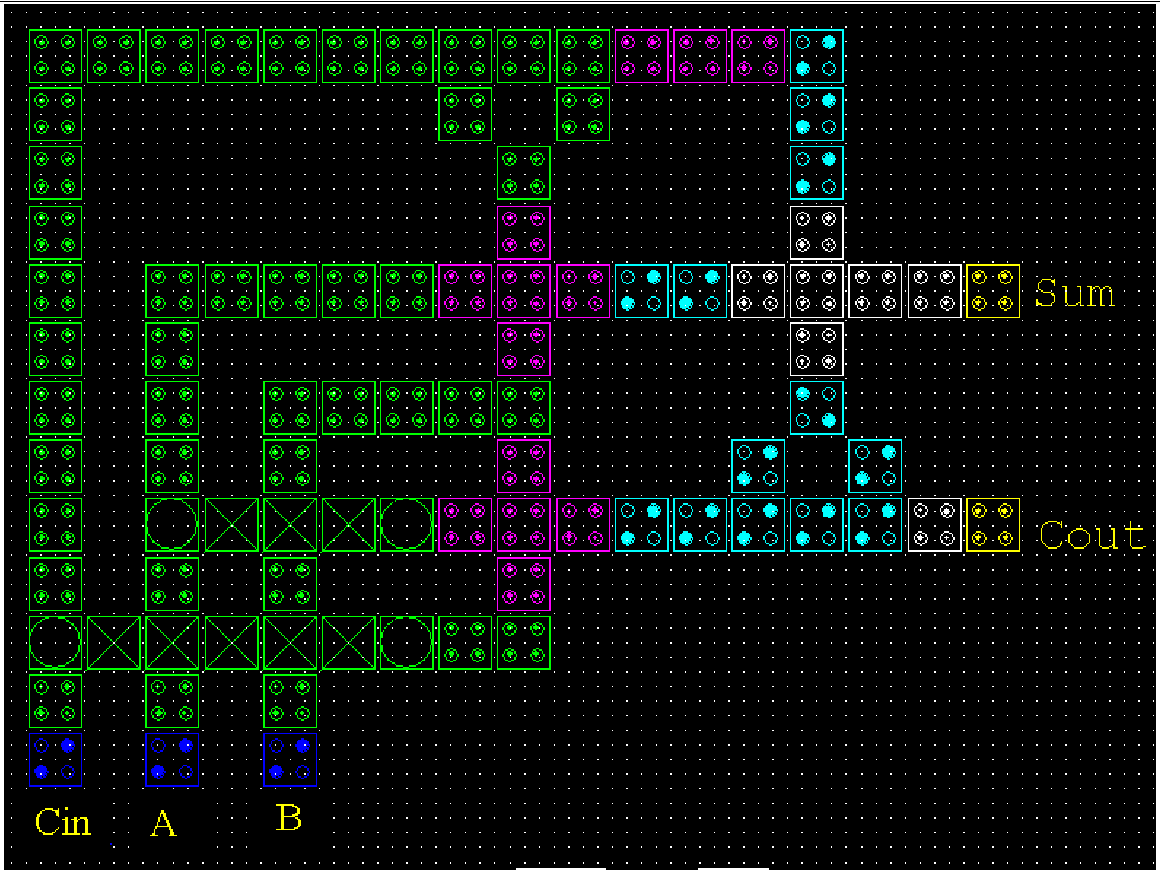

Figure 5-3: QCA Layout of Full Adder

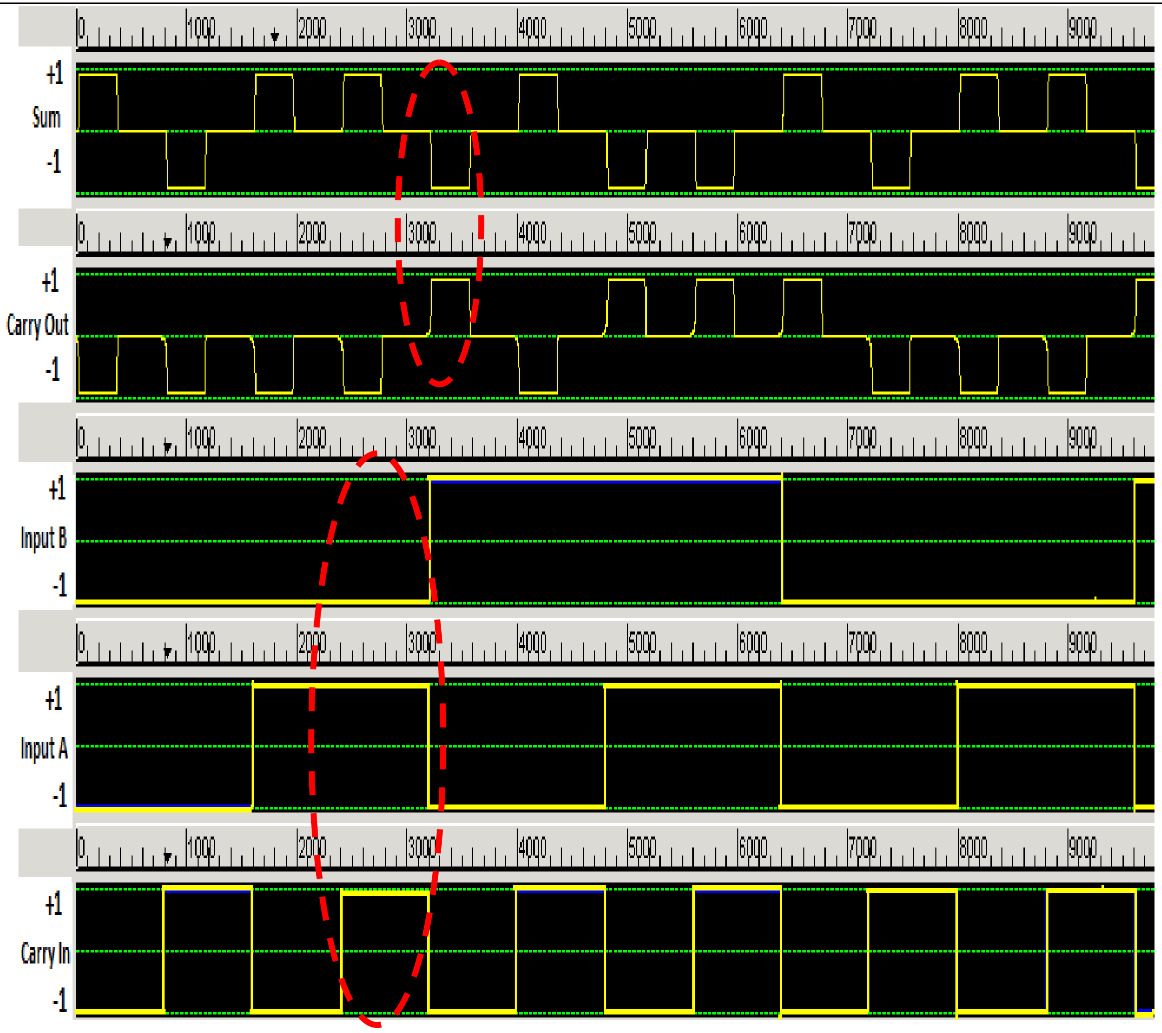

Figure 5-4: Simulation of Full Adder Circuit

5.2.2 D Flip-Flop

The next component required for the design of LAB is the D flip-flop. D flip-flop is used to store the output of LAB. Then this output is routed to interconnect for connection to the other LABs. Figure 5-5 shows the design of D flip-flop in QCA. The design for D flip is very simple. It is designed using some modification in SR flip-flop design. D flip-flop is the simplest of al flip-flops, which is why it is used in the design of LAB to reduce the complexity of design. Figure 5-6 shows the simulation of D flip-flop.

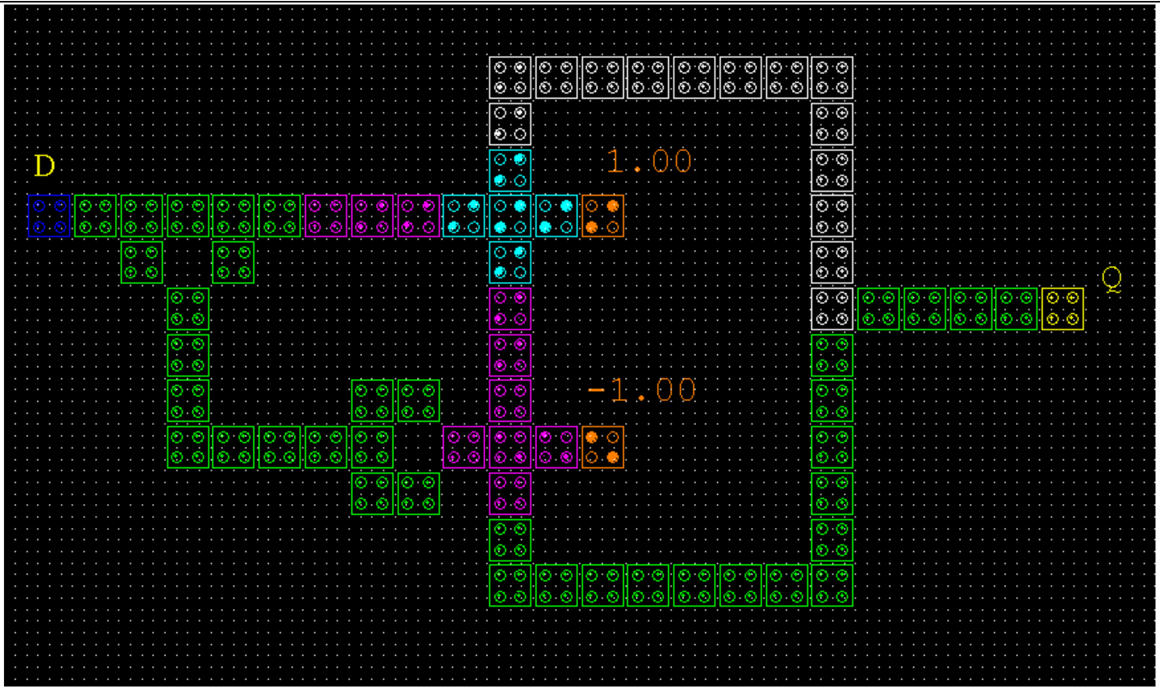

Figure 5-5: QCA Design of D Flip-flop

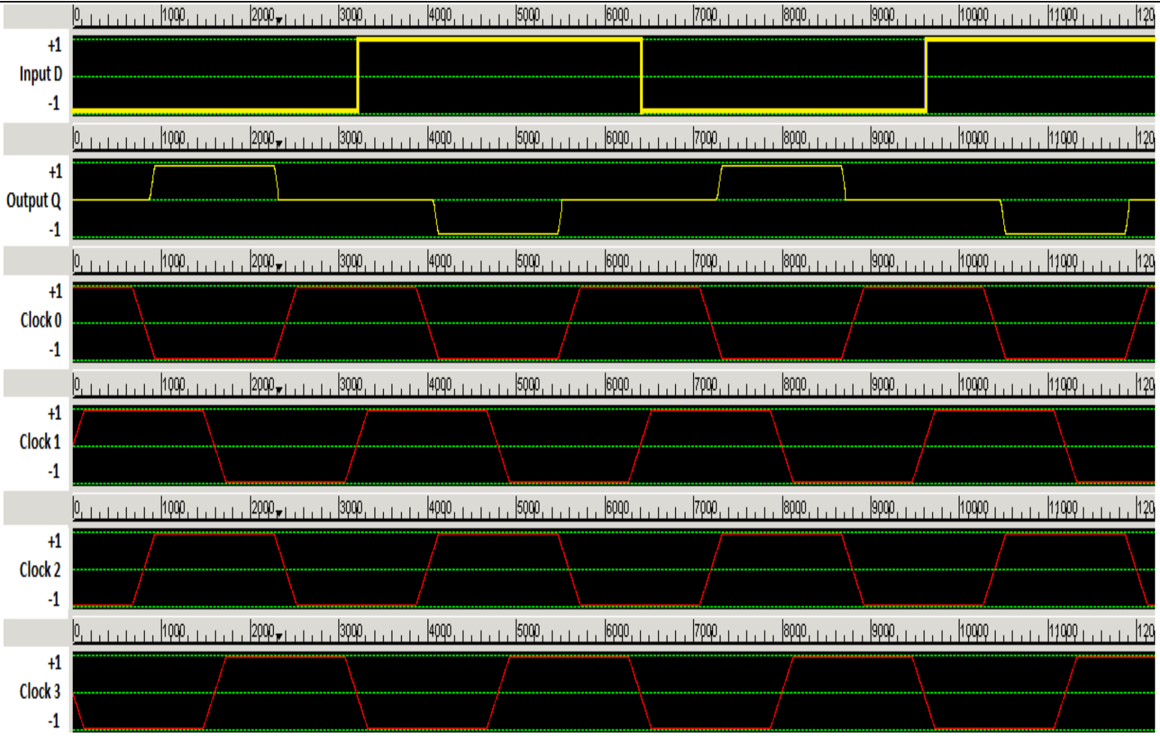

Figure 5-6: Simulation of D Flip-flop in QCA

5.3 Altera Stratix Logic Array Block in QCA

The basic building block of logic in the Stratix architecture, the logic array block (LAB), provides advanced features with efficient logic utilization. Each LAB contains a variety of look-up table (LUT)-based resources that can be divided between LUTs. With up to eight inputs to the LUTs, one LAB can implement various combinations of two functions. This adaptability allows the LAB to be completely backward-compatible with four-input LUT architectures. One LAB can also implement any function of up to six inputs and certain seven-input functions. In addition to the LUT-based resources, each LAB contains two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain, and a register chain. Through these dedicated resources, the LAB can efficiently implement various arithmetic functions and shift registers. Each LAB drives all types of interconnects: local, carry chain, shared arithmetic chain, register chain, and direct link interconnects. Figure 5-7 shows a high-level block diagram of the Stratix LAB. One LAB contains two programmable registers. For combinational functions, the register is bypassed and the output of the LUT drives directly to the outputs of the LAB. The LUT, adder, or register output can drive these output drivers independently. This allows the LUT or adder to drive one output while the register drives another output. This feature, called register packing, improves device utilization because the device can use the register and the combinational logic for unrelated functions.

Another special packing mode allows the register output to feed back into the LUT of the same LAB so that the register is packed with its own fan-out LUT. This provides another mechanism for improved fitting. The LAB can also drive out registered and unregistered versions of the LUT or adder output.

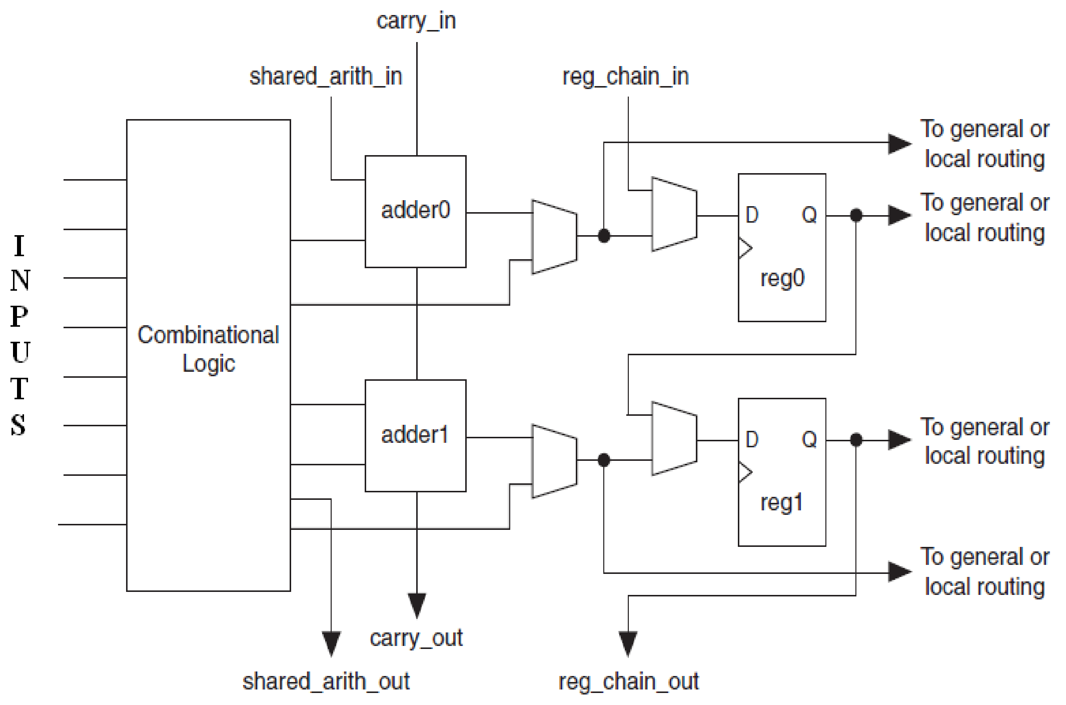

Figure 5-7: High Level Block Diagram of Stratix LAB [26]

Logic Array Block (LAB) is the basic logic structure in Altera Stratix architecture and is shown in Figure 5-7. The LAB consists of combinational logic block, adders, multiplexers and registers. A carry chain is used to connect multiple LABs in the structure. The Altera Stratix architecture in QCA uses less number of QCA cells and provides more functionality as compared to other implementations [12], [13], [14]. There are various modes in which the Stratix LAB can be operated including normal mode, extended LUT mode, arithmetic mode and shared arithmetic mode [26]. Figure 5-1 shows the block diagram of the LAB in the arithmetic mode. The details of this diagram can be found in reference [26]. In our research, we have implemented a logic array block in arithmetic mode. We have modeled all the necessary components required for the complete design of Altera Stratix LAB. This research presents a first Altera implementation of Stratix Logic Array Block (LAB) architecture using QCA technology along with simulation results. Briefly, an LAB in arithmetic mode uses two sets of two four-input LUTs along with two dedicated full adders. The four LUTs share the inputs. The carry-in signal feeds to adder0, and the carry-out from adder0 feeds to carry-in of adder1. This usage of the adder with the combinational logic output provides resource savings of up to 50% for functions that can use this ability [26].

The QCA model presented in this research is designed to operate the Altera Stratix LAB in the arithmetic mode. The arithmetic mode is ideal for implementing adders, counters, accumulators, and comparators. The arithmetic mode operation is not available in QCA designs presented in [12] and [14]. The combinational logic block in the LAB can be implemented using various combinations of LUTs [14]. This makes this design significantly more flexible and, as a result, more area efficient. This feature gives this design an advantage over the previous designs [12], [14]. In the CLB design shown in [14], the decoder output signals are used to decide which memory cell is active. It has two 2-to-4 decoders: a row decoder and a column decoder. These decoders generate the row and columns enable lines. This approach is similar to that used in typical CMOS memories. In QCA, each of the four decoder outputs has a different latency relative to the address inputs. The difference in latency between adjacent outputs is one clock cycle. Note that this matches the delay that a row or column line incurs traveling through a memory cell. The unit memory cells are arranged in a 4x4 array to form a sixteen-bit memory as shown in Figure 5-8. In Figure 5-8, the horizontal arrows are shorthand for the row decoder rails, the data input rails, the read/write enable rails and the OR chain rails. The vertical arrows represent the column decoder rails (except in the extreme right column, where they represent the vertical OR chain outputs). The clock cycle information on the arrows indicates the number of clock cycles required by an address input to reach the specified memory cell. As indicated in Figure 5-8, irrespective of the path adopted by the signal, the delay remains the same. This ensures throughput of one read per cycle. The total latency from when the decoded address arrives to when the output appears is nine clock cycles. At the bottom right of the LUT, there is the output of the LUT. It is the OR of the enabled outputs of all sixteen memory cells. The decoder ensures that only one memory cell is enabled. Thus, the output of the entire LUT is equal to the value stored in the enabled cell. The memory cell is designed to be used with separate row and column decoders as mentioned above. When ROW_IN and COLUMN_IN are both high, then only the memory cell is enabled. Note that each of these lines traverses four clock zones as it travels across the cell. This creates one clock cycle of latency traversing the cell in either direction. The memory presented in [14] is a modified version of the memory in [12]. This memory uses a serial Read/Write (RW) signal, a serial Data Input (DI) signal, a parallel Enable (EN) signal, and a parallel output (OUT) signal. The cell is designed such that multiple cells can be tiled side-by-side. Each row of cells then forms a shift register for writing. The QCA implementation of the LAB in arithmetic mode using QCADesigner is shown in Figure 5-9. The design basically consists of two sections: Combination of LUTs and Combination of Adders and Flip-Flops. The proposed design provides more functionality than [12], [14] with respect to the number of inputs used, logical operations, and arithmetic operations. The proposed LAB supports applications like addition, logic operations, and combinational logic operations which are not available in the hypothetical designs presented in [12], [14]. It should be noted that CLB in [14] is optimized version of CLB in [12]. CLB in [14] has used a set of row and column decoders to enable each of the 16 memory cells while in our design we are using only one decoder to enable the same number of memory cells. In addition, we have minimized the number of gates on the decoder and memory cell and have used a simple output circuit for the LUT as compared to [12], [14]. We have modeled the LUT using both coplanar and multi-layer crossover.

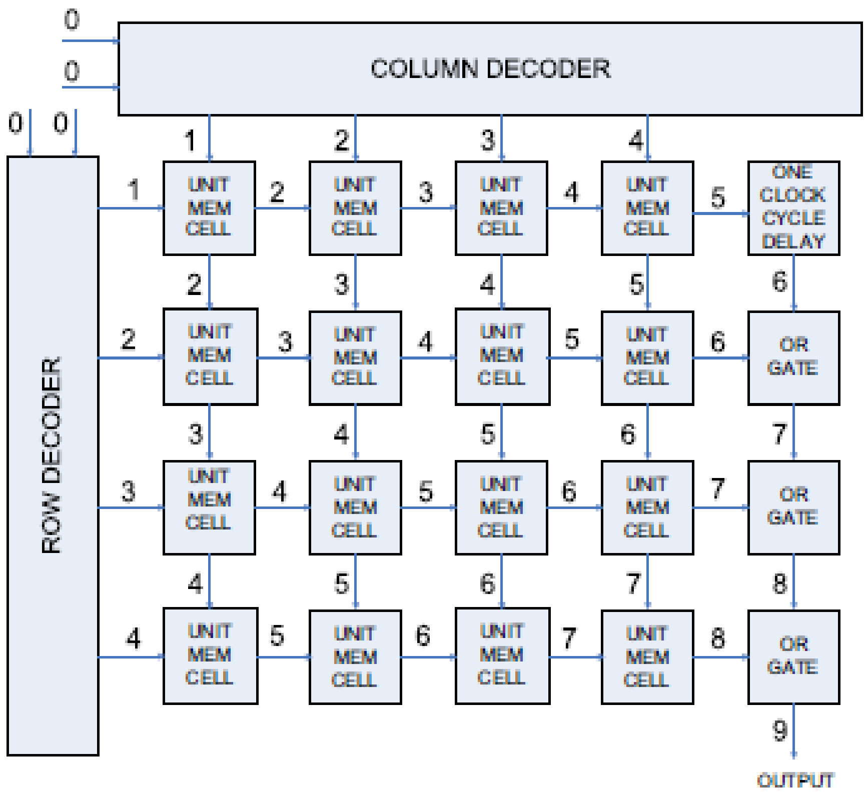

Figure 5-8: Block Diagram of Look-up Table with Memory Cells [14]

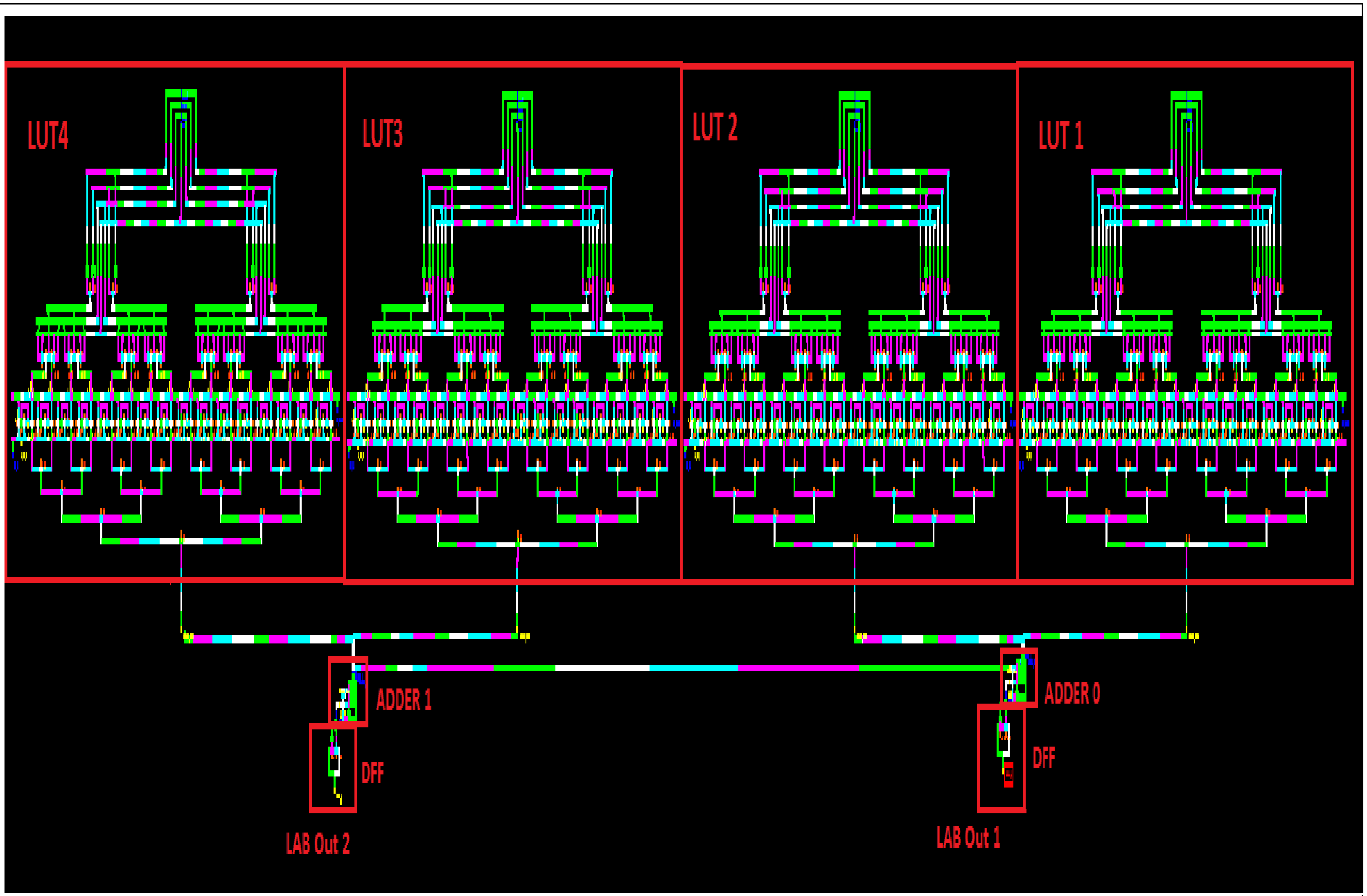

Figure 5-9: QCA Implementation of Altera Stratix LAB

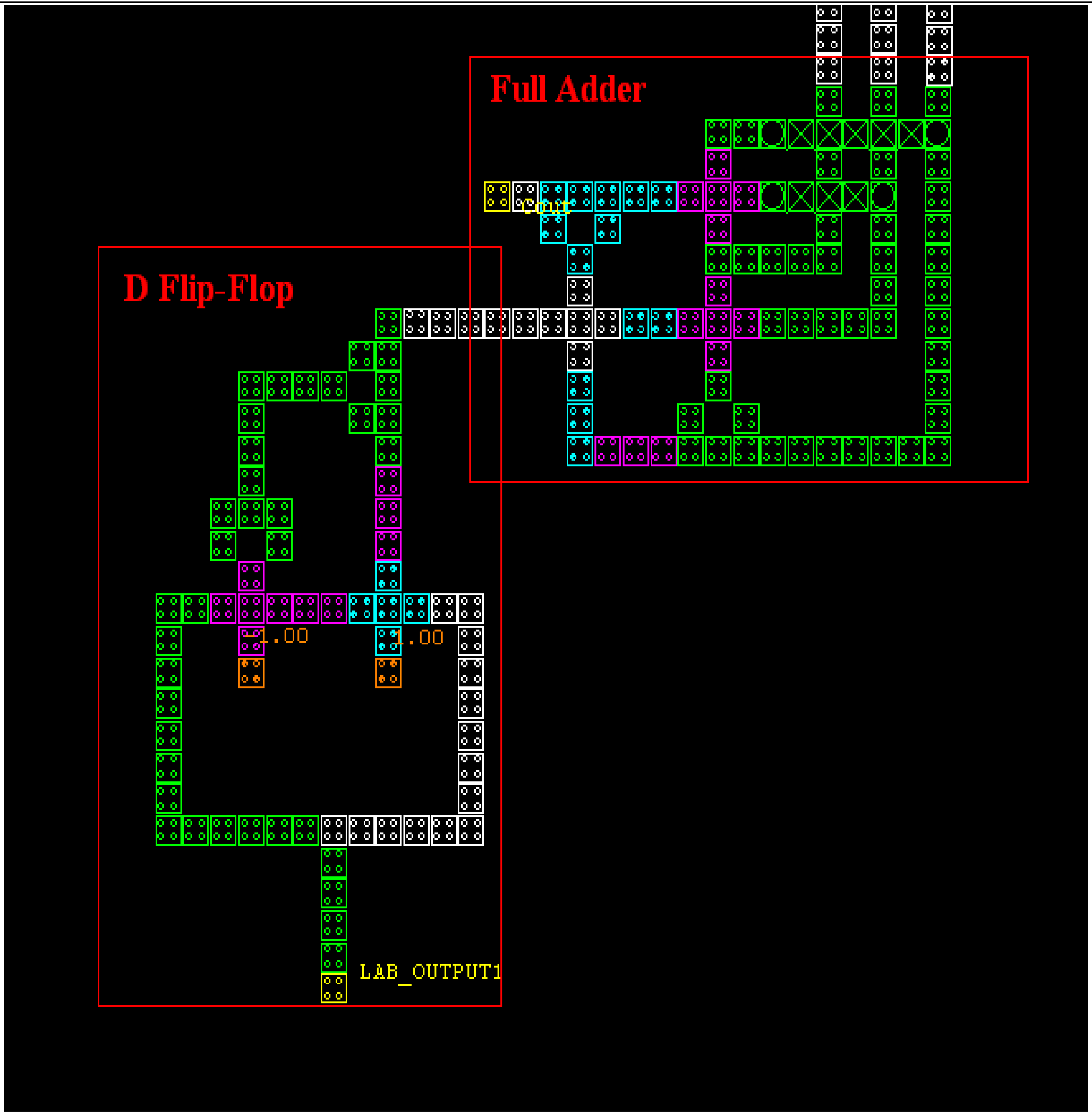

Figure 5-10: Partial Layout of QCA LAB showing Full Adder and D Flip-flop

Figure 5-10 is shown to visualize the full adder and D flip-flop components which are not clearly visible in Figure 5-9.

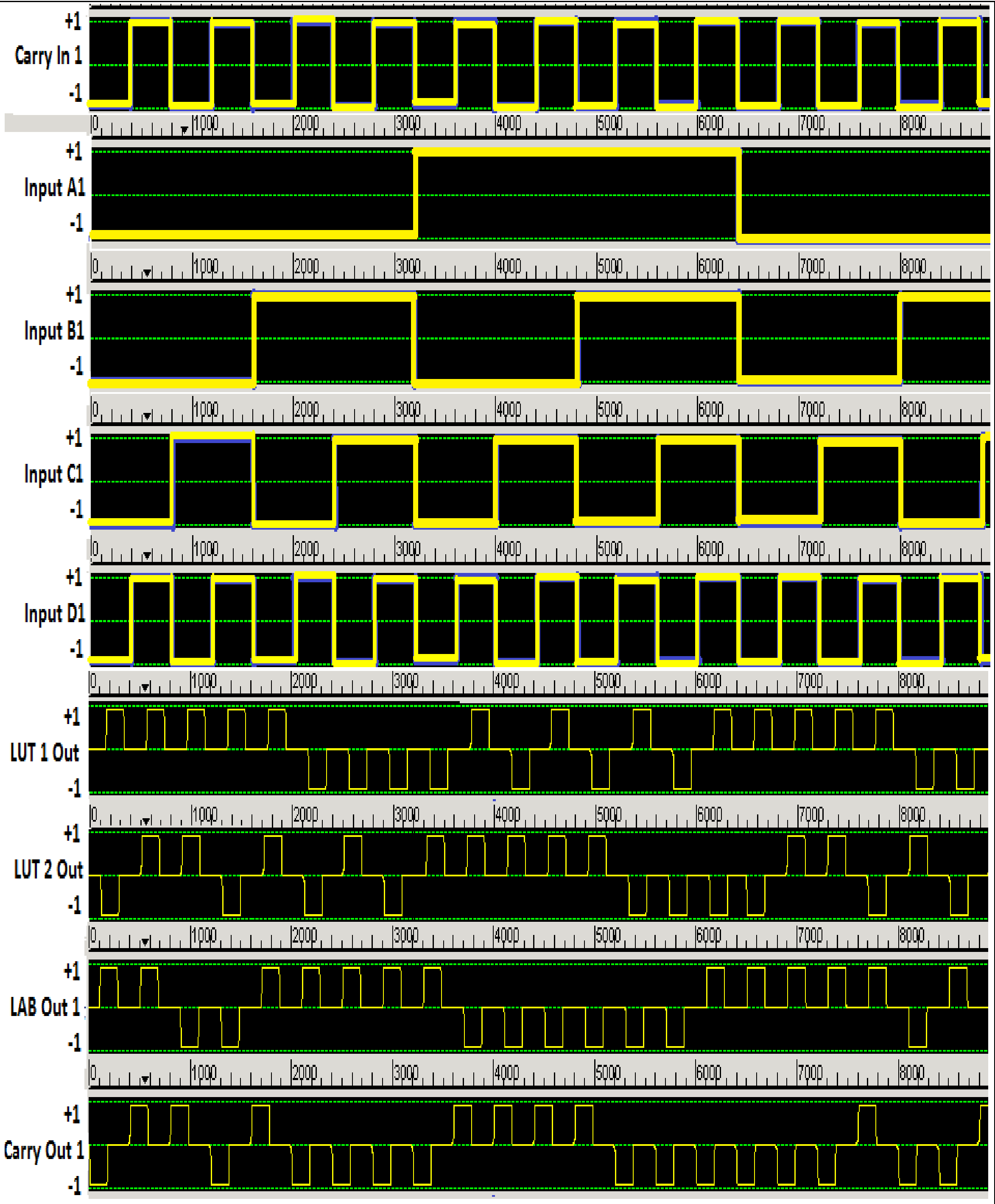

Figure 5-11: Simulation of Stratix LAB in QCA

Table 5.1: Truth Table for the LAB Operation

A	B	C	D	Carry_In 1	LUT 1 Output	LUT 2 Output	LAB 1 Output	Carry_Out 1
0	0	0	0	0	1	0	1	0
0	0	0	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1
0	0	1	1	1	1	0	0	1
0	1	0	0	0	1	0	1	0
0	1	0	1	1	0	0	1	0
0	1	1	0	0	0	1	1	0
0	1	1	1	1	0	0	1	0
1	0	0	0	0	0	1	1	0
1	0	0	1	1	0	1	0	1
1	0	1	0	0	1	1	0	1
1	0	1	1	1	0	1	0	1
1	1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	1	0
1	1	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	0

The QCA design of Altera Stratix LAB architecture is modeled and simulated using QCADesigner. For functional verification of the design, the proposed QCA implementation in Figure 5-9 has successfully been simulated to implement various equations in the LUT. The simulation shown in Figure 5-11 represents the following functions implemented in LUT 1 and LUT 2:

F (LUT1) = A'B' + BC'D' + ACD'

F (LUT2) = AC'D' + AB' + B'C'D + B'CD' + A'BCD

In the simulation results shown in Figure 5-11, we have verified the expected output of LAB for the addition of the outputs of LUT 1, LUT 2 and Carry_In1 as shown is Table 5.1. The detailed analysis of the design and comparison of this design to other designs is provided in Chapter 7.

Chapter 6

Design of Programmable Switch Matrix for QCA Applications

As discussed in Chapter 3, an FPGA is the topmost entity. Multiple logic array blocks are connected together to form an FPGA. FPGA consists of an array of identical configurable (programmable) logic array blocks (LABs), programmable I/O blocks (IOBs) and programmable interconnect structures. The interconnection between LABs is achieved using programmable interconnects. This chapter explains the design of a programmable interconnects. Section 6.1 gives a brief introduction in programmable interconnects. Section 6.2 describes the modeling of programmable interconnects in QCA.

6.1 Introduction to Programmable Interconnects

For an FPGA design, interconnects are very important as they bring connectivity between modules. Also interconnects are important because routing takes almost half of the design area. Programmable interconnects are also called programmable switches or programmable switch matrices. Figure 6-1 shows the representation of programmable switch matrix in an FPGA. There are mainly three types of switches: Basic programmable switches, Cross-point switches, and multiplexer switches [27]. A basic programmable switch consists of a transmission gate and a SRAM cell. The opening and closing of a transmission gate is dependent upon the truth table stored in the SRAM cell. The SRAM cell can be reprogrammed depending upon the need of application.

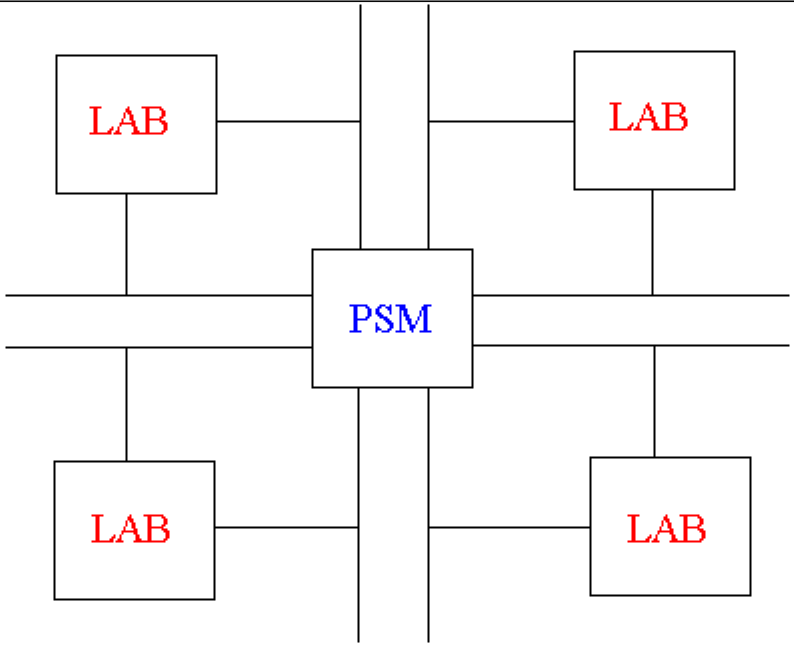

Figure 6-1: Representation of Programmable Switch Matrix in an FPGA

In electronics, a cross-point switch is also known as a crossbar switch, or matrix switch. It is a switch connecting multiple inputs to multiple outputs in a matrix manner. Originally the term was used literally, for a matrix switch controlled by a grid of crossing metal bars, and later was broadened to matrix switches in general. A Cross-point switch consists of basic programmable switches arranged in such a way that they form a network in which four LABs can be connected to each other located at North, South, East, and West directions. Figure 6-1 shows the Cross-point switch with connections for LAB in four directions. This type of programmable interconnect switch has been implemented in QCA technology for the QCA applications [13]. The author has modeled 4 X 4 cross-point switch in QCA using basic AND gate as a transmission gate. The operation of logic gates is controlled by programmable input to the gate. This design is very simple and easy to implement.

Multiplexer switches use multiplexer to switch the connections. A multiplexer is a device which selects one of several digital input signals and forwards the selected input into a single line. The interconnectivity between the LABs or CLBs is dependent upon the multiplexer select line. A truth table is programmed into the switch for selection of SELECT inputs. Thus, the multiplexer size can be selected depending upon the requirement of switch matrix.

6.2 Programmable Switch Matrix

To continue our research in designing an FPGA, there is a need of a programmable interconnect switch to connect LABs together to form an FPGA. Here we propose a interconnect switch designed in QCA technology using the basic gates designed earlier. The interconnect switch can be modeled as unidirectional or bidirectional switch.

6.2.1 Programming Grid

In modeling the programming grid, it is assumed that the input and output end points are placed at the grid in horizontal or vertical direction. Here, an assumption is made that the grid is a square [28].

As shown in Figure 6-2, four ports North, South, East, and West are placed in direction of net. A one to one connection is to be set between each of the end points. The programmable interconnect switch has been modeled by one to one mapping of a CMOS switch matrix in [29]. The designer in [29] has used one to one mapping technique to convert a CMOS programmable switch matrix into QCA technology.

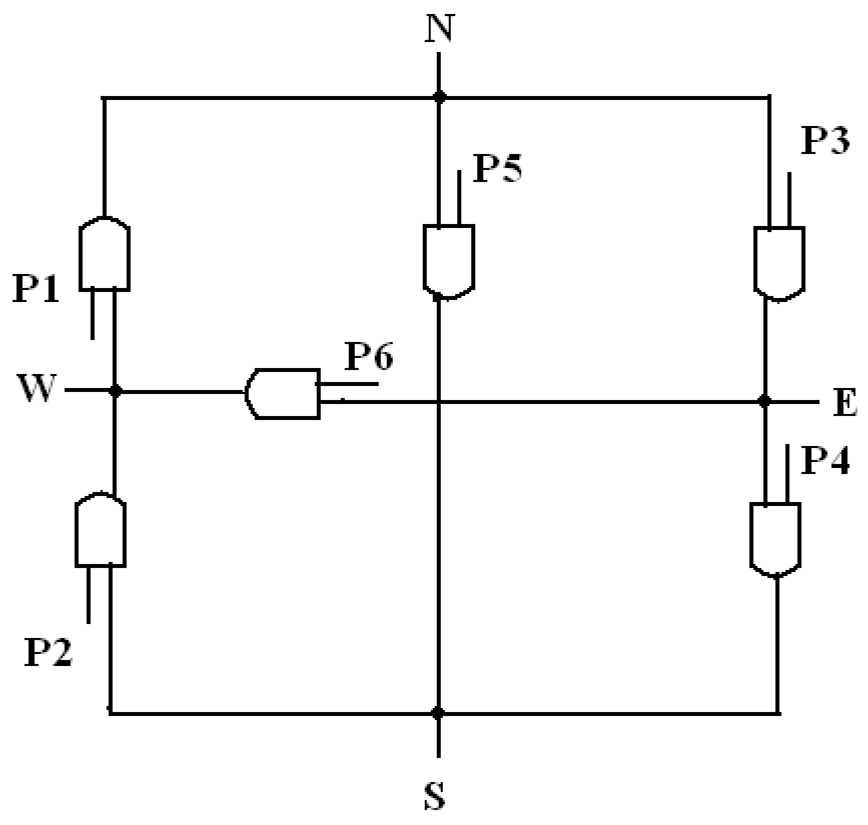

Figure 6-2 Programmable Switch Matrix

The mode of the grid is denoted by the pair of ports to which all the inputs and outputs are connected. The operation of logic gates is controlled by the programmable inputs to the gates. P1, P2, P3, P4, P5, and P6 are the programmable inputs to the gate. This design is very simple and it has limitations. For every programmable switch matrix (PSM), there are six programmable inputs. One PSM in [29] can connect 4 LABs only. The programming of the inputs is done by using AND gates. One of the input to the AND gate is programmable. If logic ‘1’ is fed to the gate, then it passes the other input value to through the gate.

6.2.2 Proposed Programmable Switch Matrix

The model shown in [29] has limitations such as limited routing of LABs. Therefore, we here propose a new design of programmable switch matrix.

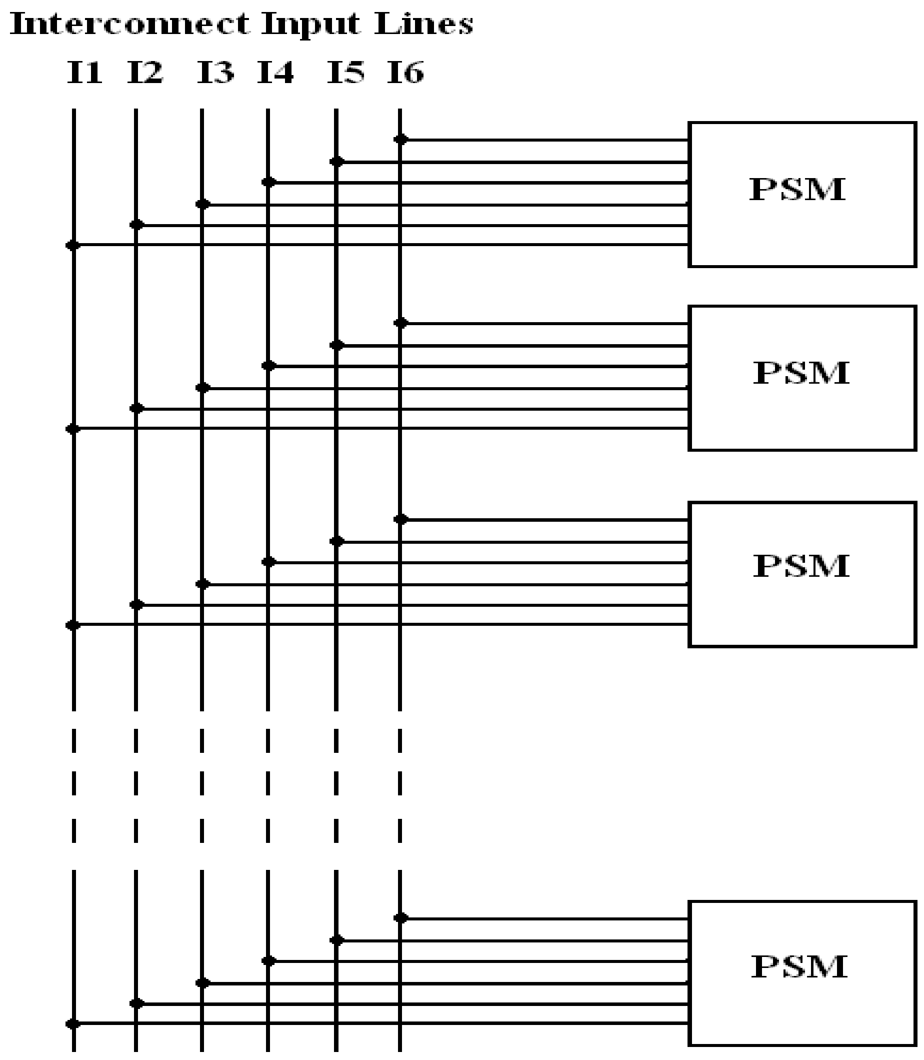

Figure 6-3: Proposed Design of the Programmable Switch Matrix

Figure 6-3 shows the block diagram of the proposed design for the programmable switch matrix. Figure 6-4 shows the gate level representation of programmable switch matrix. I1, I2, I3, I4, I5, and I6 are the interconnect lines. These lines provide input to the A1, A2, A3, A4, A5, and A6 AND gates. There is also a SELECT line which can select if we want the PSMs to be ON or OFF. We have selected a value '1' for SELECT inputs which is fed to A1, A2, A3, A4, A5, and A6 AND gates. Depending upon these input values, the PSM changes the routing of LABs. The output from A1, A2, A3, A4, A5, and A6 AND gates is then fed to the PSM AND gates. The benefit of having the interconnect lines is that, it can accommodate as many PSMs as needed. This design is very difficult in QCA because to the crossovers. We have used both coplanar crossover and multilayer crossover for the design of programmable interconnect switch matrix. For this research work, we have modeled only one PSM and the interconnect lines. Addition of PSMs can be done as per the requirement of the application. Table 6.1 shows the operation of PSM depending upon the inputs to the PSM. It incorporates all the combinations for interconnectivity. Figure 6-5 shows the QCA design of the PSM in QCADesigner. The simulation has been performed to verify all the combinations of ports. Figure 6-6 shows the simulation showing West port connecting to the North Port. Figure 6-6 shows the simulation showing North port connecting the East port.

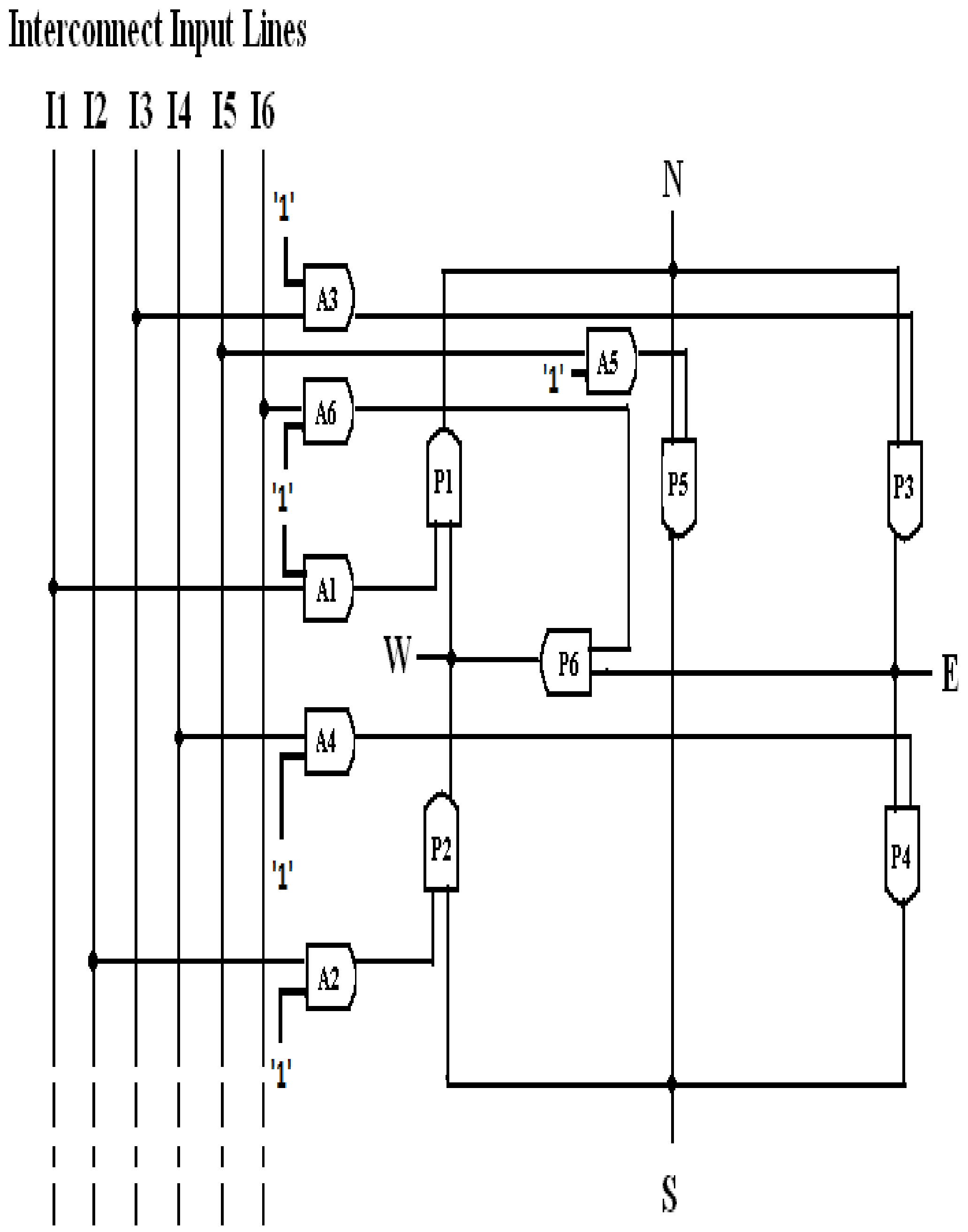

Figure 6-4: Gate Level Representation of Programmable Switch Matrix

Table 6.1: Port Connections according to Interconnect Lines

Port Connections	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**
WEST-NORTH	1	0	0	0	0	0
SOUTH-WEST	0	1	0	0	0	0
NORTH-EAST	0	0	1	0	0	0
EAST-SOUTH	0	0	0	1	0	0
SOUTH-NORTH	0	0	0	0	1	0
EAST-WEST	0	0	0	0	0	1
WEST-EAST	1	0	1	0	0	0
NORTH-WEST	0	0	1	0	0	1
EAST-NORTH	1	0	0	0	0	1
SOUTH-EAST	0	0	1	0	1	0
NORTH-SOUTH	0	0	1	1	0	0

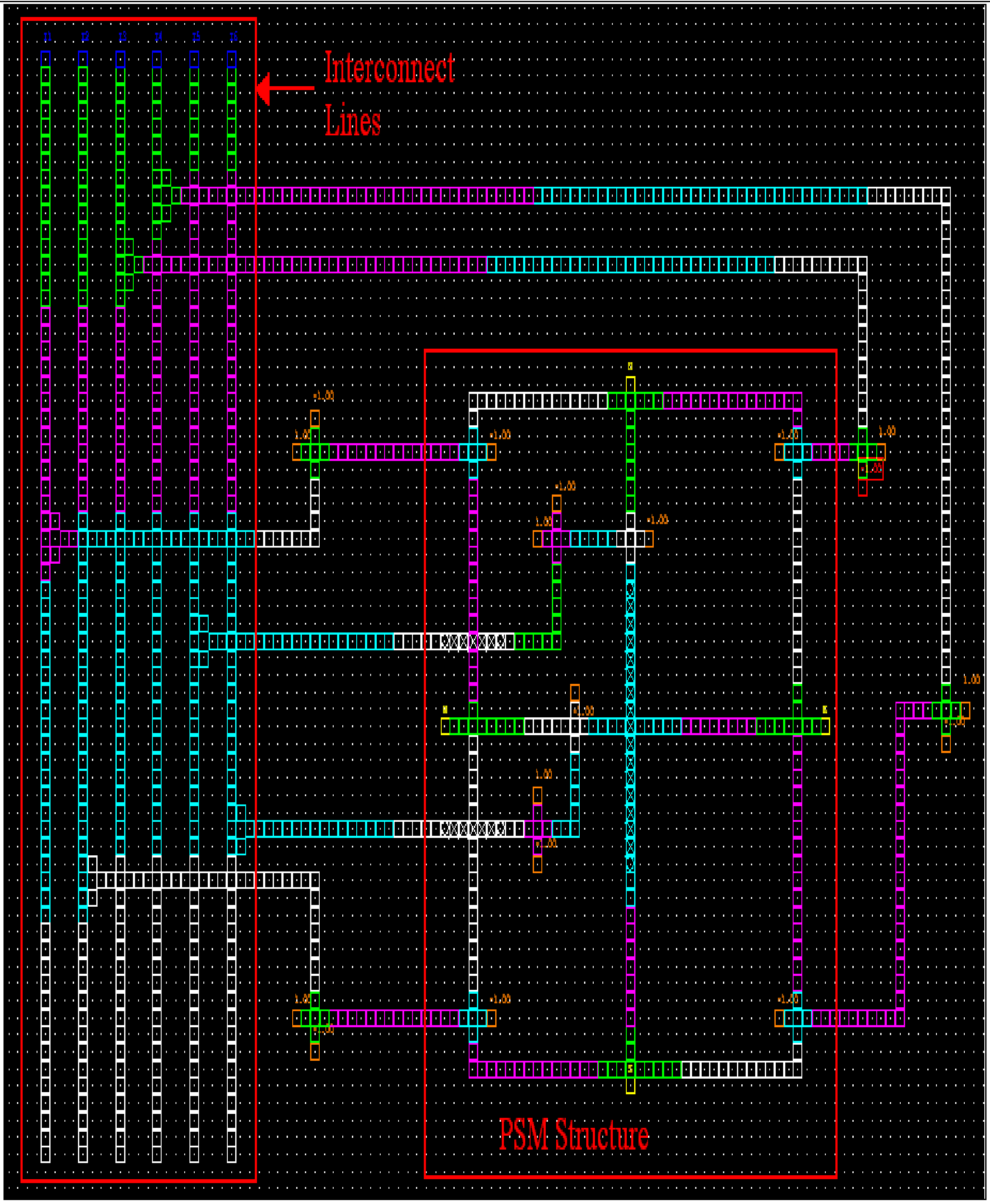

Figure 6-5: QCA Layout of Programmable Switch Matrix

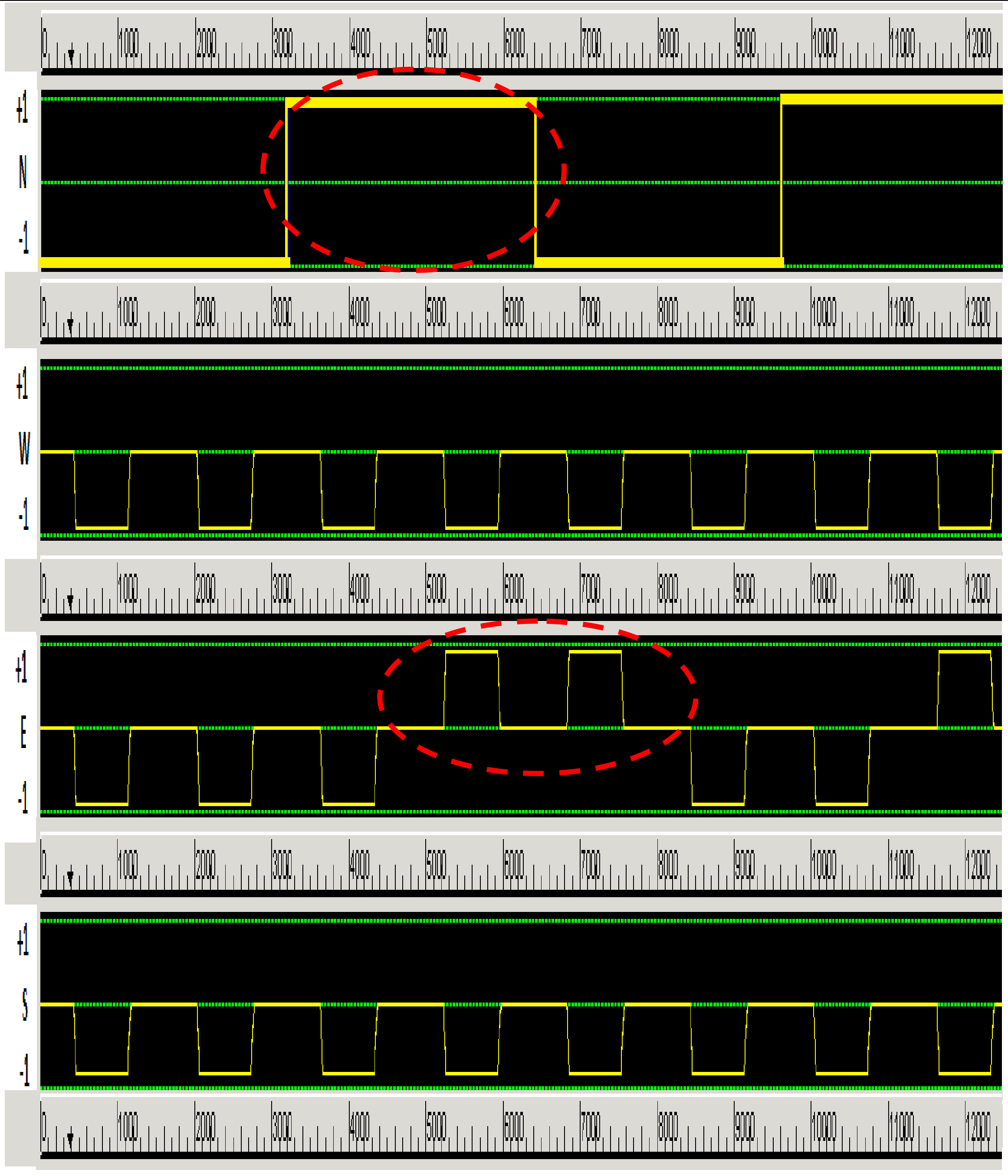

Figure 6-6: Simulation of PSM showing North Terminal connecting East Terminal

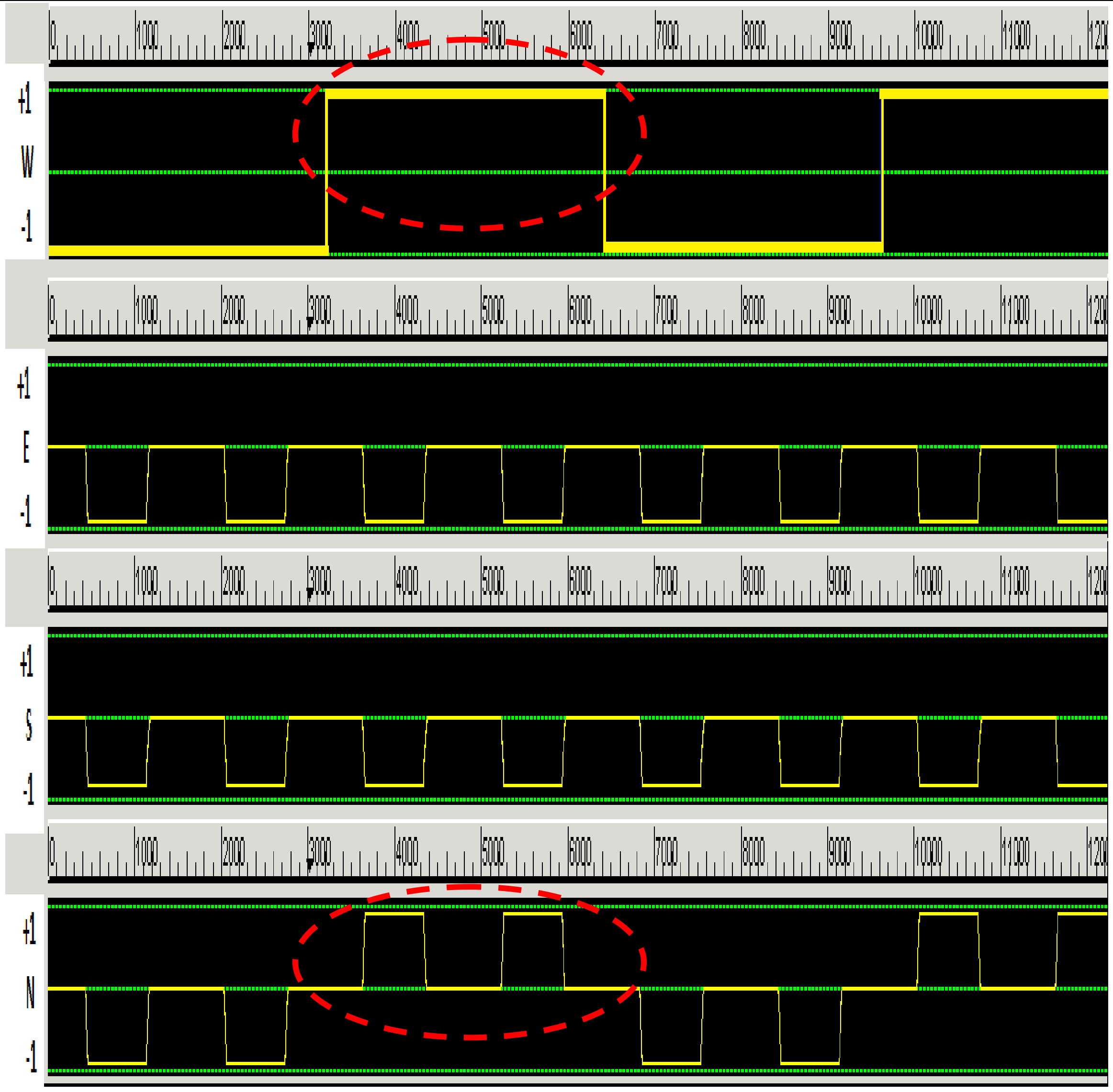

Figure 6-7: Simulation of PSM showing West Terminal connecting the North Terminal

Simulation results show that the design for the programmable interconnect switch matrix is successfully modeled in QCA. This design gives more routing flexibility than [29]. With this PSM design, we can connect many LABs together to form an FPGA.

Chapter 7

Design Analysis and Comparison

We have modeled the QCA design of Altera Stratix LAB and programmable switch matrix; however these deign has to be analyzed for their performance, functional verification and accountability in real world.

The first ever Altera Stratix logic array block is successfully designed and simulated at the quantum level using QCA technology. The LAB has been successfully modeled, implemented, and simulated using the QCADesigner tool. The simulated result of a LAB is verified with its expected outcomes to confirm the proper functioning of the LAB. The proposed architecture has also been optimized with respect to various design parameters like the number of QCA cells used to implement the logic circuit, the area occupied by the logic circuit and the overall latency of LAB.

7.1 Logic Array Block

The design parameters of various sub-components of the LUT have been tabulated in Table 7.1.

Table 7.1: Design Parameters of Sub-components of LUT

Component	**No. of QCA Cells**	**Area in nm^2**	**Latency**	**Contribution to overall Area (%)**	**Simulation Time (Seconds)**
Decoder	2818	913032	5	51.26	36
Memory Architecture	1838	595512	1	33.43	24
Output Circuit	842	272808	5	15.31	7

From Table 7.1, it can be seen that the decoder takes more QCA cells and it contributes more than 50 % of the QCA cells. As the number of QCA cells increase, the simulation time also increases.

The proposed architecture is compared with the existing architectures with respect to read latency, write latency and area. The existing CLB architectures are all based on design of a 4-input LUT. Table 7.2 shows the comparison of latency of various architectures with the proposed architecture. The proposed design provides more functionality than [12], [14] with respect to the number of inputs used, logical operations, and arithmetic operations. The proposed LAB supports applications like addition, logic operations, and combinational logic operations which are not available in the hypothetical designs presented in [12], [14]. It should be noted that CLB in [14] is optimized version of CLB in [12]. Also, the QCA LUT design presented in this paper has an advantage over the LUT design in [12], [14] that it uses less number of QCA cells. Also, the QCA LUT design presented in this research has an advantage over the LUT design in [12], [13], and [14] in that it uses less number of QCA cells.

QCA cell dimension of 18 nm and a dot size of 5 nm are used. Table 7.2 shows the detailed analysis of proposed LAB. Area acquired by each component and the respective latency contribution is shown in Table 7.2. Figure 7-1 shows the graphical representation of area acquired by each component of LAB. It can be seen that the decoder takes most of the area of QCA design. It is very essential that the decoder design has to be minimized. We have minimized the decoder design with respect number of majority gates used, crossovers, and routing of inputs. In previous designs, a multiplexer is used for the output circuit of the LUT which takes more QCA cells to design as compared to the simple output circuit for the LUT used in this design.

Table 7.2: Detailed Analyses of Components in Proposed LAB

Components	Area (μm^2)	No. Of Cells	Latency
Decoders	3.65	11272	5
Memory	2.36	7284	1
Output circuit	1.09	3368	5
D Flip-flop	0.045	140	2
Full Adder	0.0706	218	1
Routing inside the LAB	0.339	1045	3

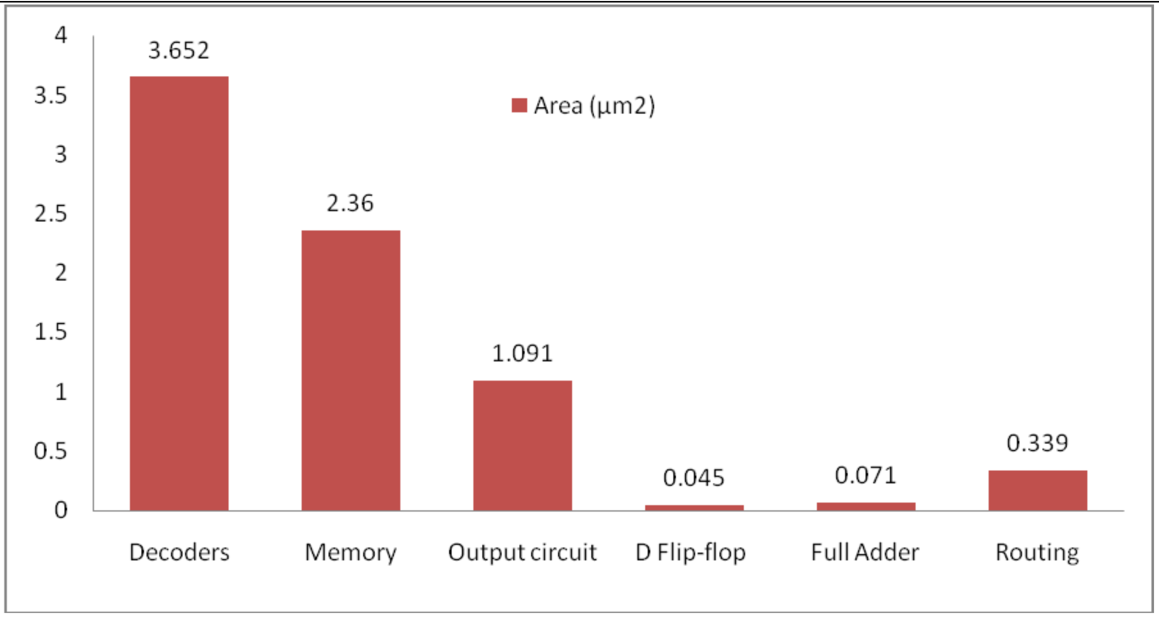

Figure 7-1: Graphical Representation of Area acquired by Components in LAB

Also, a latency pie diagram (Figure 7-2) is shown to represent latency contribution of each component. From this pie diagram, it is seen that the output circuit and the decoders contribute most of the latency in the LAB.

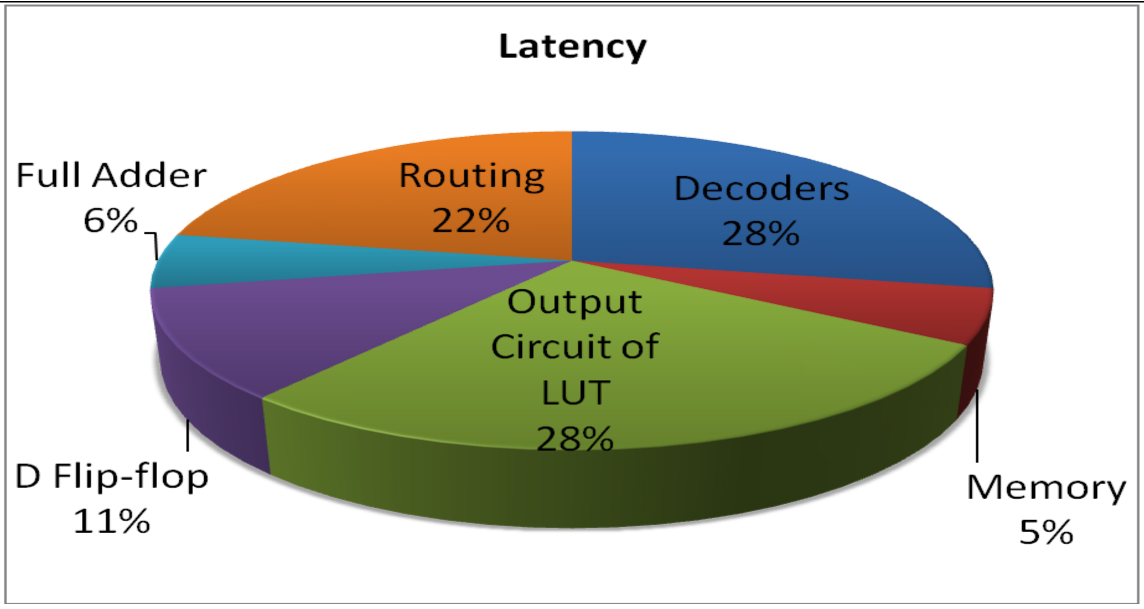

Figure 7-2: Latency Pie Diagram

A comparison of the area of the proposed architecture with other designs is shown in Table 7.3. A comparison of the area of the proposed architecture with other designs is shown in Table I. These numbers do not include QCA cells used for internal LAB routing. CLB in [14] has used a set of row and column decoders to enable each of the 16 memory cells while in our design we are using only one decoder to enable the same number of memory cells. In addition, we have minimized the number of gates on the decoder and memory cell and have used a simple output circuit for the LUT as compared to [12], [14]. We have modeled the LUT using both coplanar and multi-layer crossover. From Table 7.3, it is observed that the design presented in this research occupies less QCA cells and area than designs in [12], [14].

Table 7.3: Area Comparison of Proposed LAB

Design	Number of QCA Cells	Area (μm^2)
Proposed LAB	25K	8.094
CLB in [12]	203K	65.77
CLB in [14]	105K	34.02

The latency comparison with other designs is shown in Table 7.4 for various components. From Table 7.4, it is observed that the proposed design has less latency than the designs presented in [12], [13]. The latency is comparable to the design presented in [14]. However, design in [14] lacks ability to perform arithmetic operations and is suitable only for logic implementation.

Table 7.4: Latency Comparison of Proposed LAB

Components	Proposed LAB	CLB in [12]	CLB in [13]	CLB in [14]
Decoder	5	5	3	5
Memory & Output Circuit	6	6	9	9
D Flip-flop	2	-	2	-
Routing inside the LAB	3	19	4	-
Full Adder	1	-	-	-
Complete Design	17	30	18	16

From the results tabulated above, it can be inferred that the architecture of the proposed LAB with a 4-input LUT is better than the rest of the existing architectures [12], [13], [14] with respect to all design parameters. Similar to CMOS technology, QCA has the ability to be designed in multiple layers. A multilayer QCA can be used to further optimize the area of the proposed design. An earlier design by the authors of this paper in [30] adopts the strategy of multi-layer QCA design applied to a Xilinx Virtex FPGA. Although the multilayer approach reduces the area and the latency considerably, it increases the complexity of the design. In this design, area and latency has been reduced using a single layer approach. We are not able to compare our work with the work in [31] because it presents a programmable QCA device based on a 4-diamond circuit [8] which is a simple computational building block. Authors in [31] have used a MATLAB platform to simulate the QCA device.

7.2 Programmable Switch Matrix

The proposed PSM is modeled in QCA using QCADesigner. The proposed PSM is comprised of 1107 QCA cells. The Proposed PSM has a latency of 1 clock cycle. There has been some work on QCA interconnects in [7], [29], [32]. The authors in [7], [32] propose a different approach for interconnection between LABs and use QCA clock phases to route the signals in between the components. The author in [32] has proposed Universal Crossing Element (UCE) which is programmed at its fabrication. The UCE cannot be modified once fabricated. Its clock phases cannot be altered after fabrication. For UCE, the inputs have to be configured at the fabrication to set the direction and output. We cannot change the direction of flow of information once fabricated. However, setting up inputs to a particular clock phase is difficult from the fabrication point of view. Due to different phases of adjust cycles, the fabrication might not be simple. The authors in [32] are using a simulator based on Inter Cellular Hartree Approximation (ICHA) to design and simulate the UCE. The authors in [32] are not using QCADesigner to simulate the UCE. The proposed PSM is more flexible than [32] and it can accommodate more routing resources.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

Quantum-dot cellular automata is a novel technology with characteristics that differ substantially from CMOS. QCA offers an avenue toward very high density with low power requirements and minimal heat dissipation. The economic concerns of chip fabrication costs can be addressed by taking advantage of the self-assembling nature of the molecular world (Quantum Dots). In short, QCA is poised to breach the red brick wall of the ITRS roadmap [11].

This research presents the design, layout, and successful simulation of a LUT-based logic array block (LAB) using QCADesigner. To the best of our knowledge, this design is the first implementation of Altera Stratix LAB architecture. Results show that the proposed LAB occupies less area than [12], [13], [14]. Also, the latency is less than designs presented in [12], [13] and comparable to [14].

This research also presents the design, layout, and simulation of a programmable switch matrix (PSM) using QCADesigner. The simulation results are generated using the 'QCA Designer' software version 2.0.3. Bi-stable approximation is used to simulate the results presented in this research.

8.2 Contributions

The major contributions of this research are as follows:

1. Design of a parallel read write QCA memory cell.
2. Design of a LUT with less number of QCA cells than previous LUT designs.
3. First successful implementation of an Altera Stratix LAB architecture. Previous designs are based on Xilinx family architectures.
4. Design of a LAB which has more functionality than previous designs with respect to the number of inputs used, logical operations, and arithmetic operations.
5. Design of a LAB which has less area and latency than the previous designs mentioned in this work.

8.3 Future Work

Some of the future work related to this research are as follows:

- Similar to CMOS technology, QCA has the ability to be designed in multiple layers. A multilayer QCA can be used to further optimize the area of the proposed design.
- The LABs and programmable interconnects presented in this work can be integrated to implement an FPGA for digital applications in the nano-electronics industry.
- In this work, the proposed LAB is transformed from the gate level to the QCA logic using a one-to-one mapping which often does not exploit the advantages of the QCA technology. Therefore, optimization during the translation of the gate level to the MV logic by eliminating redundant logic can be investigated.

- The QCADesigner tool has many limitations. Further research is needed to develop other tools which will give the user better control in the design, layout, and simulation of QCA circuits.

References

[1] K. Walus, Wei Wang, and G. Jullien, " Quantum-Dot Cellular Automata Adders," IEEE Transactions in Nanotechnology, vol. 2, pp. 461-464, August 2003.

[2] K. Walus, Wei Wang, and G. Jullien, " A Method of Majority Logic Reduction for Quantum Cellular Automata," IEEE Transactions in Nanotechnology, vol. 3, issue 4, pp. 443, December 2004.

[3] C. Lent, P. Tougaw, and W. Porod, "Bistable Saturation in coupled Quantum Dots for Quantum Cellular Automata," Appl. Phys. Lett., vol. 62, pp. 714-716, February 1993.

[4] C. Lent, P. Tougaw, W. Porod, and G. Bernstein. "Quantum Cellular Automata," IEEE Journal on Nanotechnology, vol.4, pp. 49–57, December 1992.

[5] S. Brown and Z. Vranesic, "Fundamentals of Digital Logic with VHDL Design," 2nd Edition, McGraw-Hill, pp. 904-914, 2005.

[6] W. Wolf, "FPGA-Based System Design," Upper Saddle River, NJ, Prentice Hall PTR, 2004.

[7] M. Niemier, A. Rodrigues, and P. Kogge, "A Potentially Implementable FPGA for Quantum Dot Cellular Automata," First Workshop on Non-Silicon Computation (NSC-1), Boston, MA, May 2002.

[8] M. Niemier and P. Kogge, The "4-Diamond Circuit — a minimally Complex Nano-scale Computational Building Block in QCA," Proceedings of IEEE Computer society Annual Symposium, pp. 3-10, February 2004.

[9] K. Walus, T. Dysart, G. Jullien, and R. Budiman. "QCADesigner: A Rapid Design and Simulation Tool for Quantum-dot Cellular Automata," IEEE Transactions in Nanotechnology, vol. 3, issue 1, pp. 26-31, March 2004.

[10] QCADesigner, http://www.atips.ca/projects/qcadesigner/

[11] International Technology Roadmap for Semiconductors, http://www.itrs.net/Links/2009ITRS/2009Chapters_2009Tables/2009_ExecSum.pdf, 2009.

[12] T. Lantz and E. Peskin, "A QCA Implementation of a Configurable Logic Block for an FPGA," Proceedings of the Third International Conference on Reconfigurable Computing and FPGAs (ReConFig 2006), pp. 132-141, September 2006.

[13] Puneeta Bhadsavle, " Modeling and Testing of FPGA LUT at the Nano Scale using Quantum-dot Cellular Automata," Master's Thesis, Department of Electrical Engineering and Computer Science, University of Toledo, February 2008.

[14] Chia-Ching Tung, Ruchi Rungta, Eric Peskin. " Simulation of a QCA-based CLB and a Multi CLB Application," Proceedings of International conference on Field-Programmable Technology, pp. 62-69, December 2009.

[15] K. Walus, A. Vetteth, G. Jullien, and V. Dimitrov. "RAM Design using Quantum-dot Cellular Automata," Technical Proceedings of the Nanotechnology Conference and Trade Show, vol. 2, pp. 160-163, April 2003.

[16] M. Ottavi, V. Vankamamidi, F. Lombardi, S. Pontarelli, and A. Salsano, “Design of a QCA Memory with Parallel Read/Serial Write,” Proceedings of IEEE Computer Society Annual Symposium on VLSI: New Frontiers in VLSI Design (ISVLSI’05), pp. 292-294, May 2005.

[17] V. Vankamamidi, M. Ottavi, and F. Lombardi, “Tile-based Design of a Serial Memory in QCA,” Proceedings of 15th ACM Great Lakes symposium on VLSI (GLSVSLI ’05), pp. 201-206, 2005.

[18] S. Frost, A. Rodrigues, A. Janiszewski, R. Rausch, and P. Kogge, “Memory in Motion: A study of Storage Structures in QCA,” First Workshop on Non-Silicon Computation (NSC-1), May 2002.

[19] D. Berzon, and T. Fountain, “A Memory Design in QCAs using the Squares Formalism,” IEEE Computer Society's Ninth Great Lakes Symposium on VLSI (GLS ’99), pp. 166, March 1999.

[20] E. Blair, and C. Lent, “Quantum-dot Cellular Automata: An Architecture for Molecular Computing,” Simulation of Semiconductor Processes and Devices, SISPAD International Conference, pp. 14-18, September 2003.

[21] K. Hennessy, and C. Lent. “Clocking of Molecular Quantum-dot Cellular Automata,” Journal of Vacuum Science & Technology: Microelectronics and Nanometer Structures, vol. 19, pp. 1752-1755, September 2001.

[22] C. Lent, M. Liu and Y. Lu. “Bennett Clocking of Quantum-dot Cellular Automata and the Limits to Binary Logic Scaling,” IEEE Journal on Nanotechnology, vol. 17, pp. 4240-4251, 2006.

[23] V. Vankamamidi, M. Ottavi, F. Lombardi, “Two-Dimensional Schemes for Clocking/Timing of QCA Circuits,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems vol. 27, issue 1, pp. 34-44, January 2008.

[24] J. Huang, M. Momenzadeh, F. Lombardi, “Design of Sequential Circuits by Quantum-dot Cellular Automata,” Microelectronics Journal vol. 38, Issues 4-5, pp. 525-537, April-May 2007.

[25] K. Walus, G. Schulhof and G. Jullien, “High Level Exploration of Quantum-dot Cellular Automata,” Proceedings of Asilomar Conference on Signals, Systems and Computers, pp. 30-33, November 2004.

[26] Altera Stratix Device Handbook, vol. 1 ver. 4.4, pp. 2.7- 2.21, July 2009.

[27] H. Michinichi, "A Test Methodology for Interconnect Structures of LUT-based FPGAs," Proccedings of IEEE 5th Asian Test Symposium, Computer Society Press, November 1996.

[28] W. Huang, X. Chen and F. Lombardi, “On the Diagnosis of Programmable Interconnect System: Theory and Application,” 14th IEEE VLSI Test Symposium, pp. 204-209, April-May 1996.

[29] Amrutha Asthana, “ Modeling and Testing of QCA Circuits Including FPGAs,” Master’s Thesis, Department of Electrical Engineering and Computer Science, University of Toledo, 2008.

[30] M. Niamat, S. Panuganti, T. Raviraj, “QCA Design and Implementation of SRAM based FPGA Configurable Logic Block,” Proceedings of Circuits and Systems (MWSCAS), 53rd IEEE International Midwest Symposium, pp. 837-840, August 2010.

[31] R. Devadoss, K. Paul, M. Balakrishnan, “A Tiled Programmable Fabric using QCA,” Proceedings of International Conference on Field-Programmable Technology (FPT), pp. 9-16, December 2010.

[32] A. Jazbec, N. Zimic, I.L. Bajec, P. Pecar, M. Mraz, “Quantum-dot Field Programmable Gate Array: Enhanced Routing,” Proceedings of Conference on Optoelectronic and Microelectronic Materials and Devices, pp. 121-124, December 2006.