A Design for Testability Scheme for Modular and Non-modular Quantum Dot Cellular Automata (QCA) Employing Stuck-at Fault Model

Sayeeda Sultana

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in partial fulfillment of the requirements
for the degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

September 2006

Your file Votre référence
ISBN: 978-0-494-20755-0
Our file Notre référence
ISBN: 978-0-494-20755-0

ABSTRACT

A Design for Testability Scheme for Modular and Non-modular Quantum Dot Cellular Automata (QCA) Employing Stuck-at Fault Model

Sayeeda Sultana

Today leading VLSI experts predict a hard wall for CMOS and other conventional fabrication technology due to fundamental physical limits (ultra-thin gate oxide, short channel effects, doping fluctuations, etc.), and increasingly difficult and expensive lithography in nanoscale. Extensive research conducted in recent years at nanoscale aiming to surpass CMOS has proposed Quantum Dot Cellular Automata as a viable alternative for nanoscale computing.

Quantum Dot Cellular Automata (QCA) paradigm is an innovatory approach to computing, which encodes binary information by means of charge configuration of nanostructures instead of current switching devices. The fundamental building block of QCA devices is the QCA cell, and electrostatic interaction between neighboring cells governs the design of all QCA wires and logic gates. The two primary logic elements in QCA technology are: majority voter and inverter. Binary wires and inverter chains are used for interconnection purposes. Logic operation AND, and OR can be achieved by maneuvering inputs to the majority voter. Clocking enables precise control over timing and data flow direction, as well as power gain in QCA circuits. Also proper clocking can achieve computational pipelining and can drastically reduce circuit power dissipation.

Manufacturing of a QCA cell is expected to result in defects like cell displacement, misalignment, and absence of cell or additional cell in circuitry, causing the circuit to exhibit faulty behavior. So a well-defined testing scheme becomes necessary for this technology. Though the technology is different from conventional CMOS design, it is shown to be effective and realistic to use existing testing schemes at this stage. Stuck-at (s-a-v) fault model is quite acceptable in this regard in spite of the fact, that this model does not incorporate all the defective behaviors occurring in the fabrication process. With this in view, single stuck-at value faults have been considered for testing QCA circuits.

In this thesis a new strategy for designing QCA logic, exhaustively testable for single s-a-v faults, is presented. In particular, the method facilitates QCA functionality testing. Any combinational logic can be implemented using only AND-OR gates (with negated signals available), and in QCA this generally results in reduced test set for exhaustive fault detection within the data path. Previously this strategy was used for QCA logic testing considering only primary inputs (either true or complemented, but not both) feeding different majority voters, which fails for general circuits where fanouts are allowed for primary inputs and their complement. Here, a design scheme has been proposed which makes testing possible for any combinational QCA circuit. The extension to modular design testing is also presented.

Two design approaches are proposed for testing modular and non-modular logic. The first design uses $2n$ ($n \equiv$ primary inputs) '*Test Enable*' majority voters, and is tested with two 4-bit vectors regardless of complexity of design and the input size. Second design employs n majority voters for the same purpose, thus requiring lesser number of majority voters, but at the price of increased vector length. Application specific conditions would decide which design becomes optimal. Without going into the features of a particular QCA fabrication, errors on logic level is addressed, such that the approach achieves generality, and could be applied to any particular implementation of QCA. Also to overcome the fault masking in modular circuit design, a solution has been presented. To verify the scheme, a simulation and layout tool, QCADesigner version 2.0.3 was used. First the fault free circuit was designed and simulated. Then random s-a-v faults were injected in different locations of data path. In all cases, 100% fault coverage was achieved confirming the validity of proposed approach.

ACKNOWLEDGEMENT

I would like to take this opportunity to thank some of the people without whose support this thesis would not have achieved.

I would first like to express my gratitude towards my supervisor Dr. Katarzyna Radecka, who has guided me towards this achievement. She took the time and effort to guide me to finish this project successfully. As my supervisor, she always had time for me and my queries and her knowledge in the subject matter has refined me from time to time. Her encouragement towards this topic and her support, both academic and financial has kept me aiming to excel. I remain ever grateful to her for giving me the opportunity to continue my work in flexible hours after my first baby's birth. If not for her, I doubt I would have come this far.

A special acknowledgement right after my supervisor must be made to Dr. Zeljko Zilic of McGill University, who has extended his helping hands towards me in many ways. His prompt steps regarding any problem in my research have guided me to come to a fruitful end.

I would also like to thank the faculty of the department, specially the teachers with whom I have taken course. I would like to thank Dr. Asim J. Al-Khalili and Dr. Sofiene Tahar for being so supportive during my courses with them and afterwards. I would like to take a moment in remembrance of Dr. Victor Rossokhaty, with whom I did my first course here in Concordia University. His affectionate smile encouraged me to pursue my studies in this field.

At this point I have to thank my family and friends, whose love and affection kept me going. I am grateful to my parents for believing in me and being there always for me, sheltering from whatever and whenever, and mention specially my husband, for being so supportive and understanding during this period when he had his tough time with his thesis. He had been my support and the source of my energy for pursuing this research, and beyond. He has been weathered in the process and still had the energy to absorb all the hardships and have a smile for me. Thanks to my daughter, who was born during this time, for being so patient seeing her parents busy with their thesis, and providing me with the greatest joy of my life. Holding her in my arms, and looking at her face has been worth living for.

Thank you all.

Sayeeda Sultana

Table of content

List of figures

Chapter 1: Introduction

The age of Very Large Scale Integrated (VLSI) circuit fabrication is now about half a century. In this long journey, microelectronics industry has switched from planner to lateral devices, and adopted Complementary Metal Oxide Semiconductor (CMOS) instead of Bipolar Junction Transistor (BJT), enjoying dramatic improvements in the speed and size of electronic devices. This trend has long obeyed Moore's law, which predicts that the number of devices integrated on a chip will be double every 18 months. Adherence to this exponential growth curve has been a monumental task requiring rapid improvements in all aspects of integrated circuit fabrication to permit manufacturers to both shrink the size of the devices and increase chip size, while maintaining acceptable yields. Experts' ardor for yet more speed and miniaturization has brought them to the verge of classical devices, forcing them to peek in an unknown and astonishing world of nanotechnology and quantum devices.

Today's leading VLSI experts predict a hard wall for the CMOS and other conventional fabrication technology in about a decade. This lithography-based technology is facing serious challenges due to the fundamental physical limits of the CMOS technology such as ultra-thin gate oxides, short channel effects, doping fluctuations and the increasingly difficult and expensive lithography in nanoscale regimes. Future trend of the National Technology Roadmap for Semiconductors (NTRS) predicts that it may reach its extreme end as early as 2012 as shown in figure 1 [1]. It then becomes prudent to look for an alternate paradigm for information processing which will overcome the physical limitations of transistor based designs.

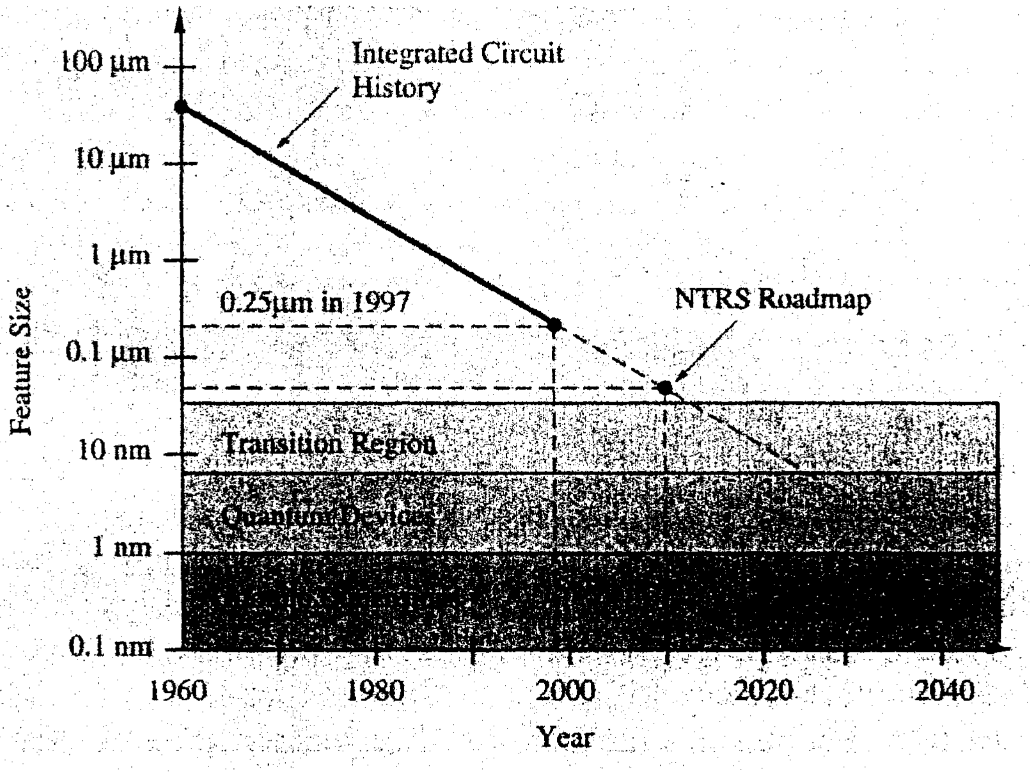

Figure 1: NTRS roadmap for devices [1].

Extensive research conducted in recent years at nano-scale aiming to surpass the conventional CMOS technology has anticipated that these devices might achieve a density of 10^{12} devices/cm^2, and operate in THz frequencies [2]. Among such devices, *Quantum Dot Cellular Automata* (QCA) proposed originally by Dr. Craig Lent [3] offers a new and extremely low powered method of computation. In terms of feature size, it is projected that QCA cell of few *nm* size would be possible in near future [2].

In quantum physics, basic particles such as electrons are associated with wave nature. The principle of uncertainty imposes on such particles to have a minimum positional spreading. The so-called atomic Bohr radius then becomes the minimal size of an electron. Whenever one creates a structure with all three dimensions less than that of the atomic Bohr radius of an electron, the electron in such structure becomes confined, its wave function overlaps with itself, and energy quantization occurs. Such 3D confined structures are called quantum dots. These quantum dots has been fabricated and demonstrated in various materials, and as shown in this thesis, have been used for devising computation machines. A quantum dot structure is shown in figure 2.

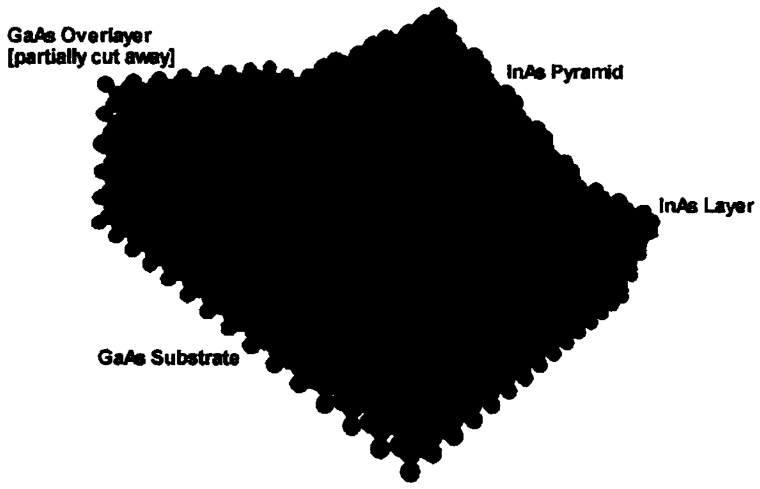

Figure 2: A possible Quantum dot structure as used in QCA fabrication.

QCA is an innovative device that stores logic states not as voltage levels, but rather in the relative position of individual electrons. The Coulomb interaction between stationary electrons provides the basis for a current-less (no electron flows and hence no current is drawn from the source) computation and data processing. QCA offers the possibility for circuitry to dissipate many orders of magnitude less power than CMOS, is scalable to molecular dimensions, and provides the power gain necessary to restore signal levels. No voltage and current based complicacies are faced; rather somewhat logical reversibility can be reached. In fact, for a very low power overhead, it is possible to eventually reach the absolute reversible logic [4].

There has been considerable research over the past decade on designing combinational and sequential logic devices using QCA cells, as well as finding the suitable fabrication process. In current technology the cells must be aligned precisely at nanoscales to provide correct functionality [2]. Thus proper testing for manufacturing defects and misalignment plays an important role for quality of QCA-based designs. Recent developments in cell manufacturing involving the deposition of molecules on a substrate surface by self-assembly process [5] have indicated that missing or additional cells are inevitable for molecular implementation because the process of cell deposition is very sensitive [6]. A small variation in process parameters may result in a defect. Moreover, these defects have pronounced functional effects when they occur within, or very near to the target device due to strong cell interactions [7]. Therefore testing is an absolute necessity for detecting various types of defects in QCA like cell misplacement, presence or absence of cell etc.

1.1 Motivation

One of the primary issues in the testing community now a day has been the drastic swing in fabrication technology and its effect on test flow. Since the manufacturing process of nano-devices is still ill-defined, it is extremely difficult to address related manufacturing testing problems. However it would be inappropriate to ignore testing of these devices until manufacturing processes are well developed. Instead the trend is to rely on the vast knowledge of testing of CMOS circuits, in attempts to modify and adjust the research from the CMOS world to QCA. It seems that in many cases faults in circuits fabricated in nano-technology exhibit similar properties to fault models of classical

CMOS circuits [8]. Even faults, which exhibit probabilistic nature typical to quantum circuits, can be successfully considered using adjusted classical methods.

The basic logic elements in QCA technology are the majority voter and inverter. Binary wires and inverter chains are used as an interconnect fabrics. Logic elements AND and OR can be obtained by manipulating the majority voter [9]. Though the technology is different from the conventional CMOS designs, it is effective and realistic to use the existing testing scheme at this moment. Over the years the single stuck-at value fault model has been successfully used in case of different technology changes; from BJT to CMOS to Hybrid technology. It has been observed that, test sets generated based on the stuck-at fault model are quite acceptable for testing CMOS-based design even though this model does not incorporate all the defective behaviors occurring in fabrication process [10]. Therefore, in the aspect of QCA circuit testing, the use of this model can be a good option at its very early stage.

1.2. Objective and Scope of the Thesis

There is little work done on testing and detecting defects in QCA designs. Fijany and Toomarian proposed a fault tolerant implementation of a majority gate called a *block majority gate* [11]. A block majority gate shows fault tolerance to some of the defects, however, it does not address interconnection faults. Baradaran et al. proposed a testing scheme for QCA based designs [2, 8, 12]. A number of unique testing features have been identified through these studies. For example, the characterization of QCA defects and their effects on logic level are presented, as well as a testing scheme requiring constant number of test vectors has been demonstrated. However, their approach does not address the generality of application to any combinational circuit. The objective of this thesis is to find a design scheme that will overcome the limitations of Baradaran work. Additionally, the method to extend testing for modular structures is addressed.

In this thesis a design for testability of arbitrary combinational QCA logic circuit has been proposed. In particular, the method facilitates QCA functionality testing. Any combinational logic can be implemented using only AND-OR gates (with negated signals available), and in QCA this generally results in reduced test set for exhaustive fault detection within the data path. In Baradaran work, this strategy was used for QCA logic testing considering only primary inputs (either true or complemented, but not both) feeding different majority voters in AND-OR block, where it can be possible to set all the inputs having the same logic value for testing purpose. However, this approach fails for general circuits where both primary inputs and their complements may present. Logically it is impossible to set both original and its complement having the same logic simultaneously. So testing of these circuits will not be possible with proposed two test vectors. In this work a design scheme has been proposed that makes testing possible for any combinational QCA circuit. To achieve this, a block of test enable majority voters has been introduced which facilitate to set all inputs to AND-OR block to be of same value. In more complex circuits, sometimes the entire circuit is realized with different small blocks or modules to ease the testability. Such modular design may mask any fault on the cascaded line where the output of one stage goes as input to the next stage. To detect this fault an extra fault propagation path of majority voter has been proposed in this work.

We adopt a single stuck at fault model s-a-v for representing manufacturing defects in QCA, and propose a design scheme for testing modular and non-modular logic. Our results are demonstrated on basic logic circuits like an adder and a multiplier. Without going into the features of a particular QCA fabrication, errors on logic level are discussed, such that the approach achieves generality, and could be applied to any particular QCA implementation. To verify the scheme a simulation and layout tool named QCADesigner version 2.0.3 [13] is used. First the fault free circuit is designed and simulated. Then a number of faults have been injected in different locations of data path to find the faulty behavior. Simulated results show the validity of the approach.

1.3 Thesis Organization

The thesis report is organized chapter wise as outlined below:

Chapter 1: Introduction – in this chapter a prologue of the work done, the intensions, motivation, objectives and the scope is briefed. An introduction of a new Quantum Dot Cellular Automata (QCA) technology is presented. This technology is promising to surpass conventional CMOS technology in near future.

Chapter 2: Literature Review –this chapter presents the trend of research community toward QCA physics and designs. There has been considerable research on QCA as a new computing scheme in the nanoscale regimes. A theoretical finding for the root of Quantum Dot Cellular Automata and work done so far for achieving QCA circuits, their implementations, defects in circuit and probable testing scheme is highlighted.

Chapter 3: Basics of QCA – QCA is a relatively new concept in nanotechnology. In order to be acquainted with this technology this chapter presents the very basics of Quantum Dot Cellular Automata. The basic operation of QCA cell and the fundamental logic gates are demonstrated using basic majority voters. The clocking of the QCA circuits and different implementations of this technology are discussed to have a better idea on this novel area.

Chapter 4: QCADesigner: Simulation Tool – QCADesigner is a tool developed at the University of Calgary ATIPS laboratory to rapidly design and simulate new QCA circuits. This tool remains the only CAD tool versatile enough for designing practical QCA circuits and addressing functional issues regarding cell orientation. This chapter illustrates the various features of the tool to properly design and simulate QCA circuits which has been used to verify their correct functionality.

Chapter 5: QCA Physical Defects and Abstracted Faults – In literature it has been found that during the fabrication process different defects may occur in QCA cell causing the circuits to malfunction. This chapter describes such physical defects, their manifested errors and the behavior of them in logic level which have been simulated to reestablish the previous findings. It is necessary to be acquainted with these defects to come up with a proper fault model.

Chapter 6: Testing QCA Circuits - Proposed Scheme – In this chapter logic level testing aspects of QCA circuits are introduced. The unique feature of these circuits useful in easing the testing process, are presented. As single stuck-at fault model is used in this work, a brief overview of this model is also given here. The AND-OR configuration of

QCA circuits which facilitate testing is discussed. The limitations of the testing scheme proposed earlier employing this configuration is highlighted in this chapter. Finally the solutions to this problem are as well as modular design testing proposed in this research work is presented.

Chapter 7: Simulation Results – In this chapter, simulation findings are presented. The proposed design schemes are simulated to check proper functionality. Fault injection at different locations of data path has been exploited to detect faulty behavior. The simulation results validate the schemes presented in Chapter 6.

Chapter 8: Conclusion – This part present the conclusions of the thesis representing the findings in this study, and a directive for future.

Chapter 2: Literature Review

Up until now, the mainstream in fabrication technology has been CMOS. Whether in microwave, or in optoelectronics or in microelectronics, it has been in the forefront of fabrication technologies [14]. In the last few decades it has attained drastic improvements and tremendous miniaturization leading to submicron dimensions. However, since then, submicron physics is beginning to take over, and CMOS seems to loose its sole grip on fabrication monopoly. New physics resulted in new designs and hence emerged new paradigms of computing. Starting mostly in research oriented environment, nanotechnology is now beginning to find stronghold in its industrial footing.

As smaller and smaller gate lengths are achieved (65nm design and 45nm etch length in 2003 [14]), it is predicted that by 2010 gate length would reach 10nm. This then creates probably the toughest hurdle for CMOS. As researchers look for yet smaller devices, Quantum dot (QD) seems to be the limit in dimensional reduction in mesoscopic systems. Carrier confinement and quantum effects predominate at this level, as well as wave properties of particles becomes the key feature of electronics. Though molecular structures for electronic devices are also being investigated [15], still they do not pose to be the major driving force in near future.

It had been predicted that in densely fabricated quantum dots, the confined carrier wave function of individual dots would overlap, and thereby create miniband structures

[16]. This could be the basis for them to work as elements of cellular neural networks [17]. Fabricated by Fulton and Dolan [18], metal tunnel junction structures exhibited single electron properties. Such properties can be categorized as pseudo quantum dots, as they do not quantize carrier energy, but exhibits ultra small capacitance from the effect of charging one single electron.

The first computing scheme using quantum dot structures was proposed in 1993 [3]. Still emulating the Von Neumann type-computing machine, the quantum dots play a role of central elements, as they respond to the electronic charge condition of the adjoining dots. This proposed scheme is the Quantum Dot Cellular Automata (QCA). It was predicted that semiconductor Quantum Dots would be the forerunner in QCA implementation [19], but later it was found that charge fluctuations caused by impurities and defects tend to make the implementation burdensome. Hence metal tunnel type junctions are now the prime choice for implementation.

Since QCA is an emerging technology much work has been done on the physics of Quantum Dots, their implementations, switching and clocking requirements for proper operation. Different architectures have been proposed. The logical devices based on QCA have been demonstrated as well as design methodology has been introduced. A number of researchers have concentrated on the faults and testing of QCA circuits. Therefore the technical literatures available are discussed in this thesis under four different categories.

2.1 Physical and Device Level QCA

Quantum Dots have been demonstrated and fabricated in different materials, and have found their use in a diverse list of frontiers. When fabricated in an orderly cellular fashion, they have shown to be useful for computational device construction. Among such studies, only a few directly related with this work are briefed here.

The Quantum Dot Cellular Automata (QCA) was proposed in [3] as a viable technology. According to this study, QCA consists of an array of quantum device cells in a locally-interconnected architecture. The proposed cell is composed of coupled Quantum Dots occupied by two electrons. An idea of constructing logical gates and inverters is also presented.

A more detailed demonstration of the possible implementation of logic devices using coupled dot cells is found in [9]. It has been observed that, the charge distribution in each cell tends to align along one of two perpendicular axes, which allows the encoding of binary information using the state of the cell. A line of the cells can be used to design inverters, programmable logic gates, dedicated AND and OR gates, and non-interfering wire-crossings.

In [20] the authors describe a computing scheme with Quantum Dots and a possible such quantum dot cell is given in figure 3. An adiabatic switching paradigm is developed for clock-controlled pipelined QCA architecture. The binary information is stored as electronic charge as a current less computing is achieved.

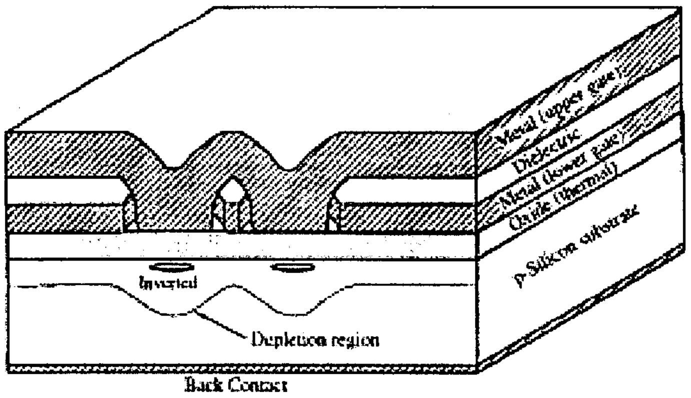

Figure 3: Possible implementation of a QCA cell [20].

Other investigators have been extending the theoretical analysis of QCA arrays. Tanamoto et al. [21] have proposed alternative ways of assembling QCA cells into useful devices. Lusth and Jackson [22] have applied graph theoretic analysis to QCA design. Chen and Porod [23] have developed sophisticated finite element models for gate depleted QDs in semiconductors that can relate dot occupancy to particular bias conditions. Fountain et al [24] have been matching the QCA approach with massively parallel processing schemes which require simple computational elements at each node.

Authors in [25] presented a simple clocked molecular QCA cell. The molecules show intrinsic bistability as a result of dipole charge configuration, which strongly couples to its neighboring molecules. Using local field authors achieved clocked control. The study is elementary, indicative only of the possibility of using molecular QCA. Then again, review of the possibility of implementing QCA in molecular scale or in nanomagnets has been done [14] [26]. Quite a number of molecular structures for QCA have been suggested, and one such 2-dot QCA is given in figure 4.It is predicted that the ultimate utility of QCA will come from combining currently existing technologies with advanced nanotechnology.

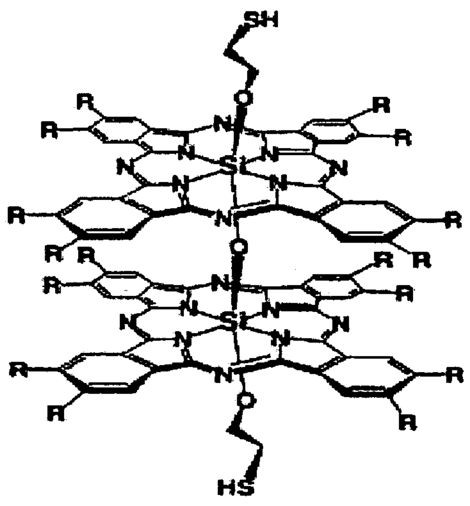

Figure 4: A possible implementation of a 2 dot cell [14].

Experimental implementation of QCA in nanoscale metal dots defined by tunnel barriers is reported in [27]. Here the authors demonstrate a controlled polarization of QCA cell switching agreeing with theoretical predictions. Clocked QCA operation is demonstrated on an example of a two cell shift register. In other study, [28] fanouts are presented. Additionally, authors proved the power gain as one driver cell induced logic in multiple fanned out cells.

Investigations related to switching speed and temperature dependence of QCA have been presented in [29]. Orthodox Coulomb blockade and master equation dynamics approach was taken into consideration for a semi-infinite shift register design. The crucial role of power gain as a function of temperature was shown. The behavior of such circuits as a function of clock speed and temperature is yet to be fully explained [30]. It is found that circuit speed is limited by RC time constant, so majority voters could work up to 450^0K, while semi infinite QCA wire could work up to room temperature.

The QCA physical design problem was addressed in context of VLSI physical design issues in [31]. It presents a comparison between ILP formulation and heuristic solution for problems like QCA partitioning, placement and routing of QCA circuits. It was found that heuristics approach gives the most optimized result. Unidirectionality and metastability problem within a QCA wire is studied, and 3D architecture is proposed in place of asymmetric spacing [32]. The classical calculation carried here showed that 3D configuration could also be the way to overcome metastability problems.

While H-memory is a design developed particularly for QCA, authors propose a new execution model that combines with H-memory for distributing the functionality of the CPU throughout the memory structure [33]. This resulted in efficient designs.

Today QCA is almost always constructed as four dot cells. But it is also reported in five dot and even six dot cell configurations. Instead of the four dot cell, a six dot cell is studied for clocking purpose in [34]. This work shows that more precise control is achieved by clocking the cells with the electric field that is generated by a layout of clocking wires.

2.2 Logic Level

Researchers have demonstrated a number of logic designs based on QCA. These designs include combinational circuits like simple XOR gate, 2-to-4 decoder, circuits realizing various Boolean functions, arithmetic structures like different adders (ripple carry adder, carry-look-ahead adder, carry save adder etc.), multipliers. Not only that, some sequential circuits such as flip-flops, shift registers, barrel shifters, memory unit, microprocessors have also been presented in different literatures. The trends of these studies are addressed in this section.

The arithmetic structures based on QCA is discussed in [35]. As the cost function of QCA realization is dependant on the clocking zones, the total delay only depends upon the number of clocking zones, irrespective of the number of gates in a circuit. Hence a different design approach becomes a necessity for QCA arithmetic design concept. A QCA design methodology is developed in this work, and adder and multiplier circuits are explored, as the examples. Study suggested the serial arithmetic to be preferable to its parallel counterpart for increased interconnect latency and potential implementation technologies for realizing QCA based circuits.

The performance analysis of different types of adders in QCA has been carried out in [36]. It is demonstrated that a QCA-based ripple carry adder and a bit-serial adder outperform carry look ahead and carry select adders. The ripple carry adder is found to be the fastest one, while the bit-select adder is found to be the most area efficient. However, the results are only based upon two input majority voter as primary gates, so different input gates could lead to a reevaluation of performance analysis. The schematic design of the bit serial adder used for this study is given in figure 5.

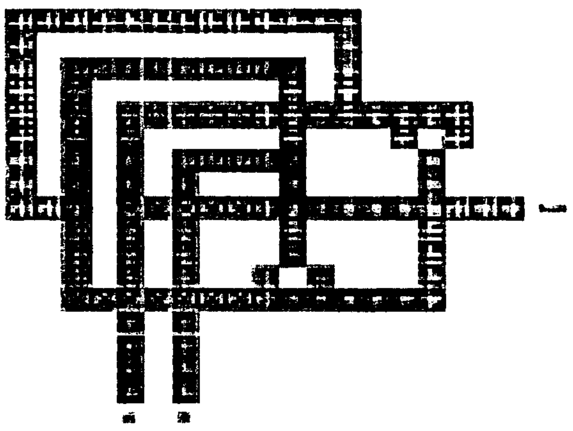

Figure 5: Bit serial adder layout [36].

In another study [37] the authors presented an improved design for QCA adder using only three majority voters and two inverters. They proposed to connect *n* 1-bit adders to obtain *n*-bit carry look ahead adder with reduce hardware using the same clocking scheme and parallel structure of the previously reported designs. In another study [38], the authors propose a pipelined carry look ahead adder (CLA) in QCA. The designs are compared to a ripple carry adder, and a modular layout is presented for CLA. The proposed designs show complexity and area which are roughly proportional to operand sizes. Also the designs give delays that increase at a slightly greater rate than the logarithm of the operand sizes.

With the motivation of finding a systematic way of translating three input Boolean function to majority functions, the authors in [39] have presented an analysis for reduction in gate count in QCA design structures. The study is based on the use of Karnaugh maps to optimize a majority function.

Authors in [40] have presented a method for mapping three variable Sum-of-Product (SOP) expressions to QCA majority logic. They also illustrated a method for reducing the total number of majority voters in a QCA design. The study identified thirteen functions to represent all three input functions, and proposed a gate reduction scheme for QCA circuits. The leading example here is a 1-bit full adder.

A multiple bit serial stream analyzer is presented in [41]. This is functionally equivalent to a shift register and a multiple bit comparator. The quasi-adiabatically switching is quantum mechanically analyzed, and the research suggests that QCA could easily work as a sequential circuit with this type of switching.

To facilitate the designing of QCA circuits a complex gate using majority voters is proposed in [42]. This complex gate comprises of 7 inputs, and can be configured to work as 4 input AND or OR gate. These gates can be later on used to represent Boolean functions in a popular Sum-of-Products (SOP) or Product-of-Sums (POS) format. Generally three of the inputs act as control inputs, while the remaining four act as data inputs to the gate. An example of such complex gate is given in figure 6 below.

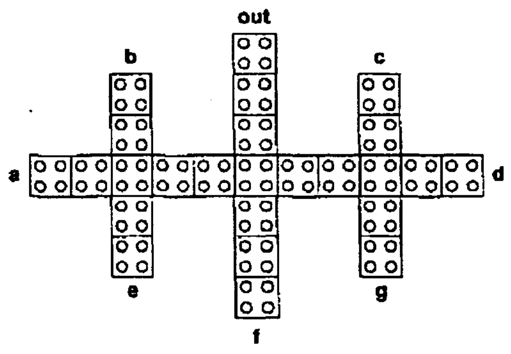

Figure 6: 7-input complex QCA gate proposed in [42].

A study of high-level evaluation of QCA circuit is also carried out [43] with a 4-bit ALU and a 4x4 memory unit. Among few requirements suggested for that type of designs, the most important one is probably the need for multi-layer QCA support as a replacement for the very unreliable coplanar crossover, and high resolution clocking for each majority voter, independent of its inputs. A parallel memory architecture is proposed in another literature [44]. Its basic operation requires three clocking zones, where the timing of the zones needs two additional clocks to implement a four step reading/writing process. This method of memory design is said to reduce the requirement of overall clocking zones and the required clocking circuitry.

The traditional wire crossings in QCA circuit sometimes cause problem in circuit functionality. To overcome this problem a wire crossing network is presented in [45]. This, so called, crossbar network can be used in PLA structures, and is implemented using parallel to serial converters. Special latching signals determine, at a particular time, signal connections made within the crossbar, so no physical change is required. Hence the same physical design can dynamically be used to realize different logic functions. The network uses parallel-to-serial converters, shift registers, and time-dependent latching devices, so that such elements eliminates the need for rotated cells on an interstitial cell spacing grid of traditional QCA wire crossings.

Researchers have also tried to focus on quantum computing using QCA technology. A quantum computing architecture based on 1D array of QCA cells is proposed in [4]. Basic quantum gates such as NOT, C-NOT, and a 3-bit gate, the modified C-NOT are proposed. Coherence vector formalism was used to describe the system.

2.3 Design Methodology and Tools

Predicting the end of the CMOS roadmap, studies have been conducted in [46] to propose thorough design methodology to address computationally viable circuits, which will be buildable with QCA. Today most of the QCA designs are done on a cellular basis, which lacks scalability, and ultimately will fail for larger circuit size. To overcome this obstacle, a hierarchical design methodology, similar to that of CMOS, has been proposed in [47] for QCA. It has been suggested that creating and verifying the circuit at higher level of abstraction would reduce design time. Moreover detection of any potential faults at behavioral and structural levels is faster; hence an efficient design can be obtained before a more comprehensive quantum mechanical simulation of adiabatic QCA switching is executed. This process allows optimization related to design efficiency and error detection. Hence error detection in function level becomes prudent. The design process for CMOS and QCA presented in this work is given below in figure 7.

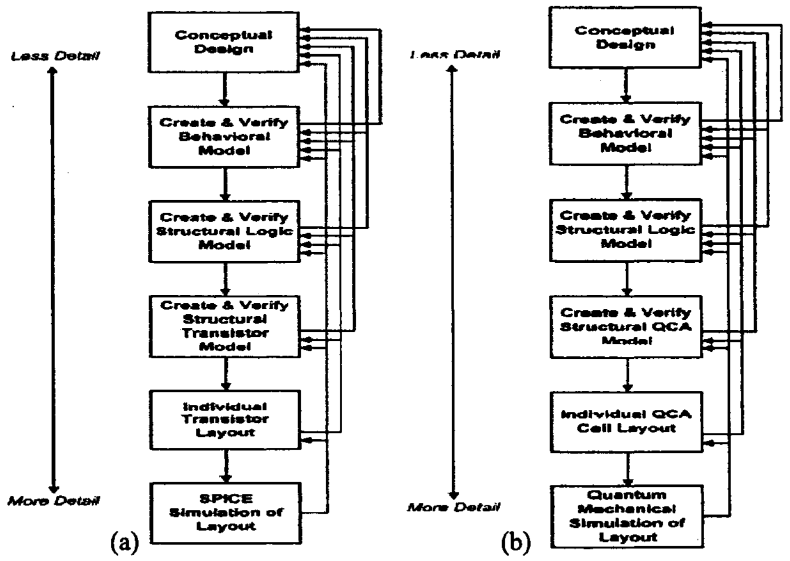
Less Detail
More Detail
Conceptual Design
Create & Verify Behavioral Model
Create & Verify Structural Logic Model
Create & Verify Structural Transistor Model
Individual Transistor Layout
SPICE Simulation of Layout
(a)
Less Detail
More Detail
Conceptual Design
Create & Verify Behavioral Model
Create & Verify Structural Logic Model
Create & Verify Structural QCA Model
Individual QCA Cell Layout
Quantum Mechanical Simulation of Layout
(b)

Figure 7: Flow chart of (a) CMOS design process, and (b) CMOS based QCA design process [47].

A design methodology for QCA based on conventional CMOS technology is proposed in [48]. This work utilizes HSPICE to produce a methodology for circuit design flow just like that done in CMOS, with the view to increase design compatibility and a feasible way to design large scale QCA nanodevice.

Also formalism is presented in [49] aimed to modify design of QCA circuits. Here a set of standard circuit elements of uniform layout rules is used. It simplifies the circuit design, and overcomes the input delay sensitivity of QCA logic, as well as increases considerably device density over conventional CMOS design.

A modular design technique based on basic building blocks referred to as *tiles* is presented in [50]. The logic capability of individual tiles is analyzed and the tiles are then used to form the final logic. A 3x3 QCA grid is chosen as the basic block, and five types of tiles are constructed. It is found that the tiles are area efficient, and offers versatile logic operations.

As QCA is an emerging technology, much work has been done with respect to design methodology, with many experimental designs of the same, rather basic, classical arithmetic logic proposed. However, due to the lack of the basic knowledge regarding sneak noise path, many of the primitive designs are prone to functional error and malfunction in real circuits. To overcome this obstacle, several critical vulnerabilities in the QCA gates and interconnects have been addressed, and a disciplinary guideline has been proposed in [51] to prevent plausible erroneous QCA designs. The guideline presents a set of rule which effectively suppress noise in QCA.

In [52] [53], the authors have presented a CAD tool specially build for working with QCA. The tool, QCADesigner has been successfully used to simulate full adders, barrel shifters, random access memories, etc. As QCA logic requires a rapid and accurate simulation and design layout tool, a full functionality check is provided. Several simulation engines are incorporated to facilitate accurate simulation. In fact, QCADesigner is emerging as the main simulation tool for QCA design and layout formation. In the following figure 8, a design layout done with the tool is given with different clock zones, as it is clearly shown in different colors.

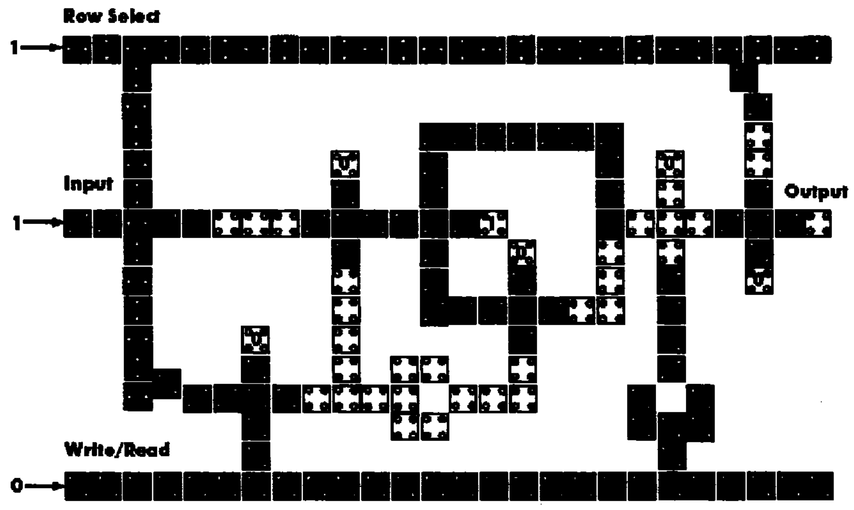

Figure 8: Snapshot of a QCA random access memory cell design with QCADesigner [52]

2.4 Defects and Testing

Recent developments in QCA fabrication process have indicated that defects are likely to occur in QCA based circuits. Therefore the testing community has extended their investigations toward the possible defects abstraction and testing procedures, as well as proposing some ways to achieve fault-tolerant circuits. The major propositions in this regard are highlighted in this section as a prelude.

QCA latch and shift registers have been studied for possible errors in [54]. An analysis of the operational error types is presented, and their properties are investigated. The results showed that smaller device size and higher operating frequency drastically reduced the number of errors.

Research has shown that fabrication defects arising from self-assembly of molecules in QCA are highly likely. Hence fault tolerant architectures become necessary. Possible defects occurrence during fabrication has been examined in [55], and their manifestation in logic level has been studied. Additionally, a strategy for understanding how groups of defects affect the system has been proposed. Finally, a prototype tool for fault modeling has been demonstrated.

The probable defect occurring in a molecular implementation of QCA is analyzed in [56]. Unlike that in metal based QCA, defects here could happen due to erroneous deposition of cells on substrate, which might pose different functional error in different QCA logic elements. One interesting observation was that a single missing cell in a wire may cause local non polarization of a particular cell, but due to the strong coupling between non neighboring cells, this fault does not propagate. Instead a weak signal from the previous cell configures the subsequent cells after the fault to have the correct logic. Hence unless there occurs a sequence of missing cells, single cell omission in wire does not pose a functional error, provided that the fault does not occur in a primary output.

The defect tolerance of molecular QCA has also been studied [57]. A QCA SR type flip-flop is proposed accounting to the strict timing issues. Then single additional and missing cell faults are introduced. It has been found that the majority of these defects manifest functional error abstracted mostly by stuck-at value (s-a-v) faults and, in some cases, with unwanted inversion. Inverters are shown to be prone to an addition fault, while majority voters are susceptible to missing cell defect.

Wire delay dominates logic delay in QCA. Additionally, wires are more prone to faults than gates. Therefore, triple modular redundancy (TMR) based fault tolerance becomes inappropriate for QCA, [58]. In [58], researchers have proposed TMR with shifted operands (TMRSO), and demonstrated it on a full adder to obtain better area utilization and throughput performance. They then suggested implementing TMRSO instead of TMR for fault tolerance aspect. A 2-bit TMRSO adder proposed in the study is given in figure 9 below.

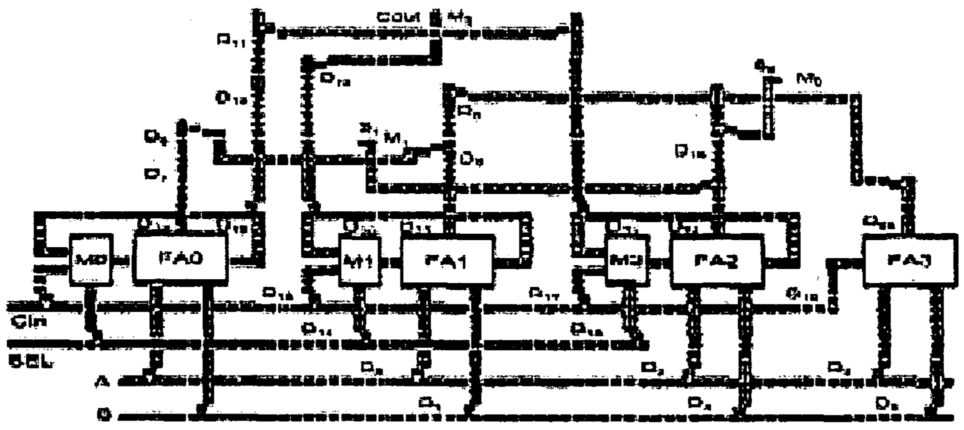

Figure 9: A 2-bit Adder in QCA implemented as TMRSO [58].

The scalability issue, and the defect tolerance henceforth have been studied in [7]. As scaling in QCA is associated with cell dimension and cell to cell spacing, displacement and misalignment defects are analyzed to study the sensitivity to manufacturing processing variation changing with device scaling. The defects are shown to have definitive behavior, and it is found that the relation between the cell size and smallest distance for an erroneous behavior remains mainly linear. In another study [59], the same issue has been addressed using the QCADesigner tool. It has been found that for displacement and most of the misalignment defects, the relation between the cell size and defect tolerance has a parabolic nature, giving an optimized range of the device size which is the most defect resistant. It is believed that a strong Columbic interaction gives better defect tolerance for smaller devices resulting in a parabolic dependence of defect to a cell size.

Defect characterization of QCA manufacturing process has been studied in [2]. A detailed analysis shows that though faults manifesting themselves in functional level can mostly be abstracted by s-a-v fault model, some faults result in unwanted complementation of the desired function. The success of s-a-v fault model then depends on specific implementation of a particular logic. As in the case of CMOS circuits, different implementations of the same logic generally result in different fault coverage. Some very unique and interesting testing properties are found for QCA. In another study [8], it has suggested that specific implementation of a particular logic must be adopted to achieve high fault coverage with s-a-v fault model. Constant testability of QCA designs has been investigated. MV-based designs offer C-testability in both 1-D and 2-D structures.

Testing of QCA circuits have also been studied. In [12], the authors present a design for testability scheme, in which circuits are implemented with inverting blocks followed by AND-OR logic. The method requires only two test vectors for complete fault coverage. Unfortunately, the scheme is illustrated on a rather elementary non inverting circuit, which fails drastically for real life more complex inverting logic.

Conventional QCA circuits with majority voters realizing AND-OR functions use inverter. However, these circuits are not area efficient. To minimize area requirement a novel basic QCA gate, AOI (And-Or-Inverter) has been proposed in [60]. This gate is said to be a universal gate with greater area minimization and delay reduction in larger logic circuits. Defects have been studied for this building block too, and a test set is provided for the gate. The study of the defects in functional level suggests that the gate is robust. The proposed AOI gate is illustrated below in figure 10.

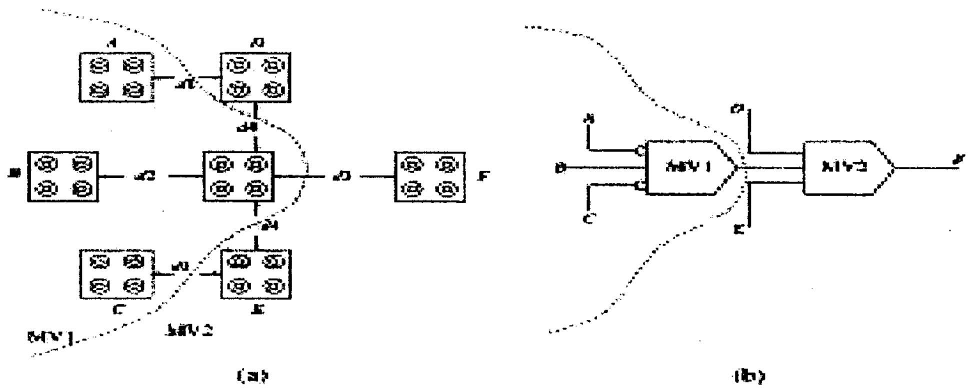

Figure 10: The AOI gate [60], (a) The QCA implementation, (b) Schematic diagram.

The defect tolerance and testing schemes mentioned in this section can be employed to have a better and effective design in QCA technology. The rest chapters of this thesis starting with very basics of QCA technology will be presented to accomplish that goal.

Chapter 3: Basics of QCA

Quantum Dot Cellular Automata (QCA) is a new way of implementing logic and information processing. It is conceptually different from conventional technologies such as CMOS, and so it is absolute essential to have some background study. Fabrication of QCA can be obtained in different technologies, like: metal tunnel junction, semiconductor, self-assembled dots, etc., however the basics would somewhat differ in various implementations. Moreover, as QCA belongs to the new, and still not well analyzed, class of quantum circuits, the understanding of its operation would require rather proficient knowledge of quantum effects such as tunneling, electron charge interaction and of quantum devices such as single electron transistor (SET). The detail theoretical approach of the device and circuitry is presented in this chapter, offering a comprehensive look into this new emerging technology.

This chapter will first present the idea and working of the basic cell, which is the fundamental building block of the system. Then it will focus on the NOT gate implementation and interconnections. Subsequently, this chapter impart the basic gate, a majority voter (MV), which through appropriate set of control inputs reduce to AND or OR gates. An important feature of QCA, the clocking will also be discussed. Some implementation techniques of QCA will be briefed. To accrue the basics of QCA an arbitrary logic and some complex logics implemented by QCA will also be presented.

3.1 The Basic Cell

A high level embodiment of the basic four dot QCA cell is given in the figure 11. It consists of four possible positions for electrons, named Quantum Dots, placed in a square configuration. Exactly two extra mobile electrons are loaded into each cell, which can quantum-mechanically tunnel between dots, but not through cells. The charge is fixed and immobile [32]. Tunneling out of the cell is assumed to be completely suppressed by the potential barriers between cells. It is also possible to add a fifth dot at the center of the square. Although such an addition improves the behavior of the cell slightly, for simplicity reasons the attention will mainly be given on the four-dot cell. If the barriers between cells are sufficiently high, the electrons will be well localized on individual dots.

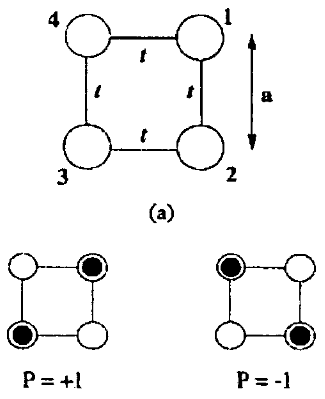

Figure 11: Schematic of the basic four-site cell. (a) The geometry of the cell. The tunneling energy between two neighboring sites is designated by t, while 'a' is the near-neighbor distance. (b) Coulombic repulsion causes the electrons to occupy antipodal sites within the cell. These two states result in cell polarizations of P = +1 and P = -1 [20]

As electrons will repel each other electrostatically, they will be forced to reside in the diagonally opposite dots of the square, and by that obtaining the minimal energy state. For an isolated cell, two energetically minimal equivalent arrangements exist as shown in figure 11(b), which could be denoted as a *cell polarization*. The term "*cell polarization*" refers only to such arrangement of charge, and does not imply a dipole moment for the cell. If the charge on a dot i is value ρ_i, then the polarization is defined [61] as:

$$P = \frac{(\rho_1 + \rho_3) - (\rho_2 + \rho_4)}{\rho_1 + \rho_2 + \rho_3 + \rho_4}. \tag{1}$$

If the two extra electrons are completely localized in dots 1 and 3, the polarization is + 1 (binary 1); if they are localized in dots 2 and 4, the polarization is - 1 (binary 0), figure 11. The tunneling between dots implies that ρ_i may not assume an integer or polarization values.

The two polarization states of the cell will not be energetically equivalent if other cells are nearby. Consider two cells close to one another as shown in the inset of figure 12. The figure inset illustrates the case when cell 2 has a polarization of 1. It is clear that in that case the ground state configuration of cell 1 is also a 1 polarization. Similarly if cell 2 is in the state -1, the ground state of cell 1 will match it. The figure shows the nonlinear response of the cell-cell interaction.

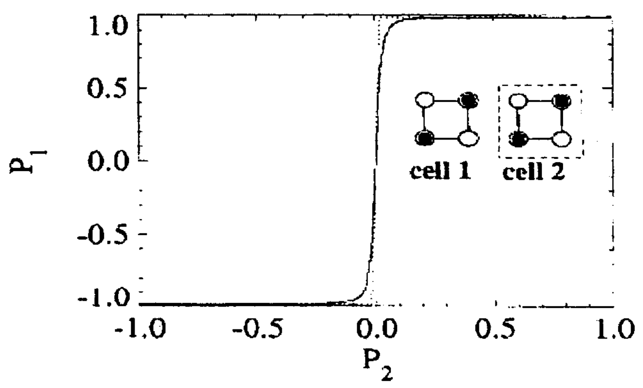

Figure 12: The cell–cell response. The polarization of cell 2 is fixed and its Coulombic effect on the polarization of cell 1 is measured. The nonlinearity and bistable saturation of this response serves the same role as gain in a conventional digital circuit [55]

3.2 Kink Energy Associated with Coulomb Interaction

The Coulomb interaction between cells allows us to transmit information along interacting arrays of QCA cells. This interaction can be described by the *kink energy,* E_{kink} associated with the energetic cost of two cells having opposite polarization [20]. The electrostatic interaction between charges in two four-dot cells *m*, and *n* is:

$$E^{m,n} = \frac{1}{4\pi\varepsilon_0\varepsilon_r}\sum_{i=1}^{4}\sum_{j=1}^{4}\frac{q_i^m q_j^n}{\left|r_i^m - r_j^n\right|}, \tag{2}$$

where ε_r is the dielectric constant, q_i^m is the charge in dot *i* of cell *m*, and r_i^m is the position of dot *i* of cell *m*. Since the kink energy represents the energy cost of the two cells having opposite polarization, the energy of the two cells having the same polarization must be subtracted from that of them having opposite polarization:

$$E_{kink}^{m,n} = E_{opposite\ polarisation}^{m,n} - E_{same\ polarisation}^{m,n}. \tag{3}$$

Although not shown in figure 12, there are background charges of *+e/2* in each dot, to ensure that the cell actually bears a neutral charge. These positive charges must be included in the calculations of the kink energy. In general, the interaction between cells can be described by a quadrupole-quadrupole interaction, in which the kink energy decays as the fifth power of the separation between cells as shown in figure 13.

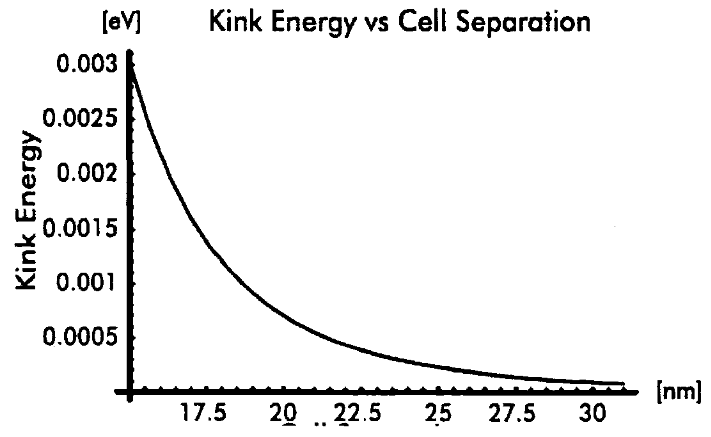

Figure 13: Kink energy between two cells as the separation between the cells is increased from 15nm to 30nm. We see that this kink energy decays as the fifth power of the separation. This is a general property of quadrupole-quadrupole interaction [59]

3.3 The Inverter

The inverter gate is illustrated in figure 14 as first proposed in [9]. It should be noted that unlike its CMOS counterpart, achieving a NOT operation is rather troublesome in QCA based on the majority voter. It requires more cells and implementation area. Cells which are positioned diagonally from each other tend to anti-align. This feature is employed to construct an inverter. The anti-alignment can also viewed as a simple consequence of the mutual repulsion between electrons and the geometry of the cells.

Although two diagonal cells function as an inverter, this more symmetric design ensures exact symmetry between the inversion of a one and a zero.

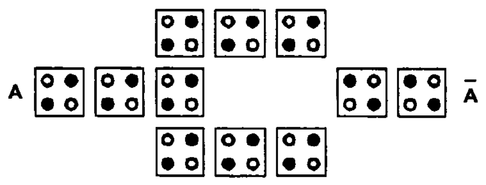

Figure 14: QCA inverter

As seen in figure 14, the signal coming from left to right splits into two (top and bottom) arrays, and is inverted at the point of convergence. The inverter is geometrically symmetric, so inversion of logic occurs with increased reliability.

3.4 QCA Wire

In contrast to conventional logic where information is transferred between devices by electrical current, QCA achieves this by Coulomb interaction, which passes the state of one cell to its neighbors. This results in a technology, in which information transfer (interconnection) is the same as information transformation (logic manipulation) with low power dissipation [61]. QCA achieves the data flow in two ways: *binary wires* and *inverter chains.*

3.4.1 Binary Wire in QCA

A QCA wire is shown in figure 15. The left-most cell's dot polarization is in accordance to input configuration. The ground state configuration of the remaining free cells is then the one with each cell polarized in the same way as the input cell. This can be considered as the transmission of the input signal from one end to the other.

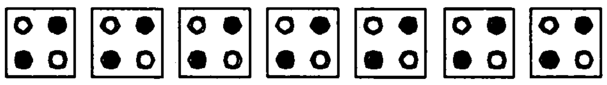

Figure 15: Binary wire

3.4.2 QCA Inverter Chain

The other wire implementation is the inverter chain or the *45^0 chain* as it is sometimes called, figure 16. Here the cells are themselves rotated by 45^0, and placed side by side. Therefore, during the propagation of the information through the chain, the original logic and its complement in adjacent cells are available. Both of these values can be obtained from the same wire with ripper cells placed at the suitable places as discussed in next subsection.

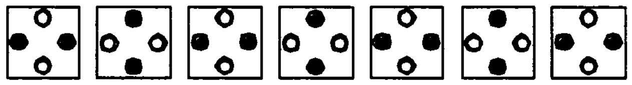

Figure 16: QCA Inverter Chain

3.4.3 Rippers Cells in QCA

The alternating polarization in the rotated wires also permits easy extraction of both, the original signal and its complement. A binary value of a signal on a 45-degree wire will alternate between logic 1 and 0. By placing a 90-degree cell between two 45-degree cells, both the original signal value and its complement can be obtained without an explicit inverter circuit. An example is given in figure 17, where two ripper cells placed in different positions extract original an inverted signal from the same wire.

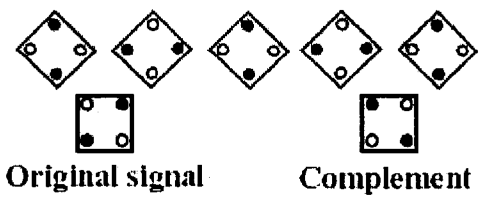

Figure 17: Ripper cells placed to extract data from inverter chain

3.4.4 Fan Out

Fan out in the case of QCA is shown in figure 18. It allows the result of a calculation to be propagated to two or more other points within the array. At this point, perhaps it is better to emphasize that QCA is a nanotechnology, which employs quantum nature of electrons to implement Boolean logic, not Quantum Logic. Therefore, a fan out of QCA is qualitatively similar to a CMOS logic fan out, and does not represent a fan out of quantum logic, as quantum logic principle does not allow fan outs under any circumstances.

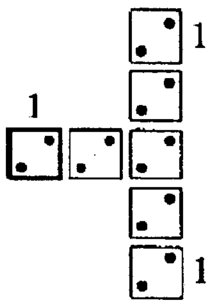

Figure 18: QCA fan out

3.4.5 Wire Crossings

A useful feature of QCA design is the ability to cross wires in the plane. In conventional technology, information is coded in voltages or currents in conductors, and wire crossings require a bridge out of the plane. In QCA many wires carrying different binary values can cross each other at the same level without interference or crosstalk, while still continuing to give correct result in this implementation. The only limiting factor is that one of the wires would have to be binary wire while the other an inverter chain. Again, this feature is a consequence of the Coulomb interaction and the symmetries present in the cell charge.

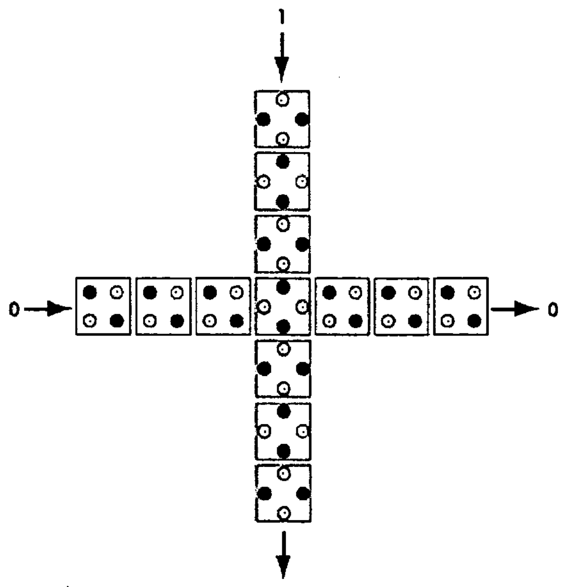

Figure 19: Wire cross over

In figure 19 a crossover of a horizontal binary wire with a vertical inverter chain is illustrated. It is to be noted that the inverter chain is continuous, while the binary wire has a break at the crossing (as seen in figure 19, the point of crossing is on a rotated cell of the inverter chain, and if only the binary wire were considered, there would have been a gap in the horizontal propagation path). If the effect of the vertical wire consisting of the rotated cells could be ignored (this is only logical as symmetry of the inverter chain above and below the horizontal line would cancel out their effect to result in a net null effect), then the information in the horizontal wire will propagate across the gap (as far as binary wire is concerned, as the crossing point cell is an inverter chain cell, a binary wire cell is missing at that point. However, no detailed analysis has been carried out regarding physical aspects of such a crossover, and it is believed that such a system would be sensitive to fabrication variations that break the symmetry. In particular, variations in the position of cells or dots in cells of this building block introduce crosstalk between the two interconnects, which quickly dominates its operation.

A further problem arise when the horizontal wire is sectioned because of the three vertical interconnects as shown in figure 20. Each section is weakly coupled to the other sections of the horizontal wire. In result, each section in between the vertical interconnects is essentially floating, and hence, very sensitive to noise. Additionally, the last two sections of horizontal wire take on the polarization of the fixed cell located directly above it rather than the polarization at the input to the horizontal wire.

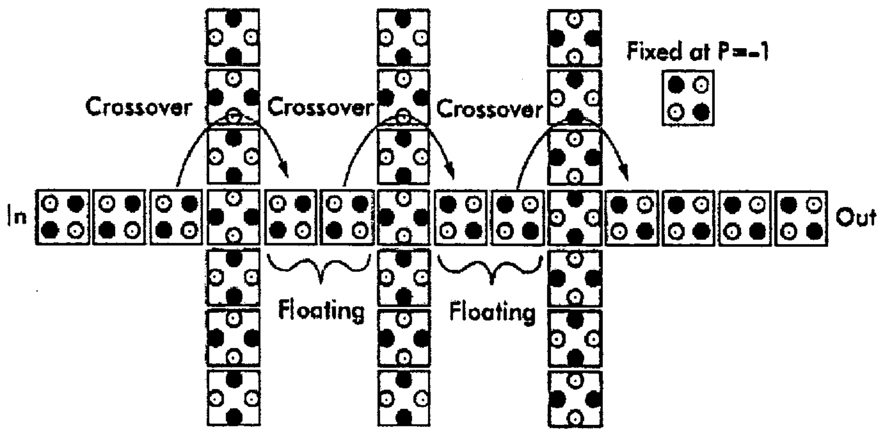

Figure 20: A section of wire with three coplanar crossovers.

This small circuit demonstrates the complexity of coplanar crossovers, and their very high probability to failure.

3.4.6 Multi-layer QCA

An alternative method of crossing signals is by using a *multi-layer* QCA, shown in figure 21. In this approach signals can effectively cross over another layer. To do this, a vertical interconnect is required. By stacking cells one on top of another the signal can transmit to a desired layer, where it is again transmitted horizontally.

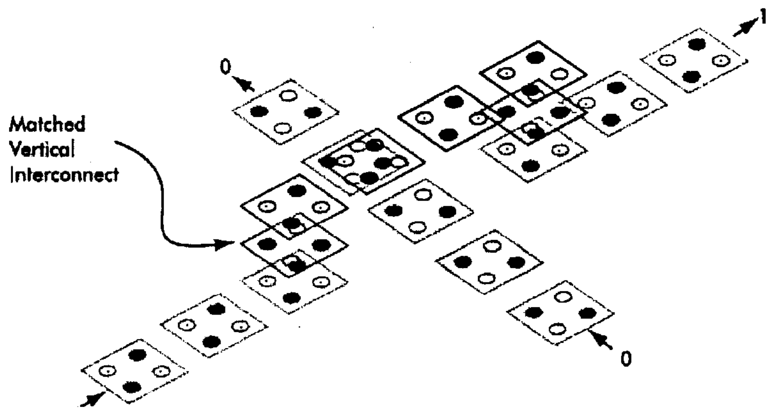

Figure 21: Multi-layer crossover. The vertical separation between cells can be turned to match design parameters.

Unlike CMOS integrated circuits, where metal layers are used to implement connections without, de facto, performing of any intelligent functions, the extra layers of QCA can be used as active components. In this way, it is believed that multi-layer QCA circuits can potentially require much less area than planar ones.

3.5 The Basic Gate: Majority Voter

The fundamental QCA logical device is a three-input *majority gate*, from which more complex circuits can be built. The basic majority voter is obtained by placing four neighboring cells adjoining to a device cell, which is in the middle. Three of the side cells are used as inputs, while the remaining one is the output. The 5-cell configuration is shown in figure 22. While all the three input cells (having their distinct logic values of either 1 or 0) tries to drive the device cell to its own accord, the system reaches stability as the device cell in the center is driven to its lowest energy state, i.e., the state of the majority of the inputs. This is then also the resulting output, which can feed other circuit wires. The difference between input and output cells in this device, and in QCA arrays in general, is simply that inputs can be set to desired (and hence, arbitrary) values and outputs then change in that accord, as clocking only allows data to flow from input to output and not vice versa. Although, the unidirectional signal follow is standard for CMOS devices, all quantum circuits are generally reversible, and hence support bidirectional signal flow. In the case of the majority voter clocking determines which cell of the four side cells would be the output and which cells would be the inputs. The inputs to a particular device can come from previous calculations or be directly fed in from array edges. In this process, the majority voter realizes the function OUT=AB+BC+CA.

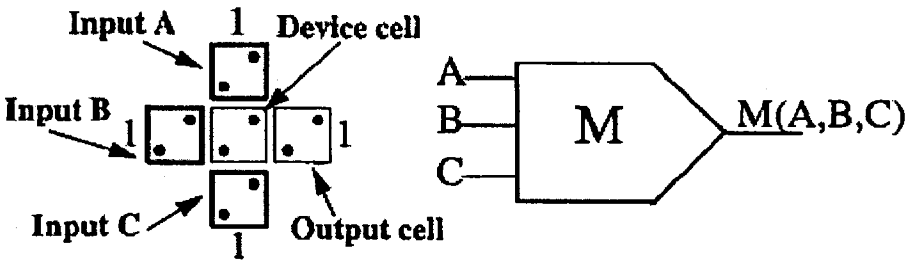

Figure 22: The 5 cell basic majority gate implementing the logic function $OUT = AB + BC + CA$ and its logic symbol.

As one would easily verify, setting one input of the majority voter permanently to 1 would transform the system into an OR gate, while setting it to logic 0 would accomplish a logic AND operation. In the figure 23, the arrangements for the OR and AND operation is illustrated.

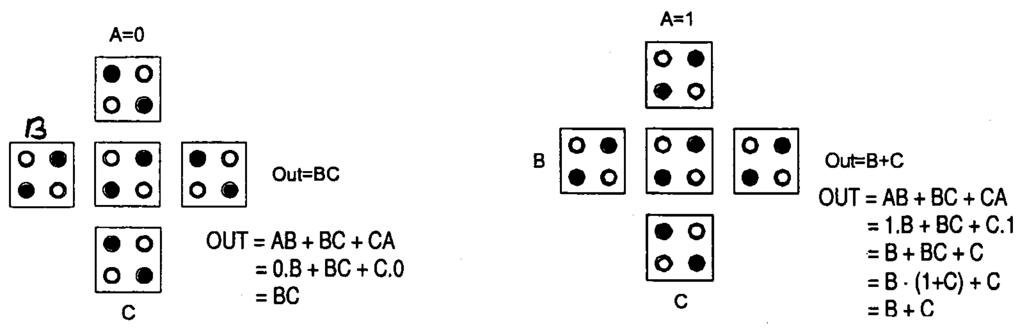

Figure 23: QCA majority voter configured as AND and OR gate

3.6 QCA Clocking

Unlike the transistor-based gates, QCA basic cell has no inherent directionality for information flow, and a circuit made of unclocked cells would propagate information in uncontrollable directions. It has been shown that signals traveling along arrays of QCA cells can be reflected off imperfections and interfaces analogous to reflections from impedance mismatches in RF circuits [37]. The only effective way to eliminate this problem is to somehow make sure that data in QCA array always flows in one direction, from input to output. This is achieved by a carefully selected sequence of clocking signals, which ensures unidirectional data flow. In every implementation of QCA (metal junction, semiconductor, etc.), clocking has seen to overcome this problem. Another important difference from CMOS is that QCA cells themselves do not exhibit any power gain. The clock is therefore required to provide the cells with the power gain to transmit signals along lengthy QCA wires.

Two types of switching are possible in the operation of QCA: *abrupt switching* and *adiabatic switching* [57]. In abrupt switching, the inputs to the QCA circuit change suddenly bringing the system to some excited state. Energy in Quantum Dots is quantized, and any energy except the lowest possible one is regarded as excited state of an electron. During this kind of switching, the electron gets into arbitrary excited state with unstably high energy. Subsequently, the QCA circuit is relaxed to ground state by dissipating energy to the environment [20]. The relaxation where energy is not conserved is called *inelastic relaxation.* The inelastic relaxation is uncontrolled, and the QCA circuit may enter a metastable state, which is determined by a local rather than a global energy ground state. Therefore adiabatic switching is usually preferred.

In adiabatic switching, the system is always kept in its instantaneous ground state. A clock signal is introduced to ensure adiabatic switching for QCA. Such a clock signal is generated through an electric field, which is applied to the cells to either raise or lower the tunneling barrier between dots within a QCA cell. When the barrier is low, the cells are in a non-polarized state; when the barrier is high, the cells are not allowed to change state. Adiabatic switching takes four phases: first is lowering the barrier, subsequently followed by removing the previous input, applying the current input, and finally raising the barriers [20], figure 24.

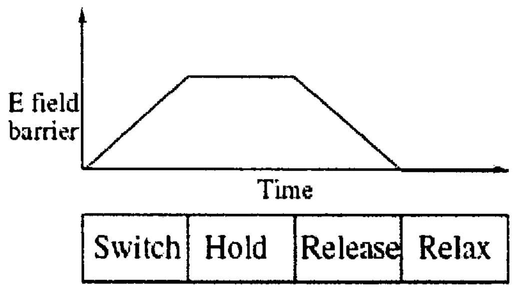

Figure 24: Adiabatic clocking in QCA

The clocking scheme, which was introduced in [61], consists of four phases: *switch*, *hold*, *release* and *relax*, each separated by 90 degrees. The QCA circuit is partitioned into so-called *clocking zones*, such that all cells in a zone are controlled by the same clock signal. Cells in each zone perform a specific calculation. After that the state of a zone is fixed so that input signals can be provided to the next zone [62]. At the beginning of the switch phase, the cells in a zone are in an *unpolarized state*. During the switch phase, the barrier is raised, and the cells in a zone begin to *polarize* according to the input value provided from the output of the previous zone. At the same time, the next zone is in an *unpolarized* state, thus not affecting its computational state. Then the cells are placed in the *hold phase*, in which the barrier is sufficiently high such that the state is unchanged, and can be used as input to the next zone. After lowering the barrier, the cells are placed in a *release phase*, in which they begin to relax to an *unpolarized state*. The last phase is the *relax phase*, in which the cells are again *unpolarized*. It can be said that the information transfer is done in a pipelined fashion. The clock signals (through the included electric field) can be generated by embedded CMOS wires below the QCA plane [57]. The four phases of clocked QCA majority voter and binary wire is shown is figure 25.

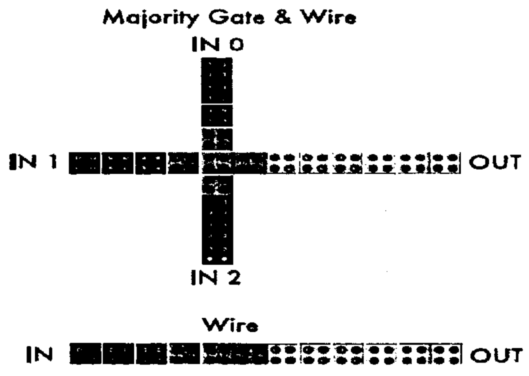

Figure 25: A clocked QCA majority voter and binary wire

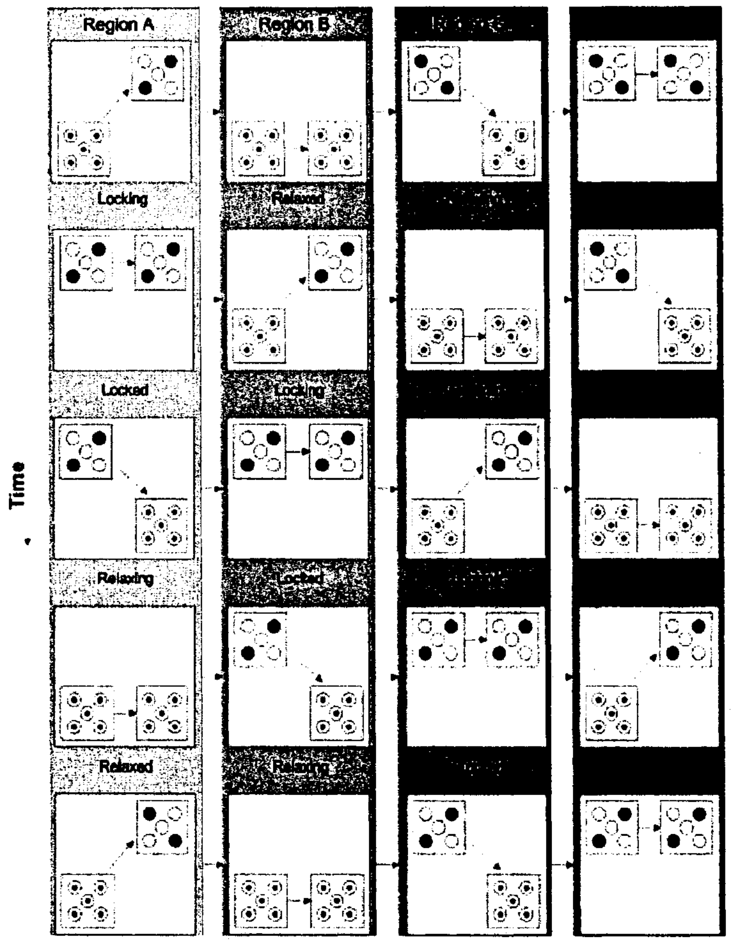

Figure 26: Illustration showing an example of adiabatic switching [29]

A detailed explanation of clocking is illustrated in figure 26. Each row represents a different time step of a four-cell device, partitioned into four distinct regions, labeled *A* through *D*. Each box shows: the current state of the cell, and the previous state of the cell, with an arrow indicating whether intracellular barriers are being lowered or raised. Based on the height of the intracellular barriers, the cell in each region cycles through four stages: *locking* or *switching* (barriers raised), *locked* or *hold* (barriers held high), *relaxing* (barriers lowered), and *relaxed* (barriers held low). Relaxed and relaxing cells are depolarized, so they will not drive neighboring cells. Locked cells are used to drive neighboring, locking cells. In this way, the flow of data can be controlled. Next, the barriers are lowered for region *A*, while they are stabilized for region *B*, and raised for region *C*. The cell in region *A* is relaxing, and will no longer drive the other cells. Region *B* is locked and drives the cell in region *C* to a value of '1', which is now locking. This process continues in the fourth time step. In the final time step shown, a new input can be applied to the region *A* cells, starting a new cycle.

3.7 Logic Implementation in QCA

The logic symbol of the basic majority voter and an arbitrary logic implementation using AND-OR logic is given in figure 27.

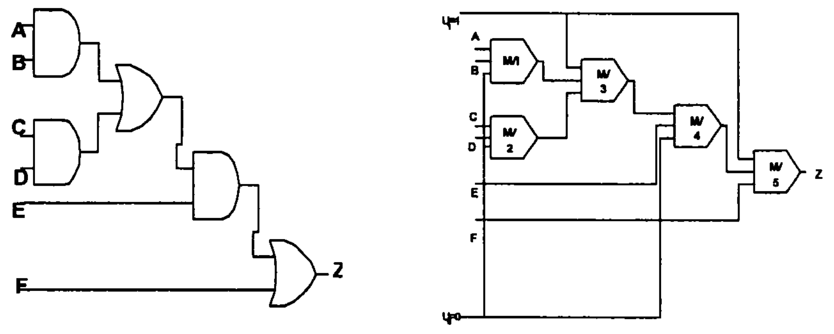

Figure 27: Logic symbols of QCA. (a) logic symbol of an arbitrary logic implemented in traditional logic circuitry and (b) the same logic implemented in QCA.

A more complex gate representation can be obtained [42]. Figure 28 shows a 7-input gate. This gate is composed of three 3-input majority gates. Three inputs to this gate: *e*, *f*, and *g* are control signals used to specify the functionality of the circuit. The remaining four inputs: *a*, *b*, *c*, and *d*, are used to implement Boolean functions of four variables.

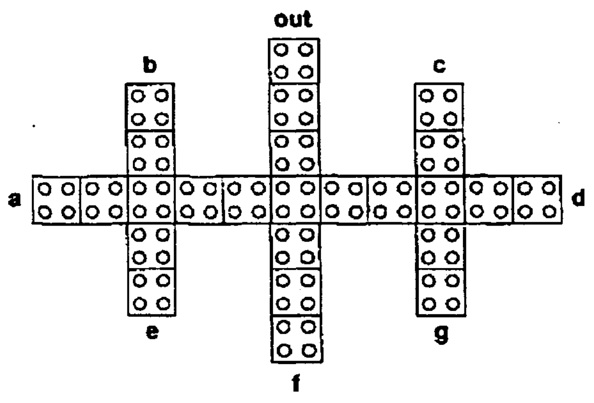

Figure 28: 7-input complex gate

3.7.1 Configuration of Four-Input AND Gate

The 7-input configuration of figure 28 can be used to form a 4-input AND gate as shown in figure 29 by restricting inputs *e*, *f*, and *g* to have a fixed polarity 0. Although the selection of signal *f* is unique, either *a* or *b* can be restricted rather than *e*, and similarly *c* or *d* can be selected for restriction rather than *g*. For example the circuit in figure 29 represents the function *out* = *abcd*.

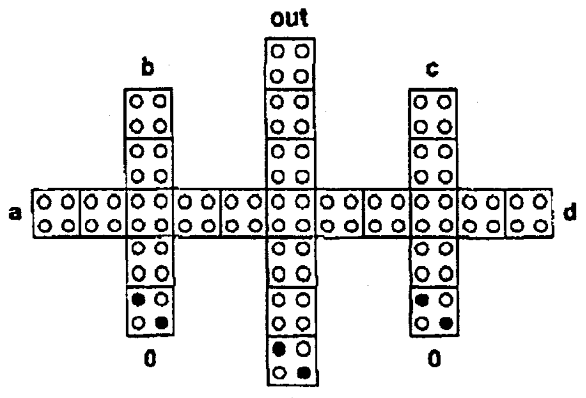

Figure 29: 4-input AND gate

3.7.2 Configuration of Four-Input OR Gate

Similarly, the 7-input configuration of figure 28 can also be used to form a 4-input OR gate. For this application the inputs *e*, *f*, and *g* are restricted to have a fixed polarity equal to 1 and the function represented is *out* = *a* + *b* + *c* + *d*. As for the AND gate, the selection of *f* is unique, however the remaining two fixed polarity cells, *e* and *g* can be interchanged for the restriction with other two inputs: *a* and *b*, or *c* and *d*.

3.7.3 Realization of Product-of-Sums Expressions

Yet another configuration of cells in figure 28 can realize a Product-of-Sums representation as shown in figure 30. In this case, *f* is restricted to have a fixed polarity equal to 0. Then either input *a*, *b*, or *e* is restricted to a fixed polarity 1. Likewise, input *c*, *d*, or *g* is also restricted to a fixed polarity 1. For example, the circuit in figure 30 represents the function *out = (a + b)(c + d),* where inputs *e* and *g* are restricted to 1.

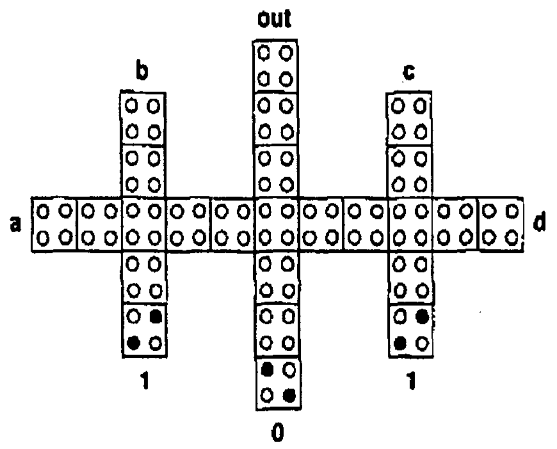

Figure 30: Product of Sum representation

3.7.4 Realization of Sum-of-Products Expressions

Yet another reconfiguration of the 7-input gate of figure 28 can be used as an implementation of a Sum-of-Products representation as shown in figure 31. In this configuration, *f* is restricted to have a fixed polarity 1. Either input *a*, *b*, or *e* is restricted to a fixed polarity equal to 0, and similarly, either input *c*, *d*, or *g* is also is set to 0. For example the circuit in figure 31 implements the function *out* = *ab* + *cd*, where inputs *e* and *g* are restricted to fixed polarity 0.

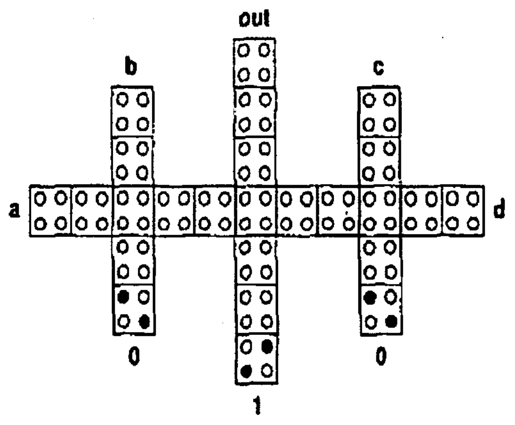

Figure 31: Sum of Product representation

3.8 QCA Implementation

The QCA concept is still in the trial phase. In consequence, there may be several different implementations possibilities, none of them determined as a leading one at this point. The basic concept, however stays the same, and is based on a bi-stable cell, which must interact locally with its neighbors such that information processing can be performed as previously described. The cell is not required to remain quantum mechanically coherent throughout the computation, thereby giving options for many non-quantum mechanical implementations. As a result, several ideas have been proposed to verify the QCA concept, as well as provide a way for an ultimate realization of QCA technology. The following four different classes of implementation have noticeably emerged.

- Metal-Island,
- Semiconductor,
- Molecular,
- Magnetic,

The *metal-island implementation* was introduced as a means of proving the QCA concept. It is based on the Coulomb blocking phenomena of nano-structures, and was shown to exhibit the electron switching as required by a QCA cell [64]. This type of implementation does not have the structural properties for a scalable design, and, as a consequence, was only meant as a proof-of-concept solution. The aluminum metal-islands are used to represent the quantum dots of a QCA cell. The metal-islands in initial experiments are in order of $1\mu m$, and therefore, the system had to be cooled to extremely low temperatures for electron switching to be observable.

Semiconductor implementations are advantageous in that they can potentially use the highly sophisticated and matured semiconductor fabrication processes. However, as of today semiconductor fabrication processes have not yet reached a point where mass production of devices with features only a few nanometers in scale is possible. To make small features feasible, serial lithographic techniques, such as electron beam lithography, are used, which are not currently suitable for mass production of devices. Semiconductor QCA technology uses nano-structure quantum dots to trap electrons [65]. The standard QCA cell is produced, where polarization is encoded in charge position, and interactions are electrostatically coupled. There has been some research into the potential realization of a quantum computer using semiconductor QCA [66].

At present, many researchers are looking for possible realization of QCA cells using single molecules [5, 25, 67]. This type of implementation of QCA offers a number of advantages including highly symmetric cell structure, very high operating speeds, room-temperature operation, and very high device density. Although molecular QCA has many attractive features, there are still many challenges to be overcome before any molecular based computing technology is available for general use. Some of these challenges are: the selective placement of molecules on a surface, the realization of a mechanism for performing I/O operations with single molecules, determining which molecules are most suitable for QCA operation, and the design of a clocking technology that can provide the clocking zone granularity required for complex circuit design.

Recently, Magnetic QCA (MQCA), has focus attention of many researchers [68], [69]. MQCA is based on interacting magnetic nano particles. The magnetization vector of these nano particles is analogous to the polarization vector in electronic QCA. The information is propagated via magnetic exchange interactions, as opposed to the electrostatic interactions in all other implementations. Although this technology is referred to as *magnetic quantum cellular automata*, the term "quantum" in this case represents the quantum mechanical nature of the exchange interaction and not electron tunneling, as in the electronic QCA. One of the reasons of considering such a technology is that MQCA cells would operate at room temperature, even for large device features on the order of a few hundred nanometers.

Chapter 4: QCADesigner- Simulation and Layout tool

For fast and reliable QCA circuit designers require a rapid and accurate simulation and layout tool. Developed in ATIPS Laboratory, at the University of Calgary, QCADesigner is the first publicly available design and simulation tool for QCA. This easy to use CAD tool is freely available to the research community via the Internet. It is capable of simulating complex QCA circuits on most standard platforms. One of the most important design specifications is that other developers should be able to easily integrate their own utilities into QCADesigner. This is accomplished by providing a standardized method of representing information within the software. As well, simulation engines can easily be integrated into QCADesigner using a standardized calling scheme and data types [52, 53]. A comprehensive tutorial is provided in the developers' website [13], so only a very brief basic introduction is presented here with the aim of acquainting the reader to the tool. For more involved reader, the website is suggested for a more rigorous introduction.

4.1. Creating Design

QCADesigner offers quite a number of flexibility for creating a complete and complex design. First, an individual cell or an array of cells can be easily added on a layer. The function of an individual cell can be set to normal, fixed, input or output cell as required. A *normal* cell has no special function other than to switch according to the influence of neighboring cells. A *fixed* polarization cell will remain at the chosen polarization during the simulation regardless of the effect of neighboring cells or the clock. *Input* cells will have a polarization that is determined by the simulation engine. The input values will be set according to the Simulation Type. *Output* cells act just like normal cells in that they are directly affected by their neighbors. The polarization of the output cells is recorded throughout the simulation and plotted in the graph dialog when the simulation is complete. Simulation can be done with all possible input combinations by choosing the Exhaustive Verification or selecting input vectors manually using a Vector Table.

The tool also allows on creating different layers with different labels and order. Layers permit on the separation of objects into different groups which can be manipulated together. They also separate different objects by virtue of their type, i.e. cells go into appropriate cell layers. Layers exist for cells, drawing objects, substrates, etc. Each layer maintains a set of template objects (for each object type that it may contain). When new objects are created, they will have the same properties as these templates.

4.2. Layer Ordering of Layout

The order, in which layers appear in the layers list, is important for cell layers because it determines their physical relation to one another. For example, the layer which appears 3rd on the list is directly above 2nd listed layer but below the 4th one. A cell layer, which appears at the top most position in the list, is considered to be at the lowest physical location. In consequence, all remaining layers appearing below are considered to be physically stacked on top of the layer form the top of the list. To accommodate multilayer circuits, it is possible to change the appearance of cells. This does not affect their simulated behavior. It is merely a means of visually identifying cells overlapping on different layers. In the cell libraries, the following convention is observed:

- Cells on the main layer are left unaltered, except for those underlying "via" cells.
- Cells on "via" layers and those accessing "via" layers are drawn using the circular appearance.
- Cells on "crossover" layers are drawn using the "X" appearance, except those accessing "via" layers.

This creates crossovers according to the following convention, figure 32:

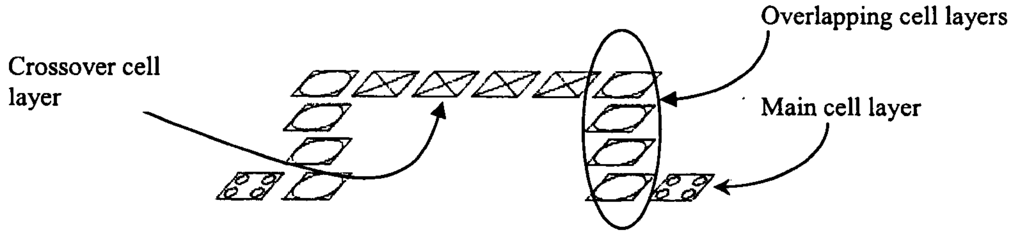

Figure 32: A multi-level crossover in QCA

4.3. QCADesigner Signal Clocking

In order to control the flow of information in a QCA circuit, it has been shown that four clock signals are sufficient [61]. Each of the clock signals is shifted in phase by

90 degrees. These four clocking zones are implemented in QCADesigner, and each cell can be independently connected to any one of the four clocking zones. One of the main differences between circuit design in QCA and circuit design using conventional CMOS technologies and devices is that QCA logic circuit has no control over the clocks. This means that information is transmitted through each cell without retention. Each cell erases its own state every clock cycle. As a result, memory units must be created using loops of cells.

4.4. Simulation Engines

Currently QCADesigner has two distinct simulation engines. Each of the two engines has a different and important set of benefits and drawbacks [13].

4.4.1 Coherence Vector Simulation Engine

The coherence vector simulation is based on a density matrix approach. It can model dissipative effects, as well as perform a time-dependent simulation of a design. As with the other simulation engines, it assumes that each cell is a simple two-state system, for which the following Hamiltonian can be constructed:

$$H_i = \sum_j \begin{bmatrix} -{}^1\!/_2 P_j E^k_{i,j} & -\gamma_i \\ -\gamma_i & {}^1\!/_2 P_j E^k_{i,j} \end{bmatrix},$$

where $E^k_{i,j}$ is the kink energy between cell i and j. This kink energy is associated with the energy cost of two cells having opposite polarization. P_j is the polarization of cell j. γ is the tunneling energy of electrons within the cell, and is directly connected to the clock; i.e. the clock value is the tunneling energy. The summation is over all cells within an effective radius of cell i, and can be set prior to the simulation.

The coherence vector λ is a vector representation of the density matrix ρ of a cell, projected onto the basis spanned by the Identity matrix and the Pauli spin matrices σ_x, σ_y, and σ_z. The components of λ are found by taking the Trace of the density matrix multiplied by each of the Pauli spin matrices; i.e.

$$\lambda_i = Tr\{\hat{\rho}\hat{\sigma}_i\} \qquad i = \{x, y, z\}\ .$$

The polarization of cell i, P_i, is just the z component of the coherence vector.

$$P_i = \lambda_{z,i}\ .$$

The Hamiltonian must also be projected onto the spin matrices as:

$$T_i = \frac{Tr\{\hat{H}\hat{\sigma}_i\}}{\hbar} \qquad i = \{x, y, z\}\ .$$

The vector Γ represents the energy environment of the cell, including the effect of neighboring cells. We can evaluate the explicit expression of Γ by substituting it into our Hamiltonian. This explicit expression is:

$$\vec{\Gamma} = \frac{1}{\hbar}\left[-2\gamma, 0, \textstyle\sum_{j\in S} E^k_{i,j} P_j\right].$$

S is the effective neighborhood of cell i. The equation of motion for the coherence vector including dissipative effects is:

$$\frac{\partial}{\partial t}\vec{\lambda} = \vec{\Gamma} \times \vec{\lambda} - \frac{1}{\tau}\left(\vec{\lambda} - \vec{\lambda}_{ss}\right),$$

where τ is the relaxation time, i.e., a time constant representing the dissipation of energy into the environment. λ_{SS} is the steady state coherence vector defined as:

$$\vec{\lambda}_{ss} = -\frac{\vec{\Gamma}}{|\vec{\Gamma}|}\tanh(\Delta).$$

Δ is the temperature ratio defined as:

$$\Delta = \frac{\vec{\hbar}|\vec{\Gamma}|}{2k_B T},$$

where T is the temperature in Kelvin, and k_B is Boltzmann's constant.

The simulation engine evaluates the motion equation, given in a form of a partial differential equation. An explicit time marching algorithm is used for that. For each time step the Γ and λss of each cell is evaluated, and then the coherence vector for each cell is stepped forward in time.

There are several options for the simulation engine, including: temperature which has an effect on the steady state vector, relaxation time which determines how quickly the system settles to a steady state, time step between samples, total duration of the simulation, clock signal which is tied directly to the tunneling energy in the Hamiltonian and radius of effect. As the interaction effect of one cell onto another decays inversely with the fifth power of the distance between cells, it is not needed to consider each cell as affecting every other cell. The radius of effect determines how far each cell will look to find its neighbors. The figure 33 shows such radius of effect in neighboring cells.

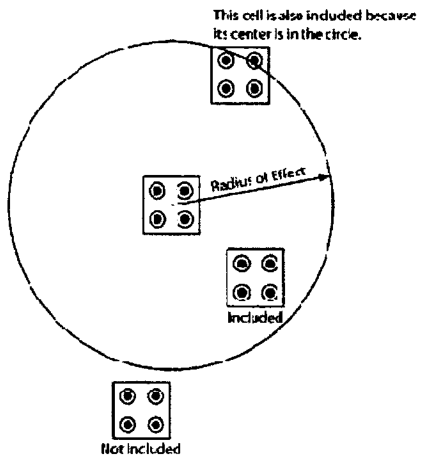
This cell is also included because its center is in the circle.
Radius of Effect
Included
Not Included

Figure 33: Radius of effect

4.4.2 Bistable Simulation Engine

The bistable simulation engine assumes that each cell is a simple two-state system. For this two-state system, the Hamiltonian stated above can be constructed. Using the time-independent Schrödinger equation it is possible to find the stationary states of the cell in the environment described by this Hamiltonian:

$$H_i \psi_i = E_i \psi_i \,,$$

where H_i is the Hamiltonian given above, ψ_i is the state vector of the cell, and E_i is the energy associated with the state. When evaluated, this Eigen value problem reduces to the following simple relation:

$$P_i = \frac{\frac{E^k_{i,j}}{2\gamma}\sum_j P_j}{\sqrt{1+\left(\frac{E^k_{i,j}}{2\gamma}\sum_j P_j\right)}} \,,$$

where P_i is the polarization state of the cell, and P_j is the polarization state of the neighboring cells. Since experimentally determined switching times are not available, the simulation does not include any timing information. Using this response function the simulation engine calculates the state of each cell with respect to other cells within a preset effective radius. This calculation is iterated until the entire system converges within a predetermined tolerance. Once the circuit has converged, the output is recorded, and new input values are set. It is believed that although this approximation is sufficient to verify the logical functionality of a design, it cannot be extended to include valid dynamic simulation. However, due to its simplicity this simulation engine is able to simulate a large number of cells very rapidly, and therefore provides a good in-process check of the design.

Like coherence vector simulation engine, also bistable simulation engine offers the options to set radius of effect, relative permittivity, clock signal, layer separation, randomization of simulation order, and animation of the simulation to check the change in cell polarization. Additionally, the number of samples can be changed. The total simulation is divided by the number of samples. For each sample, the simulation engine looks at each cell, and calculates its polarization based on the polarization of its effective neighbors (determined by the radius of effect). Also while processing every sample, each cell is converged by the simulation engine. The sample will complete when the polarization of each cell has changed by less than a predefined value set as convergence tolerance; i.e. loop while any design cell has (old_polarization - new_polarization) > convergence_tolerance.

4.5. Design Verification

Each simulation engine can perform an exhaustive verification of the system or a set of user-selected vectors.

4.5.1 Exhaustive Verification

The simulation-based exhaustive verification will simulate all possible combinations of the input vectors. If number of inputs is n, then this will create and simulate all 2^n possible vectors in increasing order.

4.5.2 Vector Table Simulation

The vector table simulation type allows specifying the input vectors, as well as the order in which they are applied to the inputs of the circuit. The vector table simulation is only available if cells are already designated as circuit inputs. Then for each available input a binary value can be assigned in each vector.

Chapter 5: QCA Physical Defects and Abstracted Faults

In the actual circuit, physical defects may occur due to some manufacturing flaw in mass production. Sometime these defects cause recurrent errors in their application, and functionally differ from the original intended specification. These are the defects that one has to test for, and determine whether a circuit is faulty or not. In doing so, the physical defect is abstracted into suitable fault model, which should accommodate for maximum possible varieties of defects. For simplicity of test methods, manufacturing defects are generally checked for at the logic level.

In order to achieve the above, it is necessary to analyze the defects that occur in a particular process technology, and develop fault models best suited for pinpointed defects. It should be noted that the science behind the particular fault model adopted should suffice in describing the behavior of the defects, and its application should be versatile enough to achieve high fault coverage.

QCA circuits are not in mass production phase at this moment, so it is not possible to accurately address defects which occur during manufacturing. However, one could have an estimate what would be the case when mass production stage is reached. In this attempt, some of the major faults predicted for functional error by the work of Tahoori et al [2] of Northeastern University, Boston are re-simulated and characterized.

In this work however, the simulator used is QCADesigner 2.0.3, and feature size was chosen to be 18nm X 18nm for the cell using 5nm dots.

5.1 Implementation-specific Faults: Sources of Defects

5.1.1 Metal Dot Implementation

In this implementation of QCA cells, steps include electron beam lithography (EBL), metal deposition and oxidation. The fabrication process is, in fact, similar to CMOS one. Hence, the probable defects are also predicted to be of similar nature. Any of the above steps can introduce faults; more over particles in clean room can also constitute extra defects. Defects during EBL stage is most likely to be junctions in the wrong spot, over or under exposed junctions, which may result in improper switching of cells. An over exposed junction will tend to release electrons too soon, while in contrast, and under exposed junction will tend to hold an electron for more energy than originally intended for. Thus a defective cell could be either fluctuating (overexposed), or be stuck to a particular switching position (underexposed).

In deposition phase, the most likely defect could be dots with wrong size and shape, thereby having an unwanted reduction of energy degeneracy resulting in a particular positional preference for the intended mobile electron within the defective cell.

Due to close spacing of cells and the dots within, Coulumbic interaction between the electrons could give rise to an unwanted parasitic capacitance, which in the end could introduce a cross talk. This, of course, is more of a noise problem than defect, and correct biasing of junction voltage could overcome the limitation within the given device specification.

5.1.2 Molecular Dot implementation

One way of implementing the molecular dot is to first generate a layer of candidate molecules on a suitable wafer, then cutting out trenches by EBL for QCA molecular deposition. Next, the produced wafer is submerged into a QCA molecule media so that molecules could self-assemble in accordance to the fabricated trenches. Here the same defects could occur associated with EBL, such as extra or missing molecule due to inaccurate trench definition. Such defects are likely to manifest into erroneous switching.

Then again there remains a defect possibility associated with self-assembly of molecules. As it cannot be predicted beforehand how the molecules would self-assemble, an error site could exist in all three dimension of the assembly. Even when molecules assemble in a smooth planner manner, different molecule could attach with different surface angles, thereby creating different dot projection. This could result in individual dots to position at slightly offset place from the intended one. In consequence, this could affect the device symmetry within the cell and create unwanted logic affinity.

Stray charges also could potentially pose defects, as they might interact with deposited or self assembled dots, which in the end could manifest stuck at fault for a cell.

5.2 Defects Beyond Device Specification

There are some faults that may result from device specification aspects and manifest themselves at functional level. These types of errors could be addressed better in specification, and hence are not dealt with as manufacturing faults. In this thesis, faults that only occur within the given specification of a workable device and manifest in functional errors are modeled.

One such defect, which is not considered here even though it can constitute a functional error, is the random thermodynamic energy of a loaded electron, which can be large enough to cause a cell to switch incorrectly. As this error is more of a QCA fabrication concern, it should be treated within the limits of device fabrication feasibility, and a correct fabrication technology should not result in such error. The energy needed for a QCA cell to switch is ideally much more than the thermodynamic energy of an electron. The thermodynamic energy of an electron is directly proportional to k_BT and the degree of freedom. So by reducing the temperature (and since the degree of freedom is already limited in a quantum confined dot structure) this energy could be minimized. It then becomes a part of the device specification whether a particular temperature comes within the operating range of the device, and the error is taken care of.

Another defect that can be of a concern for a proper operation of the QCA circuit is the clocking error. Usually clocking circuits in QCA are implemented as wires below the substrate or by extra metal dots.

5.3 Inter Cellular Defects over Intra Cellular Defects

To study the functional error of QCA defects it is necessary to use a versatile fault model which can be efficiently simulated and present a realistic abstraction of the manufacturing defects [2][59]. A previous study in this respect [25] show that in present the QCA manufacturing process is defect-prone in both synthesis and manufacturing phases. During synthesis defects could be associated with individual dots within the cell, whereas a deposition phase would normally be associated with defects at cellular level, with individual cells being correctly fabricated. The study shows that a cell level defect is much more probable to occur than a dot level one. Following these observations, the current trend in QCA testing is more focused on cell level defects, and assumes that the individual cells are error free. This viewpoint is also adopted in this work, and only cellular defects are analyzed.

5.4 Cell Level Defects in QCA

It has been found [2] that three major cases of defects are probable to occur during manufacturing. They are:

- Cell Displacement
- Cell Misalignment
- Cell Omission

5.4.1 Cell Displacement

This is a defect, in which the effected cell is displaced from its original position. It could happen to a single cell or to many cells at once. Some of the displacement defects are shown in the following figure 34, and their manifested errors are tabulated.

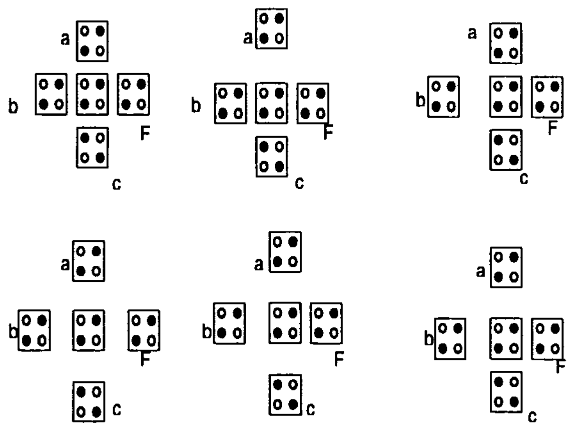

Figure 34: Different displacement faults of a majority voter,

Displaced cell A		
d ≤ 15 nm Normal Operation	d ≥ 20 nm, F=B	
Displace cell B		
d ≤ 40 nm Normal Operation	d ≥ 45 nm	
	A B C	F
	0 0 1	Z (no polarization)
	0 1 1	Z (no polarization)
	1 0 0	Z (no polarization)
	1 1 0	Z (no polarization)
Displace all input/output cells		
d ≤ 10 or 30 ≤ d ≤ 40 nm Normal Operation ≥ 45 nm F=Z (no polarization)	15 ≤ d ≤ 25 nm	
	A B C	F
	0 1 0	0/1
	1 0 1	1/0
Displace all input cells		
d ≤ 15 or d = 40 nm Normal Operation	20 ≤ d ≤ 25 nm or d = 35 nm	
	A B C	F
	0 1 0	0/1
	1 0 1	1/0
d = 30 nm		d ≥ 45 nm F=Z (no polarization)
A B C	F	
0 1 0	0/1	
1 0 1	1/0	
Displace cells A and B		
d ≤ 5 nm Normal Operation	d ≥ 10, F = C	

5.4.2 Cell Misalignment

In a cell misalignment, the direction of the defective cell is misplaced with respect to the other cells. This too could happen to any number of cells simultaneously. Some of these defects are illustrated and analyzed in the following figure 35 and table:

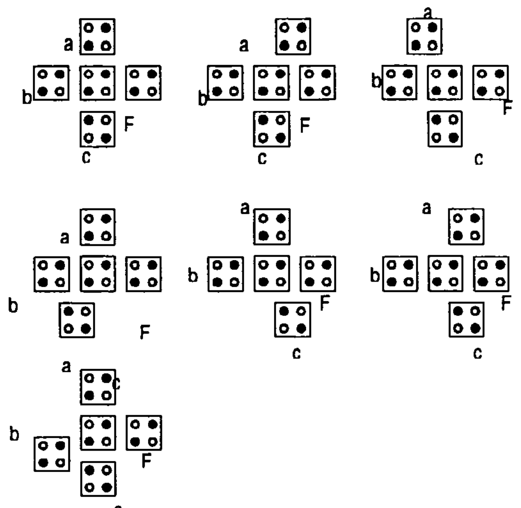

Figure 35: misplacement of cell defects in QCA majority voter

Move A towards west		
d ≤ 5 nm Normal Operation		d ≥ 20 nm, F=B
Move C towards west		
d ≤ 5 nm Normal Operation		d ≥ 10 nm, F=B
Move A, C towards west		
d ≥ 5 nm, F=B		
Move A towards east		

5 ≤ d ≤ 15 nm		d = 20 or d = 30 nm Normal Operation
		
		d = 25 nm, F =A

A B C	F	
0 0 1	Z (no polarization)	
0 1 1	Z (no polarization)	
1 0 0	Z (no polarization)	
1 1 0	Z (no polarization)	
Move A, C towards east		
d ≤ 5 nm Normal Operation	d ≥ 45 nm	
	A B C	F
	0 0 1	0/1
	0 1 1	1/0
	1 0 0	0/1
	1 1 0	1/0

5.4.3 Cell Omission

In a cell omission defect, a particular or a number of cells could just not be there at all leading to a functional error. Defects may occur in the majority voter, as well as in connecting wires. This error is particularly severe for the inverter chain and MV but not as much visible in the binary wire as far as behavioral changes in logic levels are concerned. Examples of the defects and their manifestation are given in figure 36.

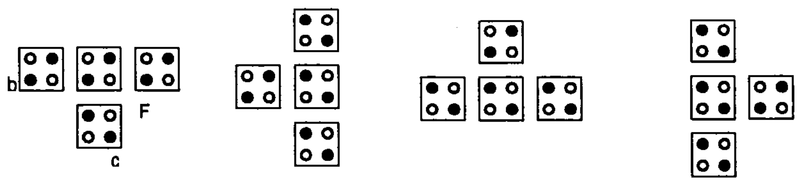

Figure 36: Cell omission fault in QCA majority voter

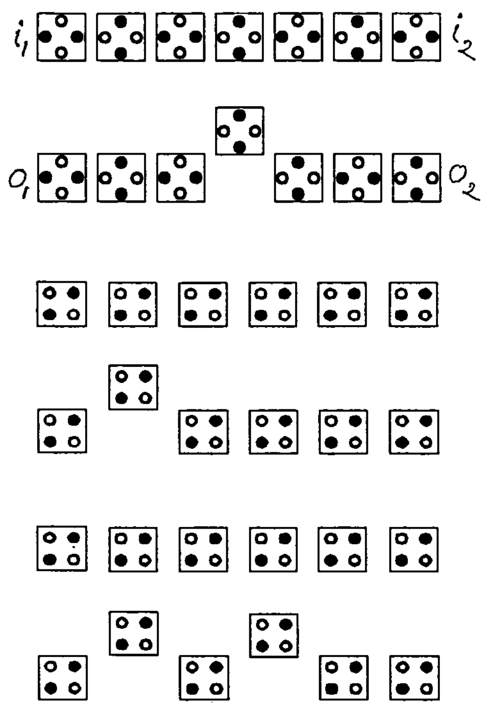

Figure 37: Defects in binary double wire

Moving Cell 1 OR Cell 2			
d ≤ 40nm Normal	d = 45 – 50nm o1 = i1 O2 – i1	d ≥ 55nm o1 = i1 o2 = Z	
Moving Cell 3 OR cell 4			
d ≤ 35 nm Normal	d = 40 - 50nm o1 = i1 o2 = ~i1	d ≥ 55nm o1 = i1 o2 = Z	
Moving Cell 1 AND Cell 2			
d ≤ 35 nm Normal	d = 40 - 50nm o1 = i1 o2 = i1	d ≥ 55nm o1 = i1 o2 = Z	
Moving Cell 1 AND Cell 3			
d ≤ 35 nm Normal	d = 40 - 50nm o1 = i1 o2 = ~i1	d = 45nm o1 = i1 o2 = i1	d ≥ 55nm o1 = i1 o2 = Z
Moving Cell 1 AND Cell 4; OR moving Cell 2 AND Cell 3; OR moving Cell 3 AND Cell 4			
d ≤ 35 nm Normal	d = 40 - 50nm o1 = i1 o2 = ~i1	d ≥ 55nm o1 = i1 o2 = Z	
Moving Cell 2 AND Cell 4			
d ≤ 15 nm	d = 20 – 25nm d = 40 – 45nm	d = 30 – 35nm	d = 50nm
Normal	o1 = i1 o2 = ~i1	o1 = i1 o2 = i1	o1 = ~i1 o2 = i1
d ≤ 55 nm o1 = i1 o2 = Z			

Fault free		
o1 = ~i1 o2 = i1		
Moving Cell 1 OR Cell 2 OR Cell 3		
d ≤ 35 nm Normal	d = 40 - 50nm o1 = ~i1 o2 = ~i1	d ≥ 55nm o1 = ~i1 o2 = Z
Moving Cell 4		
d ≤ 35 nm Normal	d = 35 - 50nm o1 = ~i1 o2 = ~i1	d ≥ 55nm o1 = ~i1 o2 = Z

5.5 Defect Analysis

After studying the defects and their affects minutely, some interesting observations could be made:

- *In most cases of displacement defects, the horizontal input B seems to have a greater impact on the functionality of the MV rather than vertical inputs A or C,*
- *Any single cell misalignment greater or equaling half a cell causes malfunction,*
- *For binary double wire, in most faulty cases, the lower wire (faulty) is dominated by the upper wire (fault free). Though in some cases, the upper wire is also complemented, the coupling does not behave like the wired bridge fault model as in CMOS,*
- *A cell displacement defect acts as a bridging fault for the inverter chain,*
- *Cell omission does not usually cause functional faults in binary wires unless the cell lies at the corner of the wire when it turns in a right angle,*
- *In inverter chains, a cell omission in any place results in unwanted complementation,*

- *Two lines can cross in the same level in QCA successfully carrying different logic value without hampering each other. However the crossing point cell defects are more prone to error in functionality than other cells.*

Considering the above, it could be concluded that the physical defects that occur in the QCA implementation manifests errors in the behavior of the circuit as listed below:

1. **Un-polarization:** un-polarization of the output, as they are left with no logic output at all.

2. **Shortening of Logic Gate:** some defects results in the output only depending on one of the inputs, thereby shortening the logic gate but still incorporating the associated delay.

3. **Unwanted Complementation:** in some cases, logic output is inverted from its intended value, thus resulting in unwanted complementation of logic.

4. **Delay fault:** some of the defects do not create a change in the functionality; instead they incur a delay in passing information, and hence degrade the performance of the circuit.

5. **Bridging fault:** some defects lead to dominating bridging faults, while some result in somewhat different bridging fault.

Chapter 6: Testing of QCA Circuits – Proposed Scheme

Logic circuits can be implemented in QCA similarly to any other conventional technology (e.g. CMOS) with AND-OR-NOT logic gates. NOT gate has explicit structure in QCA, while MVs can be converted into AND-OR logic by appropriately setting one input permanently to 0 (for AND) or 1 (for OR). These inputs are unequivocally called control inputs in QCA. The implemented logic usually contains two more pins than the number of primary inputs for any given function with one control line for 1 and one for 0. Hence an *n*-input logic implemented in QCA would require (*n*+2) inputs of which, one is permanently set to '1' and one to '0' (the control lines). On the other hand, it would not require biasing pins. From testing perspective, more pins would need to be considered for testing, but at the same time, two more inputs are available for test generation. Test sets generated based on stuck-at fault model are quite acceptable for testing conventional CMOS-based designs, though this model does not fully capture the behavior of many of the prevalent defects in CMOS fabrication process. Thus, it may be viable to investigate the effectiveness of stuck-at test sets for QCA defects even though QCA defect mechanisms cannot always be modeled by the stuck-at model. This issue is investigated somewhat in detail in this chapter.

Defects are injected into QCA-based circuits, and their effects on the fault-free behavior are investigated, as well as the evaluation of effectiveness of various test sets is performed. In this regard, unique testing properties of QCA technology have been taken into account. Constant testability (C-testability) of QCA designs based on MVs has been observed. Previously proposed design scheme [8] is discussed, as well as the problem with resulting malfunction of the scheme is identified. A new design-for-test method is introduced to overcome the existing testing problems, and to improve testability of QCA designs. A new approach for modular designs is also presented.

6.1. Stuck-at-Value Fault Model: Assumptions and Limitations

Stuck-at value (s-a-v) fault model has long been successfully used to identify the defects of many technologies. Though it has some serious limitations, it has always delivered effective ways of test generation. In the case of QCA, some basic properties of the single stuck-at value fault model are pointed out, and an attempt to give an insight to how it would affect this case is presented here.

To start with, the main postulations of the single stuck-at value fault model could be stated as follows:

1. *Only one stuck-at value fault may occur at any given instance,*
2. *Stuck-at value faults will either assume the faulty line to be stuck at logic 1 or logic 0, i.e. it does not allow any other values of the faulty line,*
3. *Faults only occur at interconnections between gates, while gates themselves are not affected.*

Considering the above postulations, it would be safe to state that regarding the first conjecture; a single stuck-at value fault model identifies $2n$ faults among the 3^n-1 possible faults for a circuit with n interconnects [63]. This, in fact has proven to be enough to detect the faults with considerable proportion of fault coverage, so single stuck-at faults will be practical for the application in case of QCA.

The second postulation requires some more explanation, and perhaps even extending for QCA circuits. Some of the faults in QCA result in un-polarization, i.e. output is left in a state without any logic. This could be a potential problem in the definition of s-a-v faults which does not allow any output value except a logical '1' or '0'. So for the case of QCA, this s-a-v fault testing postulation must be modified, and state that if the output is not the correct one, then it is a faulty output. Thus if, for example, a correct circuit should result in a particular output to be '0', then having a '1' or an un-polarized output in that node should be considered a faulty response, and hence, the limitation of having only '1' or '0' as output would be extended.

The third supposition seems only logical, as fault is an abstraction of the physical defect, and conventional logic (like CMOS) uses different materials and structures for active logic gates and interconnects. However, in the case of QCA interconnects are actually made from the same basic cells, so a distinction between the logic gate and interconnects actually disappears in this case. Therefore, this third postulation actually becomes a limitation for the QCA logic. However, as shown in previous chapter, defects in QCA could be projected to be occurred in the interconnect at a logic level, making this postulation transparent for testing QCA logic.

6.2. Testing of Single Majority Voter

Testing of a single majority voter is straightforward. Logic '1' in the control input configures it as an OR gate, and applying a test vector '00' results in '0' for a fault free case thereby testing stuck-at-1 (s-a-1) faults. Similarly, '0' in control input configures it as an AND gate, and a test vector '11' results in '1' in correct operation, thereby testing stuck-at-0 (s-a-0) fault. Therefore, generally two test vectors suffice to test individual majority voter: {UAB} = {011} (s-a-0), and {UAB} = {100} (s-a-1), where U is the control input and *A, B* are data inputs.

6.3 Testing of Chained MVs

It has been observed [8] that any number of MVs connected in a chain form is testable with fixed number of test vectors, i.e. the chain is C-testable. For example, consider the one dimensional array in figure 38. A 100% single stuck-at fault test set for a single MV has a minimal length of 4: {010011100101}, {001011100101}, {001010101110}, etc. *A* is the primary data input and B_i and C_i are the control inputs. One could set the primary input *A* to '0' and control inputs $B_i = 0$, $C_i = 1$ for all MVs in the chain. Setting $C_i=1$ converts any ith MV to an OR gate with F_{i-1} and B_i as inputs. Therefore, having A=0 and B_i=0 for all *i*, the test vector for these inputs will be '00', and any stuck-at-1 fault on F_{i-1} and B_i will be detected. Similarly, by applying $A=1$ and $B_i = 0$, any stuck-at-0 fault on F_{i-1} or C_i will be detected (in this case MVs take the form of AND gates with inputs F_{i-1} and C_i). B_i stuck-at-0 or C_i stuck-at-1 faults can be detected with vectors: $B_iC_i = 10$ and setting *A* to '1' and '0' respectively.

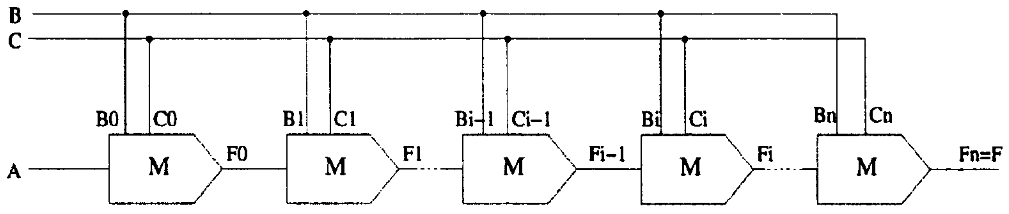

Figure 38: Testing of a chain of MVs.

A stuck-at-1 can be detected by setting *A* to '0' to sensitize the fault, and lines *B*, *C* to opposite values (i.e. *BC=01* or *10)* to propagate the fault to the output *F*. Similarly, A stuck-at-0 is detected by '101' and '110'. Hence, any 1D chain of MVs irrespective of its length can be tested for all stuck-at faults by only four vectors, i.e., is C-testable. Another noticeable thing in this 1D chain is that any number of MVs can be faulty; detection with 100% coverage of multiple faulty MVs is possible due to the AND–OR nature of the MV chains during testing.

6.4 Testing of QCA Circuits: Case Study

Somewhat ingenuous proposition of Tahoori *et al* [12] could greatly augment the testing of s-a-v faults in the QCA data path. The arbitrary function given in figure 39 is considered. The implemented function is Z = ((AB+CD) ·E) + F

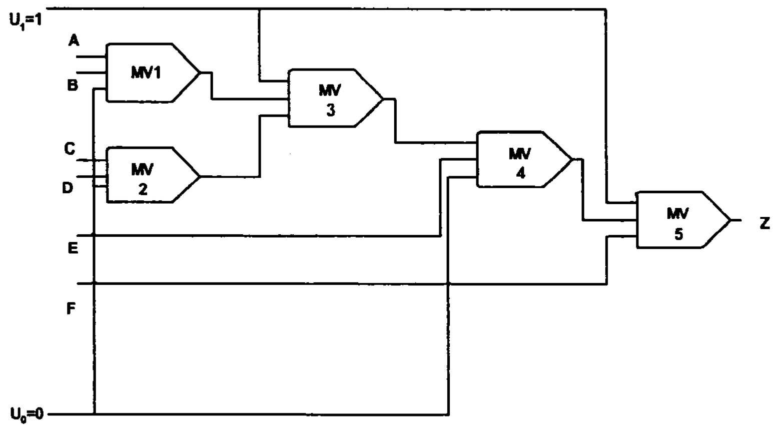

Figure 39: An arbitrary function implemented with QCA MVs.

This QCA realization includes two extra inputs U_0 and U_1 as control inputs. They do not lie in the data path of the circuit, but nevertheless they are logic inputs. These control inputs offer extra controllability to the circuit.

6.4.1. Stuck-at-0 Fault Testing

In this case both control inputs are set to '0' so that all the gates act as AND gates, as shown in figure 40. This obviously changes the function to Z = A·B·C·D·E·F, in test mode, it should not be of any concern regarding the correctness of the operational logic.

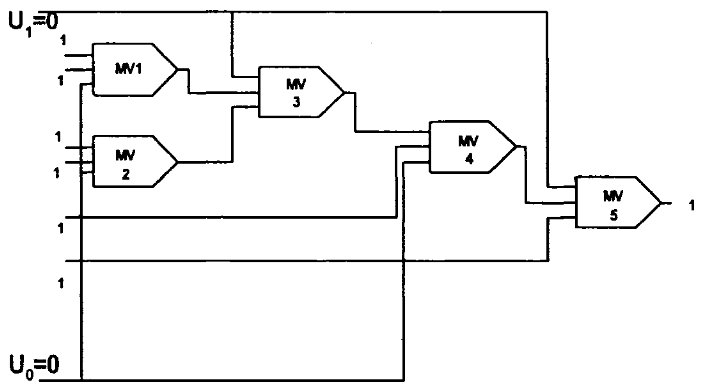

Figure 40: Testing for stuck-at-0

By setting all the data inputs to '1' the output is obtained as Z = 1. No matter what the size and function of the implemented logic is, with both control inputs set to 0 one would always attain this result. But if there exists any s-a-0 fault in the data path, then the faulty response (output) would be '0'. So by only one vector {U_0U_1ABCDEF} = {00111111} for any arbitrary logic, of any size and function, all single s-a-0 faults in the data path can be detected.

6.4.2. Stuck-at-1 Fault Testing

To test any s-a-1 fault potentially located in any node of the data path, both the control inputs of the QCA circuit are set to 1 and all the data inputs to 0 as given in figure 41. This would change the logic in testing mode to Z = A + B + C + D + E + F, and a fault free circuit will result in Z =0, while any s-a-1 fault in the circuit data path would result in a faulty response of Z = 1. So one would again detect all s-a-1 faults for this arbitrary logic by only one test vector {U_0U_1ABCDEF} = {11000000}.

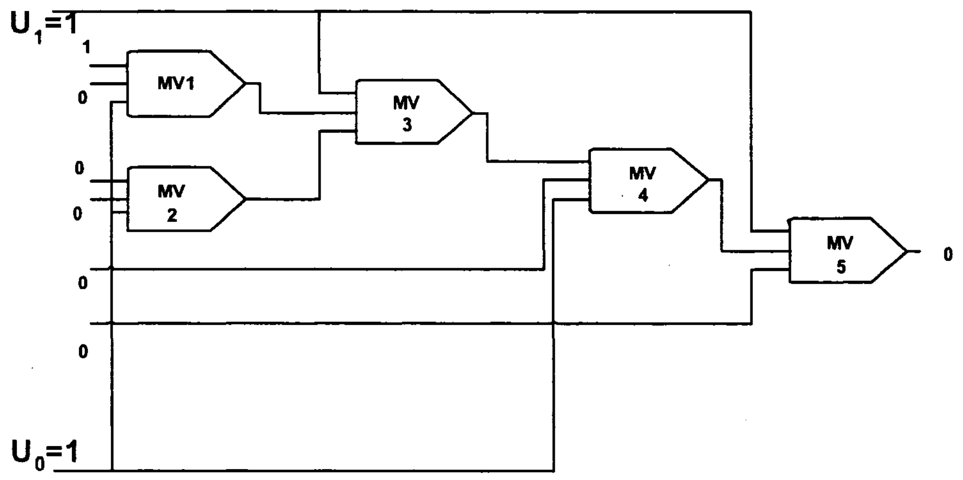

Figure 41: Testing for stuck-at-1

6.4.3. General Test Set for a Network of MVs

A logic network composed of only AND and OR gates is implemented using QCA MVs. As stated earlier QCA network has two additional control inputs other than the primary inputs. One control input is connected to one of the three inputs of all the MVs, is set to '1', and is used to implement OR function. The other control input feeding all the MVs helps to realize an AND function, and is set to '0'. Only two test vectors are needed for the MV implementation to detect all s-a-v faults in the fault list of the original design.

In order to detect s-a-0 faults, one converts AND-OR logic to a network of only AND gates by setting all MV control lines to '0'. So irrespective of the original function, the circuit will realize AND operation of data inputs. Then feeding '1' to all non-controlling inputs of this block causes every gate and path to become transparent for s-a-0 fault propagation. Hence any s-a-0 fault in any location will propagate to its corresponding output. This procedure thus sensitizes any s-a-0 fault. In a fault-free case every AND gate is fed with '1' in all of its inputs, and primary outputs become '1'. If there remains any s-a-0 fault, it will propagate all the way to its corresponding output(s). Additionally, a s-a-1 fault can be tested by first turning all gates to OR ones by setting all MV control lines to '1', and then feeding all non-controlling block inputs with '0'.

This fault list is much smaller than the complete fault list for 100% stuck-at coverage (stuck-at fault on all nodes) because all nodes in the original design are covered by those two test vectors and removed from the fault list in this phase. This results in a reduction of test generation time and the number of test vectors.

6.5 Testing of Control Lines

The generated test vectors discussed above are suitable only for data paths. Therefore, the consideration of control lines must follow. Testing of a control line of MV for stuck-at faults can be done in two ways. The first method is very simple and applicable only when the two other inputs of MV are possible to have opposite values. By applying '1' and '0' on the control line, stuck-at-0 and stuck-at-1 faults on the control line will be detected, respectively. If for a particular vector at the primary inputs, the two non-controlling inputs of each MV have different values, then all stuck-at faults on the control lines can be detected by only two test vectors. Due to different values of non-controlling inputs, the control inputs then determine the polarization of the output cell. So if the control input is set to '1', in fault free case it will output '1', while the presence of s-a-0 fault will set the output to '0'. The same thing can be observed for detecting any s-a-1 fault. In the example presented in figure 42, the two vectors required to test control line faults are: U_0U_1 ABC = (11100, 00011). The first (second) vector detects stuck-at-0 (stuck-at-1) faults on the control lines. It can be noted that the non controlling inputs of each MV have opposite values in each test vector.

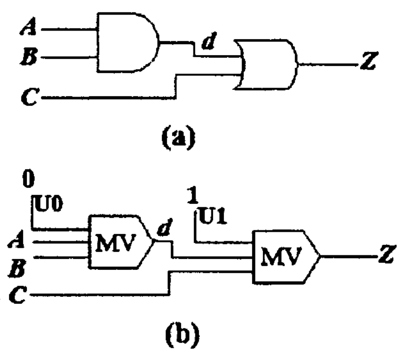

Figure 42: (a) Logic implemented in CMOS, (b) The same logic implemented in QCA.

The above method is not applicable to test control lines for complex circuits, where it is not possible to set the non-controlling inputs of all MVs to have opposite values simultaneously. In this case, testing for stuck-at faults on control lines cannot be accomplished by two tests. To detect stuck-at faults on the control lines and control inputs, conventional (combinational) ATPG tools can be used. The network of MVs is first transformed into a hierarchical gate-level netlist. Each MV is replaced by a hierarchical cell implementing the majority function. Some unique features offered by QCA discussed in next subsection can be employed to minimize the number of vectors.

6.6. Unique Stuck-at Test Properties of QCA

Since logic designs are implemented as a network of MVs and inverters (as the universal logic set) in QCA technology, it is important to investigate the properties of these networks, especially for test execution. As shown through the following statements, these networks have unique and interesting testing features, which cannot be achieved in conventional CMOS realizations [12].

Property 1: *In a MV with inputs a, b, and c, and output f if all inputs are flipped, from (a,b,c) to (a',b',c'), the output will also flip from f to f'.*

This property is not the case for logic functions like AND, NOR, etc. For example, a three-input NOR gate with inputs 100 gives output 0. If the inputs are flipped to 011, then the output will remain 0.

Property 2: *If there is an inversion at any input and/or output of the majority voter, property 1 still holds.*

Property 3: *In a MV with inputs (a,b,c) the s-a-v fault on any input/output line of the voter is detectable by (a,b,c) iff the s-a-v' on that line is detectable by (a',b',c').*

If any s-a-v fault occurs at (for example) input line *l*, then the fault is detected if and only if the value of *a* (connected to line *l*) is *v'*, and the other inputs (*b* and *c*) have opposite values (*bc=01* or *10*). As a result, *a'b'c'* have opposite values also (a'*b'c'=v'01* or *v'10*). Hence *a'b'c'* detects stuck-at-*v'* fault for line *l*. This property does not hold for

CMOS (e.g. a two-input AND gate with test vector (11) detects stuck-at-0 at both inputs. The complement of this vector (00) does not detect any single stuck-at-1 on the inputs).

Property 4: *If there are some inversions in any inputs and/or output of the majority voter, property 3 still holds.*

Property 5: *In any arbitrary network of MVs and inverters, if all bits of initial vector V are flipped to obtain V', all nodes in the network will be flipped.*

Property 6: *In any arbitrary network of MVs with input vector V, if s-a-v at any node n is detected by V then s-a-v' on node n will be detected by V'.*

Properties 5 and 6 are very attention-grabbing, and have proven exclusive features of a network of MVs and inverters. Based on property 5, the test vector pair (V, V'), where V is any arbitrary vector, causes a transition on all nodes of the network. Also, the three vectors (V, V', V), cause both fall and rise transitions on all nodes in the network. Hence, 100% toggle fault coverage is applicable for this test set.

Based on property 6, the fault list for any network of MVs and inverters can be divided into two parts: one part will detect s-a-v fault, while the other for detecting s-a-v' faults. So a single fault per node (either s-a-0 or s-a-1) is sufficient to be considered. If a vector (V) detects one stuck-at fault on that node, its inverse (V') will detect the other stuck-at fault on that node. As a corollary, this feature can be exploited to reduce the size of the fault list and, hence, Automatic Test Pattern Generation (ATPG) execution, for the control inputs (to be generated by ATPG) substantially.

At a glimpse the properties notify the following realization:

- *Conventional ATPG tools could be used where required (e.g. control line),*

- *A hierarchical net list representation of the QCA implementation of the logic could be used to simulate any MV based QCA logic,*
- *Test vector for any node s-a-v fault automatically finds s-a-v' fault for the same node. So the fault list is halved as each node only needs associated to one fault,*
- *A test set of (V,V',V) will test 100%toggle fault in all the nodes.*

6.7. Fault Masking Due to Inverters: Why the Above Scheme Fails?

One important aspect of the testing technique stated in the case study is that the logic exemplified here is a non-inverting one, i.e. no inverted signals are confronted. As an inverted signal would demolish the advantageous situation gained here, an improvement is a must.

Test generation for QCA logic networks implemented by MVs realizing AND and OR functions are considered. However, often the universal logic network contains inverters; hence a general case is presented here. Inside the logic network, inverters prevent the propagation of all faults by the proposed two test vectors (stated in section 6.4). As a result, some faults remain unnoticed, and additional test vectors are needed to detect them. This is shown in figure 43 where an inverter is present on the data path. The test vector $U_0U_1ABCDEF$ = (11000000) which supposedly detects all stuck-at faults on all nodes, cannot detect *E*/1 and *F*/1 (it detects *j*/0 and *Z*/0 instead of *j*/1 and *Z*/1).

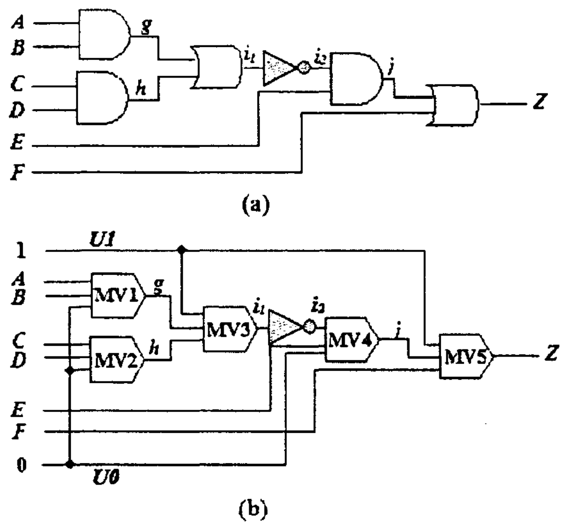

Figure 43: (a) Inverter in a general CMOS logic, (b) Inverter in QCA MV circuitry [28]

6.8. Literal Input AND-OR Logic QCA Implementation

In fact, any inverting arbitrary logic which incorporates inverters could be implemented with only AND-OR logic and literals as inputs. Therefore, the proposition is to apply some inverting scheme to the primary inputs and then feeding such prepared literals from the inverting circuitry to the AND-OR logic to obtain the desired function. Thus any arbitrary logic can be realized by AND-OR gates with literals of both polarities as inputs. As shown in previous sections, the fixed inputs of the majority voters, i.e., the control inputs, can be manipulated to convert the main circuit to a connection of AND or OR gates. As inverter chain can complement any input, it is only necessary to apply inverter chains to interconnect primary inputs to the AND-OR logic, and use ripper cells in appropriate locations to extract the desired non-inverted or inverted value of the inputs. The scheme could be represented as the block diagram in figure 44. Thus the testing of such logic would result in generating only two test vectors (as stated in previous subsections).

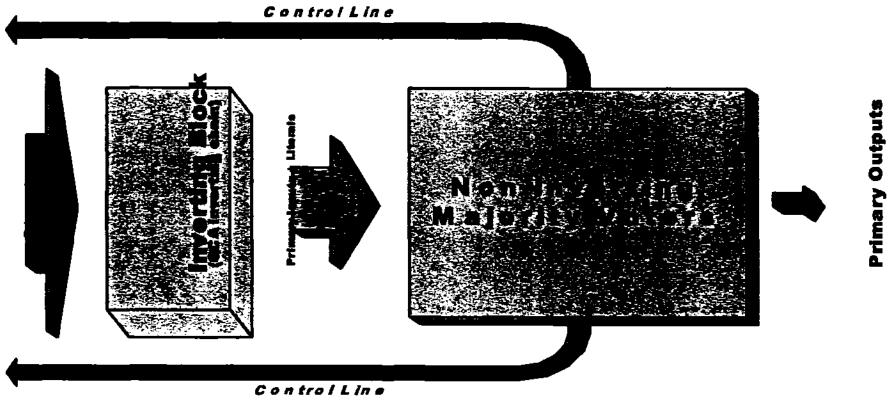

Figure 44: Block diagram for a literal input QCA AND-OR logic implementation

The design is partitioned into two blocks: the first one is the inverting block, which generates the literals, the second, following block consists of MVs implementing the AND–OR network. This architecture also saves area because expensive inversion inside the logic block is prevented (the area of an inverter is twice that of an MV). De-Morgan's law can be exploited to change the structure of the design such that all inversions are pushed back to the primary input level. Thus it is always possible to transform a general logic network of AND, OR, and NOT functions (universal logic) into an equivalent network consisting only of AND and OR functions, which take literals (variables and their complements) as inputs. Therefore, a network of AND and OR functions can be implemented only by MVs and the same test generation approach presented earlier can be used. An example of this implementation is given in figure 45.

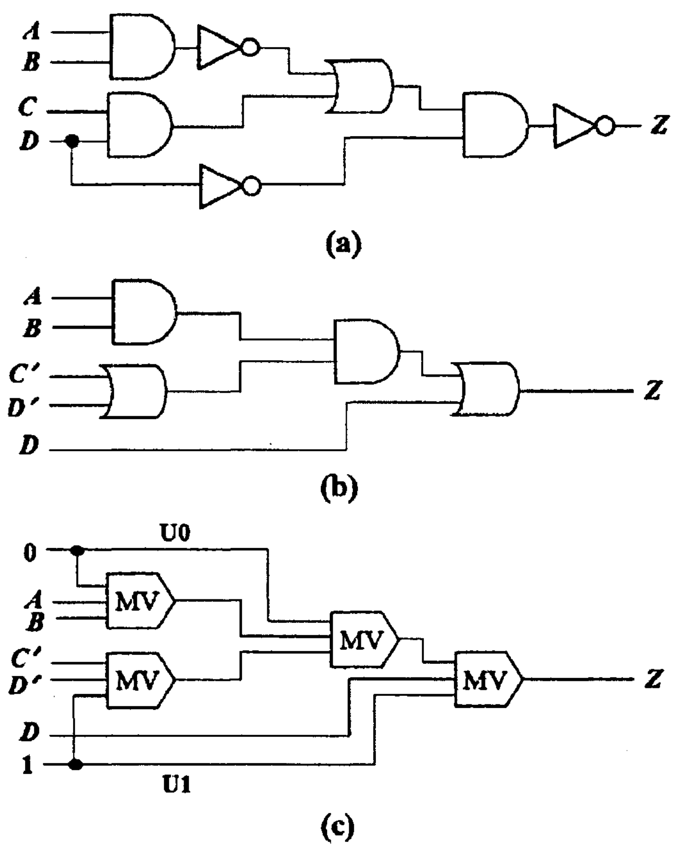

Figure 45: (a) A general arbitrary inverting logic, (b) The same logic implemented in AND-OR scheme, (c) The same logic implemented as AND-OR scheme in QCA with MVs.

6.9. Problem of Literal Input AND-OR Scheme

The apparently simple literal input AND-OR scheme becomes tricky when MV elements with non-primary inputs are to be tested in more complex designs. This is due to the fact that primary inputs are fed to an inverting block, and then necessary literals are fed to AND-OR configuration of MV elements. However, it is needed to provide all '1' or all '0' as inputs to AND-OR logic block, and not the primary inputs, which feed the inverting block instead of the AND-OR logic. This may not be possible in every case. The previously proposed scheme for the QCA logic implementation in [8] covers very elementary circuits, where only single polarity inputs are used, but fails for general logic where both primary inputs and their complements are required simultaneously. A simple example illustrates this problem, figure 46. Consider a function $f=AB+B'C$, where one can set all inputs A, B, C to '0' or '1', but never get both B and B' to be '0' or '1' simultaneously, resulting in failure of the testing scheme.

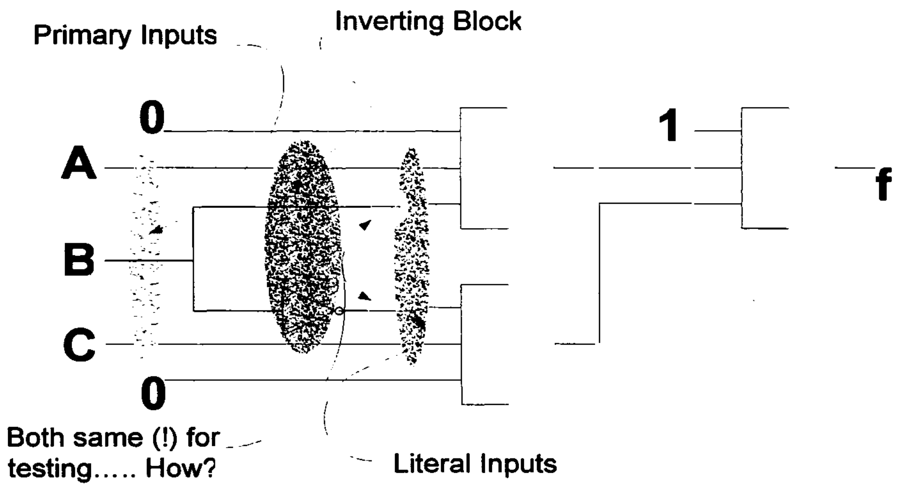

Figure 46: Problem with literal input AND-OR scheme in a simple circuit involving inverters, two of the literals cannot have the same value at the same time, as required for testing of the circuit.

6.10. The Proposed Scheme

To overcome this obstacle, a set of '*Test Enable*' MVs can be introduced after the inverting block, which can be manipulated either to pass the original literal fed from the inverting block (in the example, A, B, B' and C), or irrespective of the literals, can set its output to '0' or '1' as needed for the s-a-v fault testing, figure 47. Therefore, the proposed scheme not only reduces the test size but also actually becomes essential to test AND-OR logic.

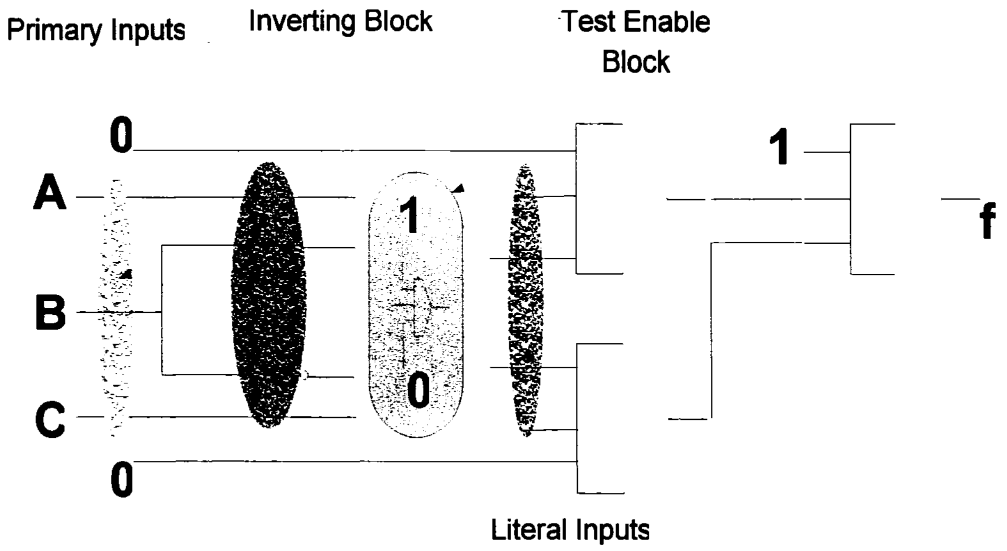

Figure 47: Problem solved with test enable block for the same simple circuit involving inverters, all the literals can now have the same value at the same time, as required for testing of the circuit.

Properly placed '*Test Enable*' MVs also deal with fault masking arising from modular expansion. When a number of circuit blocks are cascaded to build a complex circuit, then a fault at one stage may be undetected due to '*Test Enable*' MVs of the next stage. These impediments are overcome by an extra fault propagation path for fault masking due to cascading. It is to be noted that as all AND-OR logic gates changes into AND gates for s-a-0 testing and to OR gates for s-a-1 testing, and '*Test Enable*' MVs can generate all '0' or all '1' by only their control lines, so any arbitrary logic should be testable. Thus testing of such logic would also require only two test vectors, as illustrated in the following section.

Two design approaches are proposed in this thesis and illustrated on the example of a 1-bit full adder which is general enough to handle any combinational QCA circuit. A 4-bit full adder (cascades of four 1-bit full adders) is presented to explain the modular design.

6.10.1. Design 1: Shorter Test Vector

Here an adder circuit is implemented as a literal input AND-OR logic, with initial inverting block providing for the input literals. Hence, just the primary inputs and their complemented values are shown in figure 48.

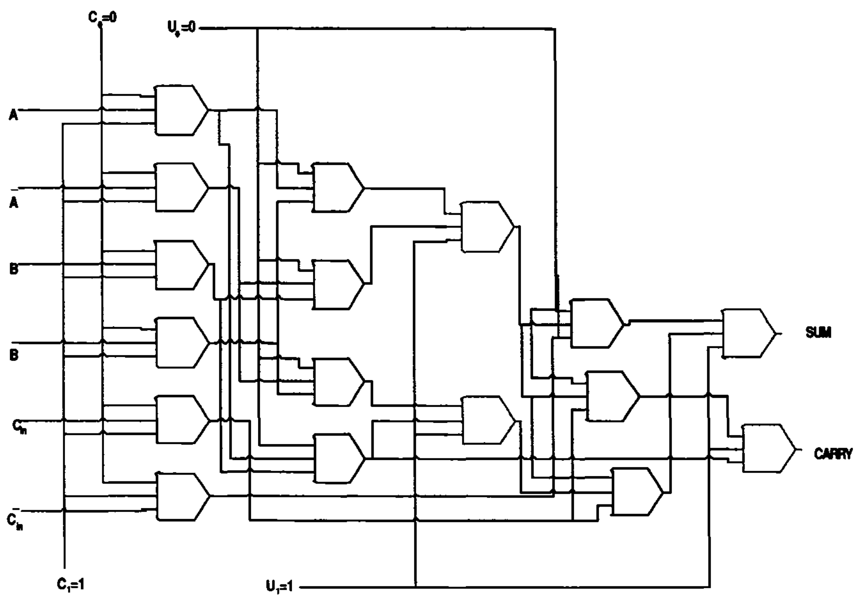

Figure 48: Design 1: A full adder is implemented with test enable block at the beginning of the non-inverting AND-OR block.

The leftmost column of MVs in figure 48 is the '*Test Enable*'. From this figure it can be observed that gate functions of MVs are controlled by two signals C_0 and C_1. When these control inputs are '0' and '1' in any order (i.e. C_0C_1= 01 or 10), the MVs will deliver actual primary input literals to the original non-inverting AND-OR logic. This is the case for normal operation. However, setting both control inputs (C_0, C_1) to '1' or '0' will feed all AND-OR logic inputs with '1' or '0' respectively. The circuit now operates in the *Test Mode*, thereby testing for s-a-1 and s-a-0 in QCA circuit data path. U_0 and U_1 are the control inputs to AND-OR MVs. They set the MVs functionality to AND or OR. Hence, two test vectors detecting data path s-a-v faults would be: $\{C0,C1,U0,U1\} = \{1100\}$ for s-a-0 and $\{C0,C1,U0,U1\} = \{0011\}$ for s-a-1.

Although, the test scheme is illustrated on the example of a full adder cell, in fact, it will always suffice to use the above two 4-bit vectors to test any combinational design regardless its complexity and number of primary inputs.

6.10.2. Design 2: Circuit with Fewer Extra Majority Voters

Another implementation is shown in figure 49 for the same literal input AND-OR logic of a full adder. However, now only inverted literals go through the *Test Enable* structure, figure 49. The number of extra MVs is reduced to the size of primary inputs, but the price played is in the increase of the vector length. The test set $\{C0,C1,U0,U1,A,B,Cin\}$ for s-a-0 and s-a-1 faults are: $\{1100111\}$ (s-a-0) and $\{0011000\}$ (s-a-1) - an increase from 4 (subsection 6.10.1) to 7 bits. Still, the circuit testing requires only two test vectors. Unlike in previous design, now vector size is not constant but increases with primary input number.

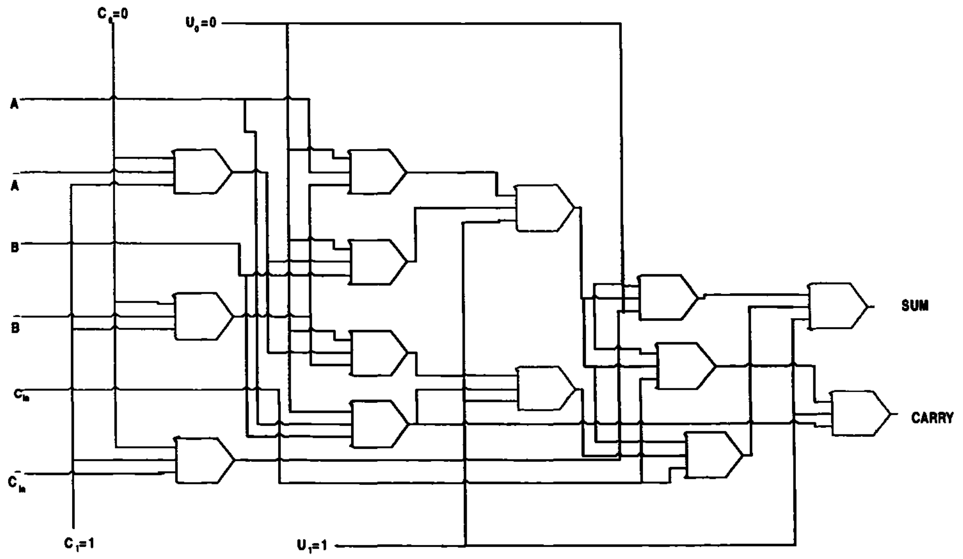

Figure 49: Design 2: A full adder is implemented with test enable block at the beginning of the non-inverting AND-OR block, but with MVs only in the inverted literals.

The two designs have their own advantages. The former has only two 4-bit test vectors for any circuit, while the latter has reduced number of extra majority voters.

6.10.3. Modular Design: Fault Masking Due to Cascading

In modular design, not all outputs of a block are primary ones; some end up in being inputs to next stages. In general, inputs coming from a previous stage could be inverted, however, both original and inverted signals may be required by the next stage. As testing the next stage for s-a-v faults under a scheme proposed in previous section would require both the inputs to be '1' or '0' at the same time, an additional '*Test Enable*' MV elements must be inserted between an output of one stage (the ones that acts as inputs to the next stage) and the corresponding input of next stage. Thus, during testing, one could generate the required literals to that particular stage. This may, however, result in masking of faults, which affects propagation to primary outputs through the path on which additional '*Test Enable*' of the next stage is inserted.

For example, consider a 4-bit adder. The $CARRY_i$ output of each intermediate i^{th} 1-bit adder stage is the input to the next stage, and only MSB CARRY4 is a primary output. When testing the circuit, '*Test Enable*' MVs of each bit (1st, 2nd, 3rd or 4th) masks previous CARRY signal coming into the MV. So if a s-a-v fault in second 1-bit adder propagates through CARRY2 line, it will be masked by 3rd bit '*Test Enable*' MV for Cin_3 ($\equiv CARRY_2$). To detect this fault, one can take fanouts from every CARRYi, $i=1,\ldots,4$, and connect them to extra consecutive MVs. So fanouts from $CARRY_1$ and $CARRY_2$ go to MV-1, its output and $CARRY_3$ go to MV-2, its output and $CARRY_4$ goes into MV-3 and so on. For s-a-0 testing, the MVs in the extra propagation line are configured as AND gates, while for s-a-1 they realize OR operation, thereby effectively propagating any such fault on CARRY path.

The proposed solution is expressed in the block diagram, figure 50. Primary inputs are fed to the inverting block, thereby making both true and complemented inputs available for '*Test Enable*' circuitry. Control inputs to the '*Test Enable*' block {*C*0,*C*1} determine the mode of operation ({01} or {10} for normal, {00} for s-a-1 and {11} for s-a-0). Control inputs *U*0 and *U*1 set MVs of non-inverting AND-OR logic block accordingly (either all ANDs or all ORs for test mode, or required combination of AND-OR for normal mode). For modular design, outputs of a stage going into next stage are fanned out to the extra fault propagation line as constructed by correctly configured MVs (AND for s-a-0, OR for s-a-1). This ensures fault propagation to the extra output, as shown on figure 50(b).

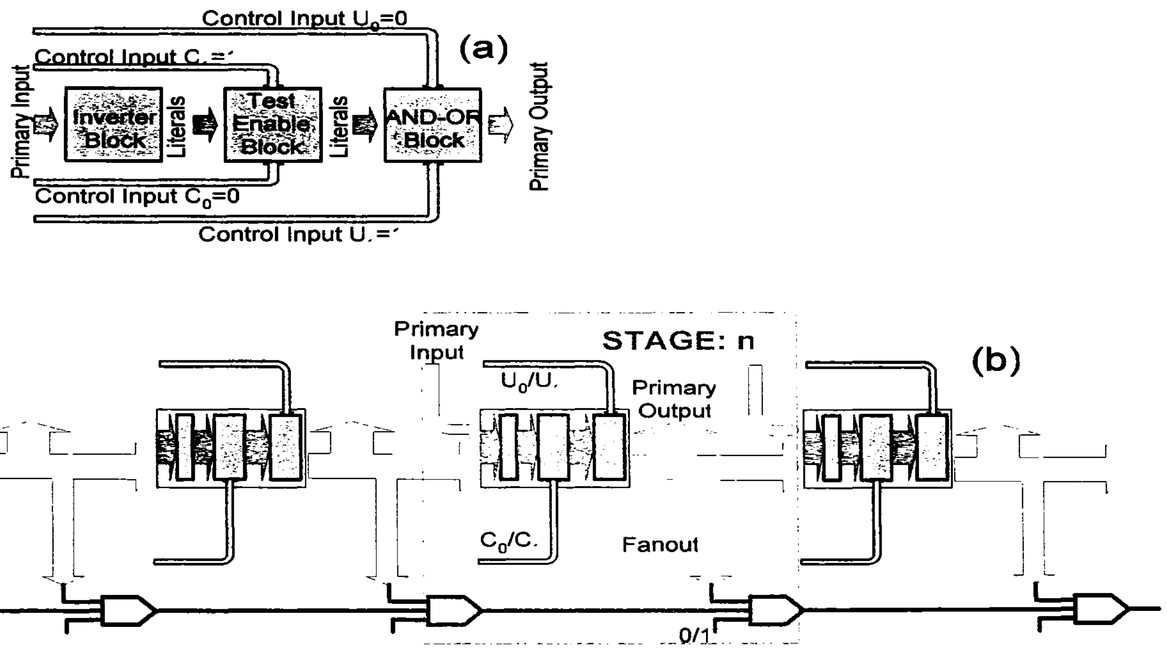

Figure 50: Block diagram for modular design for testability, (a) Block diagram of a single module, (b) Block diagram of modular design for n-bit with extra fault propagation line.

With this approach, automated test pattern generation (ATPG) is still needed for s-a-v faults on the control path. The unique testing features of QCA can be used in this case to reduce the size of test sets [12] for control line ATPG. Another observation may reduce test set for control line fault testing- in literal input AND-OR scheme, majority voters are always configured as AND or OR. Any fault in control line changes the functionality of the gate, and results in AND replaced by OR or vice versa. So control line s-a-v faults can be treated as functional gate replacement faults in data path. In that case other fault detection techniques such as satisfiability (SAT) may be used [70].

Chapter 7: Simulation Results

All the methods proposed in this thesis were verified through simulations. The *Quantum Dot* technology is very new, and not yet used commercially on a large scale. Not only the behavior of QCA logic is still not fully understood, but also lack of variety of competitive CAD tools makes it impossible to select the most appropriate working environment to simulate QCA circuits. The only, widely available on the market, tool is QCADesigner 2.0.3 software, and this is what we used to conduct our experiments.

Design specifications were as follows: cell dimension of 18nm x 18nm, dot size 5nm, radius of effect 65nm, layer separation 11.5nm. Multilevel binary wire connectivity was used to avoid possible errors arising from same level crossovers between binary wire and inverter chains. Bistable approximation and exhaustive verification was chosen with 512000 samples per run.

The defect characterization for a majority voter has been realized to verify the defects due to cell displacement, cell misalignment and cell omission. As stated earlier not all cell omission defects cause a change in circuit functionality. The characterization done within this work reestablishes those found in previous study, and has already been stated in chapter 4. So only the s-a-v fault testing is given herewith.

During simulation, the circuit was first tested for its functionality and then *s-a-v* faults were injected at various points. In every case, the proposed scheme of using *Test*

Enable block worked as expected, and the predicted test set detected all the s-a-v faults. A 1-bit full adder with literal inputs AND-OR configuration has been simulated first. It is observed that the proposed two test vectors can not be applied because it is not possible to set the same value to both true and complement of an input. As to our knowledge, the AND-OR implementation of a 1-bit full adder in QCA has not been implemented before, hence this problem was never detected. Therefore the design schemes with *Test Enable* block proposed here were implemented for the same full adder to overcome the problem.

First we verified the correctness of operation of a fault-free design using the exhaustive test set proposed. Next, different faults were injected at selected places of the circuit and the faulty behavior was compared with the correct one. Due to the lack of comprehensive test simulator, we did not use a complete fault list, but only its subset. In every case, the presence of a fault was detected by the proposed test set. For clarity purpose, only fixed polarization of individual data path cells was considered (not considered was control line connectivity to external bias application).

For complex circuits we used a hierarchical design approach. First we generated a small block, and then built up the whole design using hierarchical connection of smaller, previously verified elements. Testing of such circuits, however, is more Byzantine because inputs to all of the blocks are not transparent for setting a test vector. As a result fault masking may happen. The fault masking problem arising from the cascading of 1-bit full adder blocks was verified by connecting varied number of blocks.

7.1. Design 1:

Figure 51 shows the layout of the 1-bit full adder with test enable block design scheme1. The clock zones are selected such that the crossover lines remain at the same zone. As a longer wire may cause wrong polarization, it is divided into two clock zones. The simulation result for the circuit operating in normal mode is presented in figure 52.

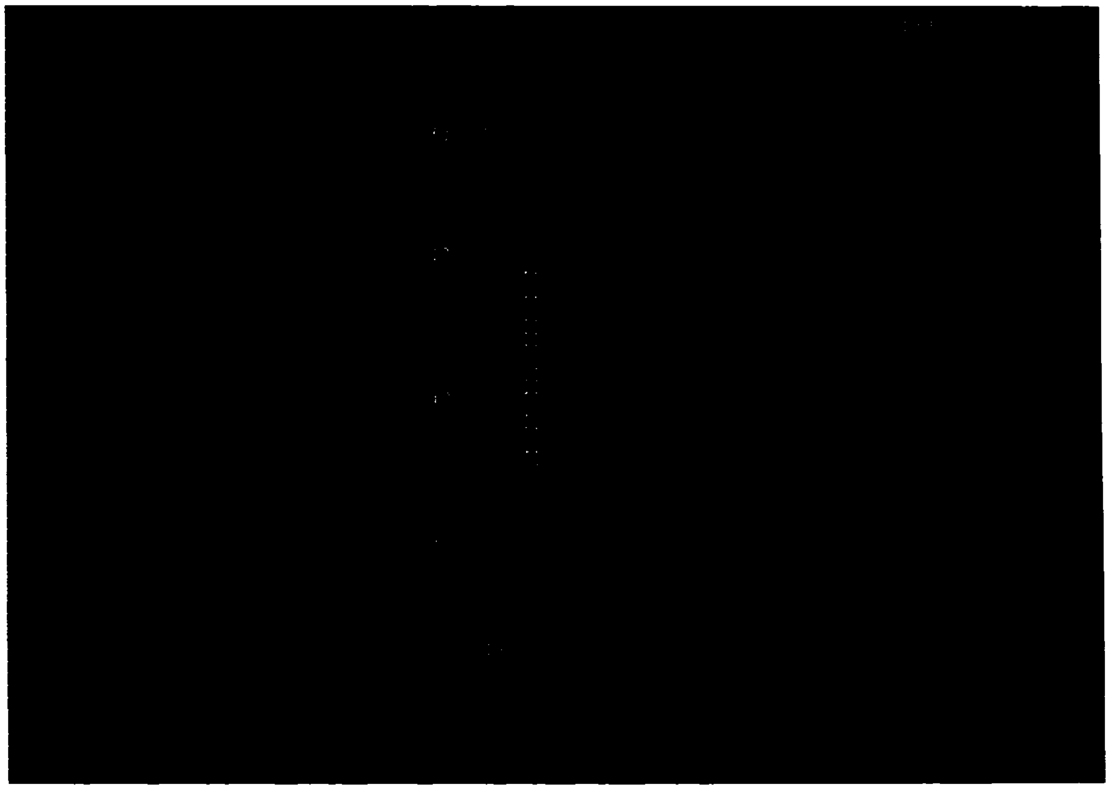

Figure 51: 1-Bit full adder with 'Test Enable' (Design scheme 1)

In test mode aiming for s-a-0 fault detection, both '*Test Enable*' and AND-OR block control lines are set to {*C0*, *C1*, *U0*, *U1*} = {1100}, which is the only test vector to test all s-a-0 faults in the data path. The setting {*U0*, *U1*} = {00} configures all the majority voters into AND gates. As both control lines of the '*Test Enable*' block are '1' both the *sum* and *carry* outputs become '1' irrespective of the data inputs. In the data path of the circuit, faults at different location are injected, and *sum* or *carry* signals or even both gave the output '0' depending on the propagation path of the fault. Figure 53 shows the output in test mode without any fault injection, and figure 54 shows the testing results of a fault injected at the output of XNOR gate. The fault follows the path toward SUM only, so the SUM output is '0'. Thus the fault s-a-0 is detected.

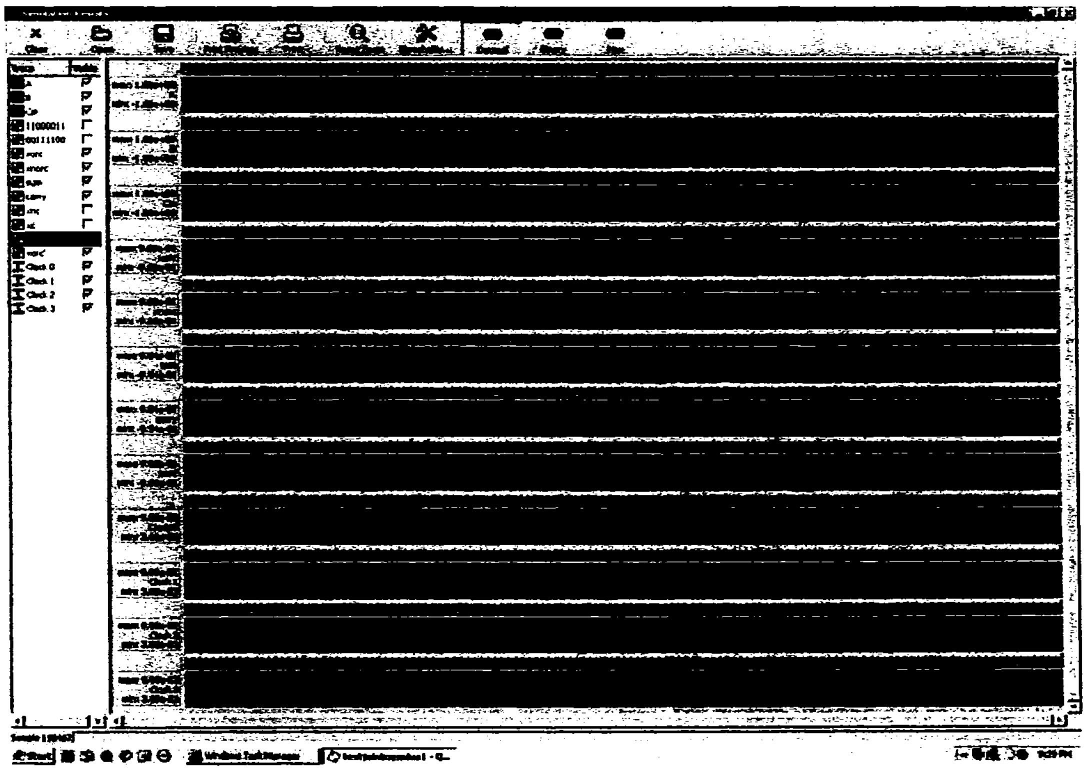

Figure 52: Design 1 normal operation: simulation results

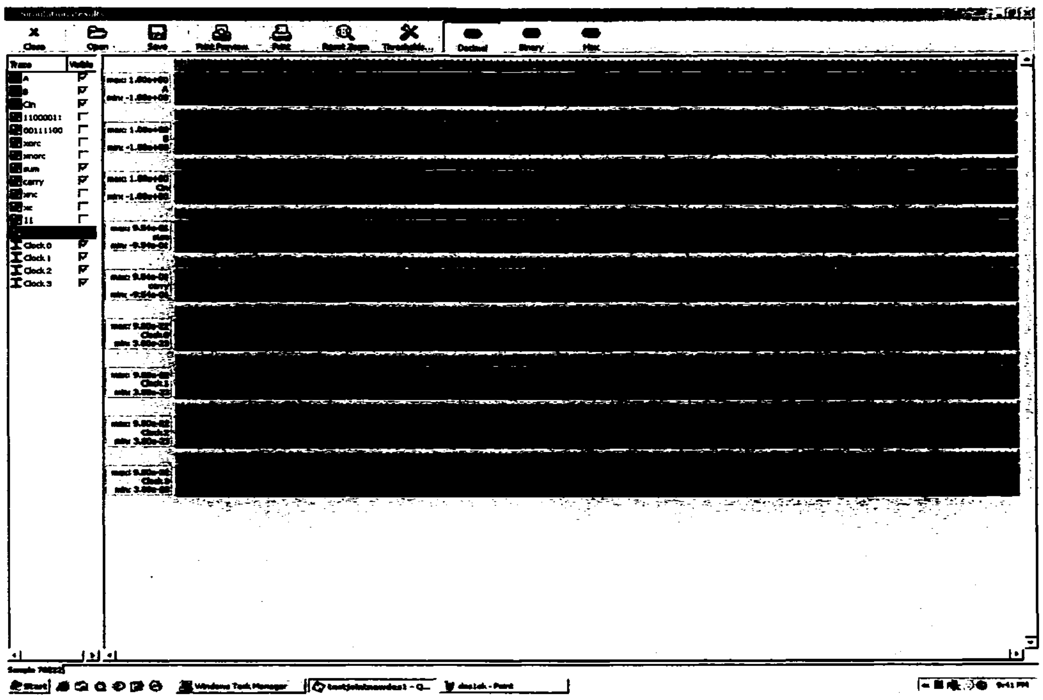

Figure 53: Design 1 s-a-0 fault test detecting no faults for a fault free case: simulation results.

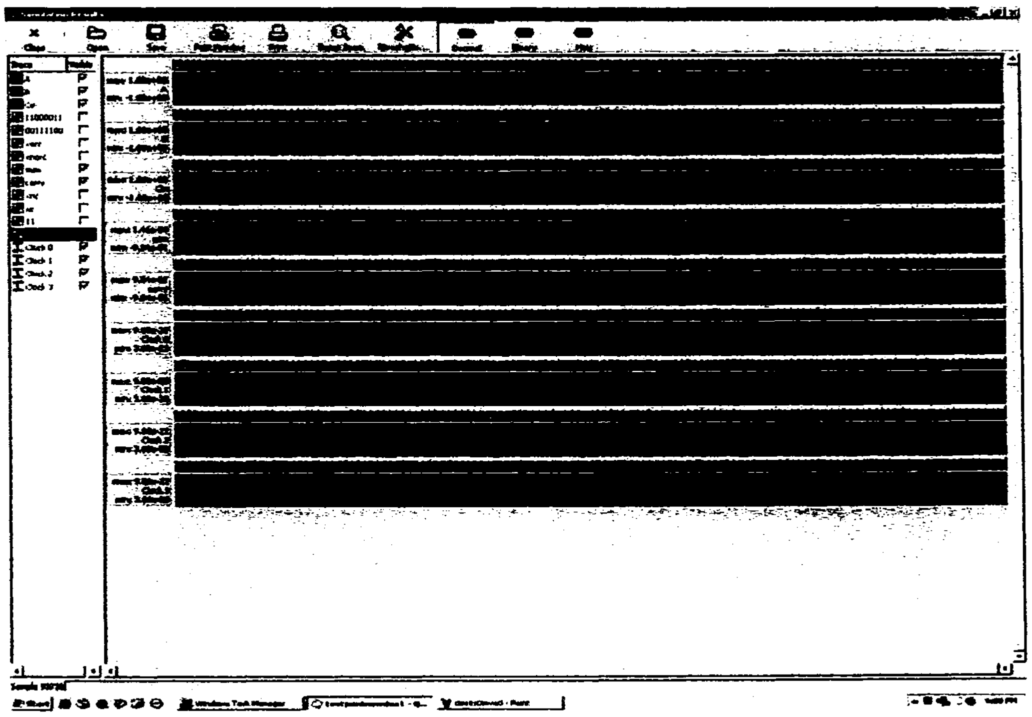

Figure 54: Design 1 fault detected in SUM for s-a-0 in front of XNOR operation for bit-1: simulation results.

Similarly, to test s-a-1 fault, the control lines were excited with test vector {*C*0, *C1, U0, U1*} = {0011}. For {*U*0, *U1*} = {11} the majority voters are configured as OR gates. As *C*0 and *C*1 are '0', all the OR gates are fed with '0', in consequence, *sum* and *carry* bits both result in '0'. The presence of any s-a-1 fault gives the improper output again either through sum or carry or both (depending upon fault propagation cone). Figure 55 shows the correct operation while figure 56 shows the effect of the fault injected at *AB* line, which propagates through both outputs.

It took 9 clock zones for data path from input to output for 1 bit, i.e., output is available after two clock cycles.

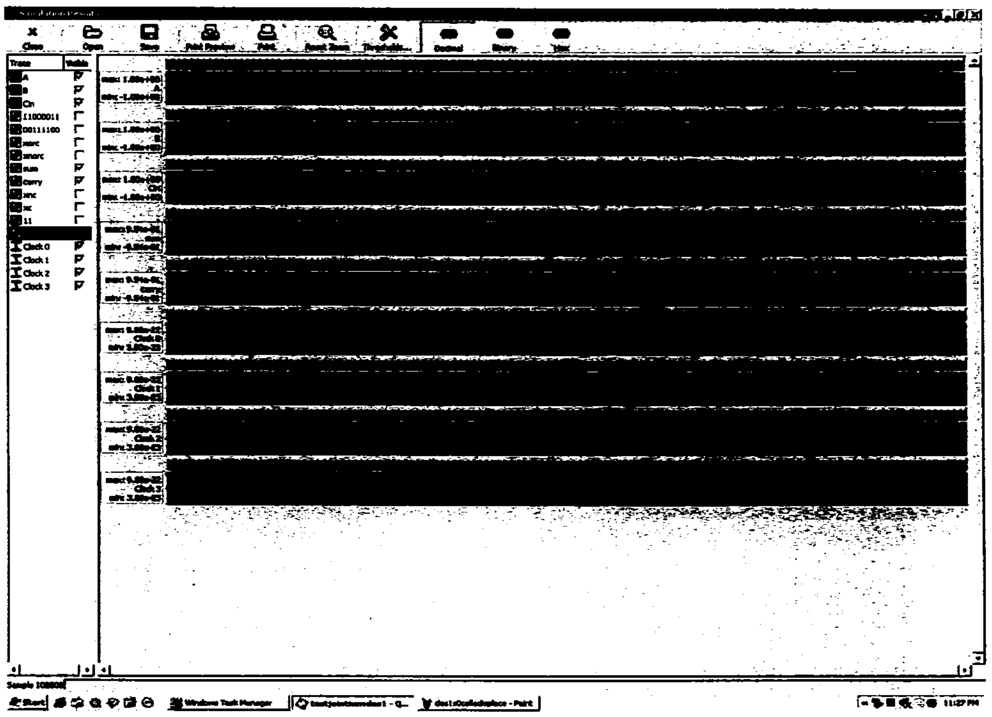

Figure 55: Design 1 s-a-1 fault test detecting no faults for a fault free case: simulation results.

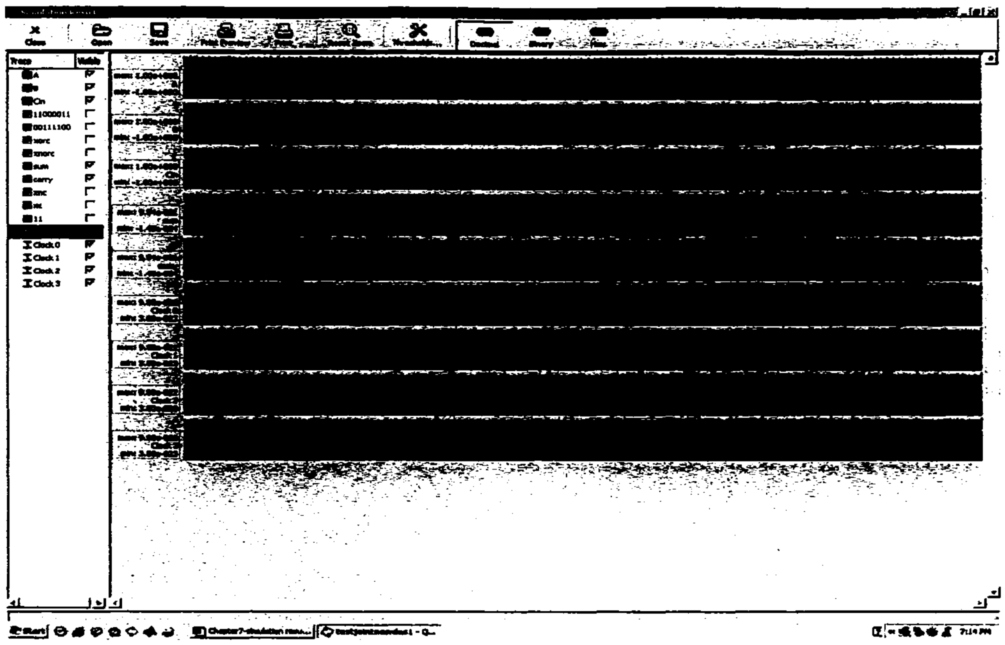

Figure 56: Design 1 s-a-1 fault detected in both SUM and CARRY for a fault injection in front of AB operation of bit-1: simulation results.

7.2. Design 2:

The layout for design scheme 2 is presented in figure 57. The circuit is verified first for correctness (normal mode of operation). The output is same as in figure 52. Then the design is set to test mode with test vectors for fault free condition, and then s-a-0 and s-a-1 faults at random data path nodes are inserted, as in the case of the design scheme1.

To test s-a-0 faults the $\{C_0C_1U_0U_1ABC_{in}\} = \{1100111\}$ vector is applied. In this case U_0U_1= 00 sets all the majority voters as AND gates, and the '*Test Enable*' block sets all the inputs to AND-OR block to 1. As before the signal took 9 clock zones to reach from input to output of the adder circuit. As a propagation delay of 2 clock cycles is present in the output, the first two outputs should be ignored. Therefore cycles 3 to 10 only are considered. In the considered case, only the 8th input cycle ensured the proper input setting as $\{ABC_{in}\} = \{111\}$, so the 10th output gave the correct result of '1' for a fault free case. This is a direct consequence of using the exhaustive simulation instead of vector table simulation. The exhaustive simulation gives a complete picture for all the possible cases, while in practice, it is possible to generate an input of $\{ABC_{in}\} = \{111\}$ in the first input cycle, thereby keeping the test time minimal.

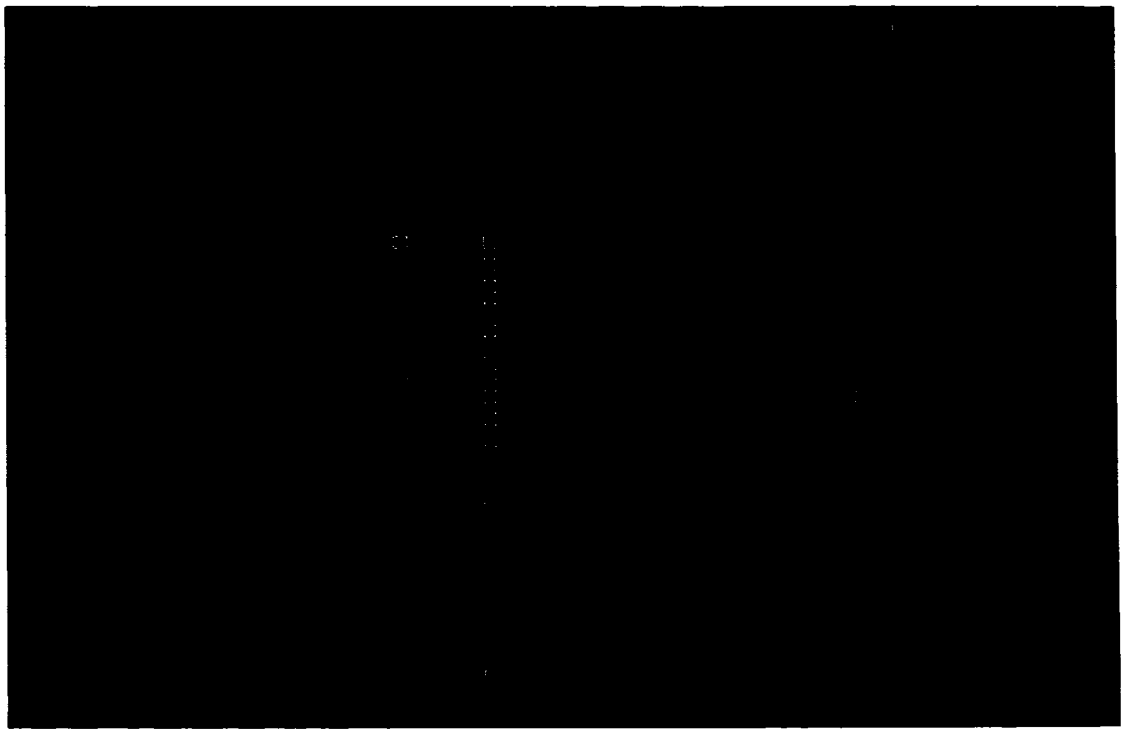

Figure 57: 1-Bit full adder with 'Test Enable' block in Design scheme 2.

Random s-a-0 faults at different location resulted in erroneous '0' at the output. Figure 58 shows the correct output, and figure 59 presents the erroneous carry output due to a fault occurred in the line representing XOR.C and propagated to the carry only.

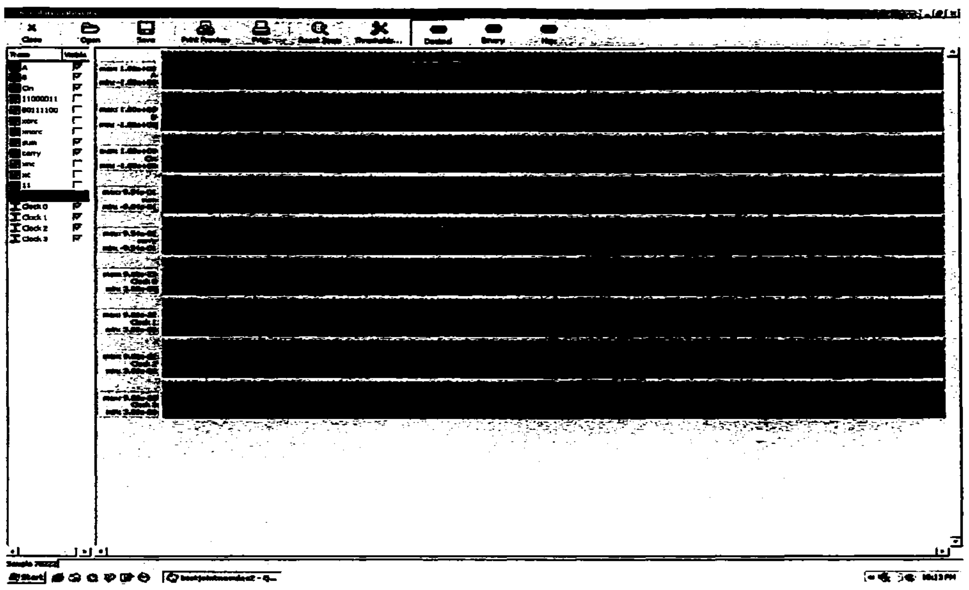

Figure 58: Design 2 test for s-a-0 fault detecting no fault for a fault free case: simulation results.

In the case of detecting s-a-1 the vector is {*C*0, *C1, U0, U1, A, B, Cin*} = {0011000}. This vector sets all the majority voters to act as an OR function. {*C*0, *C*1}={00} sets the complemented inputs to the OR block and {*A, B, Cin*} = {000} results in'0' in the 3rd and11th output cycle for fault free circuit of figure 60. Any s-a-1 fault at an arbitrary place of the data path gives erroneous result '0'. In figure 61 the output of an AND gate feeding inputs *A'B'* is contaminated with s-a-1 fault. The fault is propagated through the *sum* output only, and as a result the *sum* is incorrect '1' instead of '0'. So the fault is detected in this case.

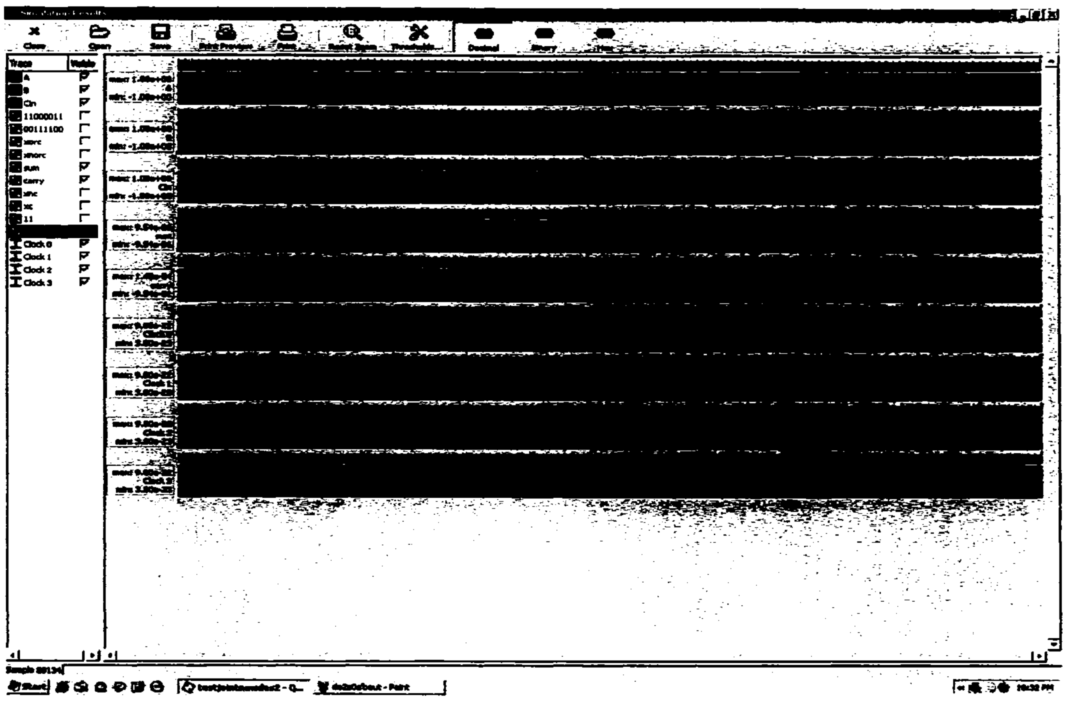

Figure 59: design 2 test for s-a-0 fault detected in CARRY for a fault site in front of XORC operation for bit-1: simulation results.

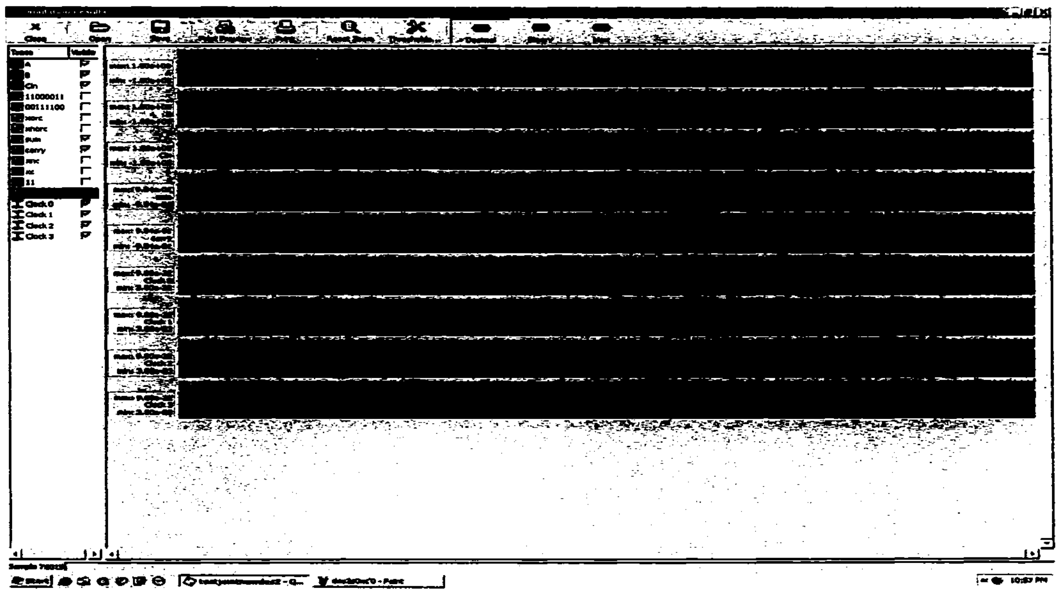

Figure 60: Design 2 test for s-a-1 fault detecting no fault for a fault free case in bit-1: simulation results.

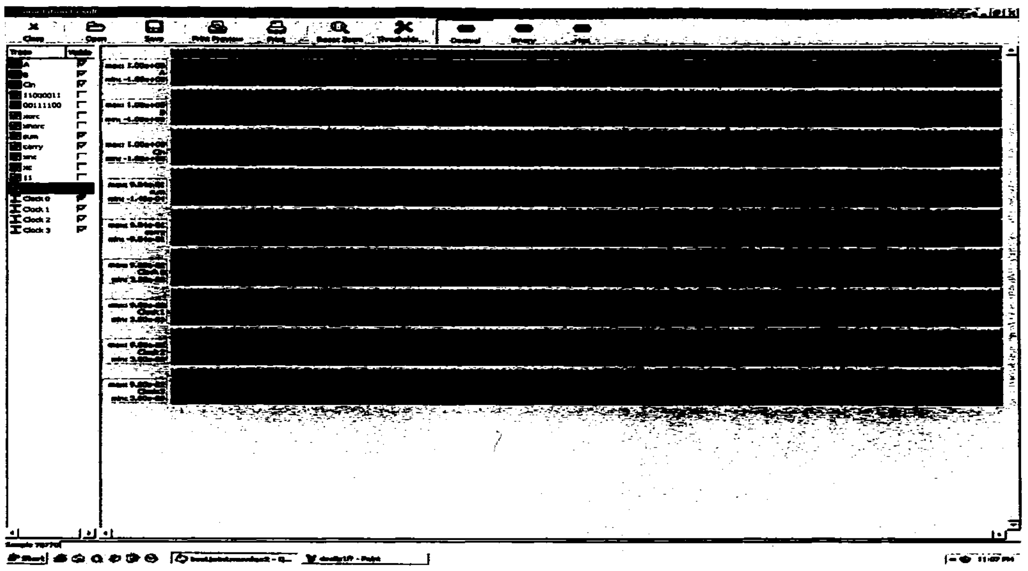

Figure 61: Design 2 test for s-a-1 detecting fault in SUM for a fault site in front of A'B' operation in bit-1.

7.3. Modular Design:

Cascaded full adders of varied sizes have been simulated to verify the proposed modular design. For the simplicity, results for 2-bit adder (figure 62) are discussed. Figure 63 shows the outputs with setting C_{in}= 0. The *sum* and *carry* outputs for each stage are presented individually to observe the fault effects. Like in the case of design1 s-a-1 and s-a-0 faults are detected in this modular approach with two test vectors. The carry outputs from both stages are AND-ed setting the output to '1' when CTEST is '1'. The other outputs remain the same as in normal operation mode. To test the s-a-0 the same test vector as for design1 {*C0*, *C1*, *U0*, *U1*} = {1100} is applied to the modular design. All randomly injected faults in every block are detected. To detect any s-a-0 fault on carry output of first stage the output CTEST is observed. As in normal operation one of the CTEST majority voter inputs are set to '0' for AND operation, the s-a-0 fault on carry will make output '0' instead of correct output '1'. So the fault is propagated through this additional MV. Figure 64 shows outputs in test mode fault for free operation, while figure 65 shows the faulty output.

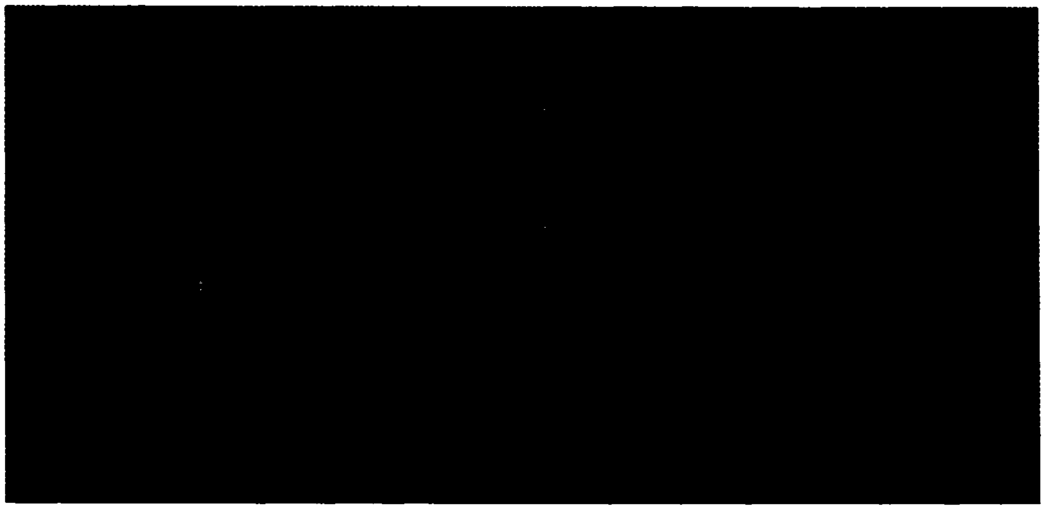

Figure 62: 2-bit adder modular design for scheme 1.

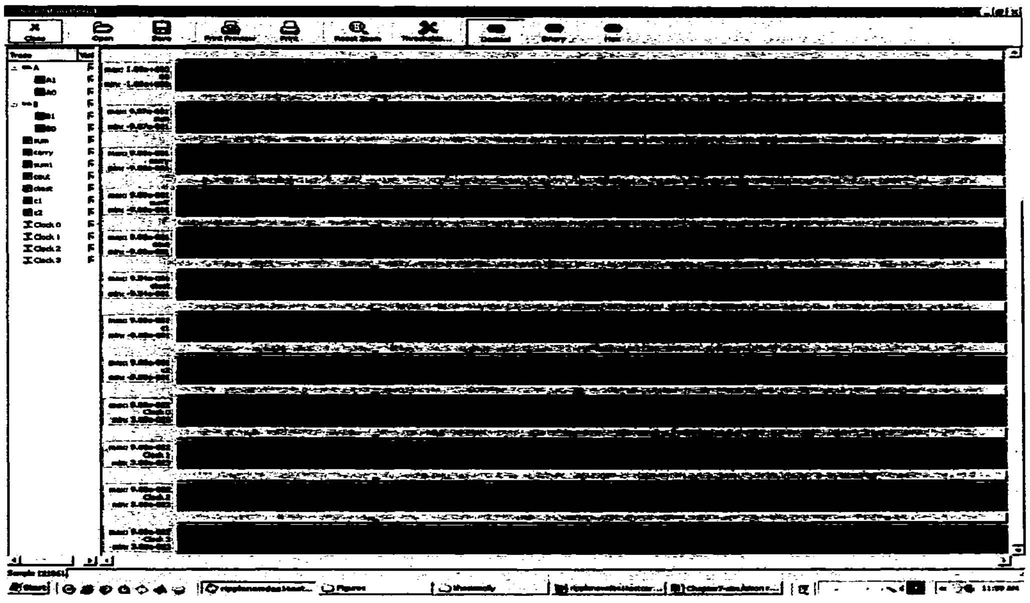

Figure 63: 2-bit adder with CTEST to detect masked fault in $CARRY_1$ output: normal operation.

In similar way, the s-a-1 fault in data path of each stage can be detected with test vector {*C*0, *C1, U0, U1*} = {0011} and setting the CTEST MV as an OR gate. For {*U0, U1*} = {11} all the MVs are configured as an OR gate, and the outputs are'0' in fault free case. Now, if a s-a-1 fault happens to carry an output of first stage, then this fault will be propagated through CTEST, and an erroneous'1'' is observed. Figure 66 and figure 67 reveal fault free and s-a-1 faulty circuits' operations.

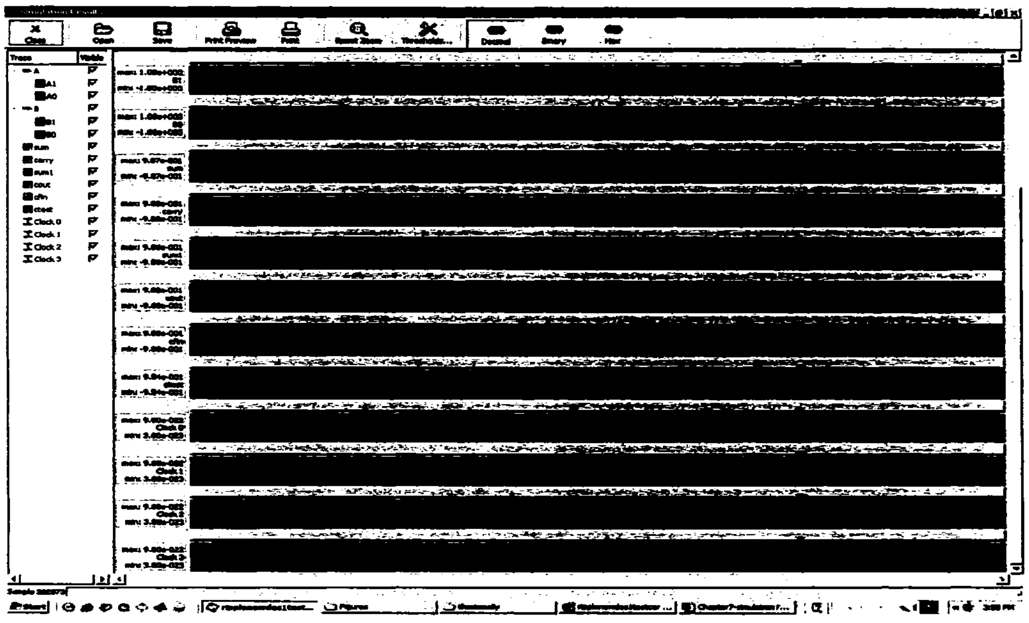

Figure 64: 2-bit modular design fault testing for s-a-0 detecting no fault for a fault free case.

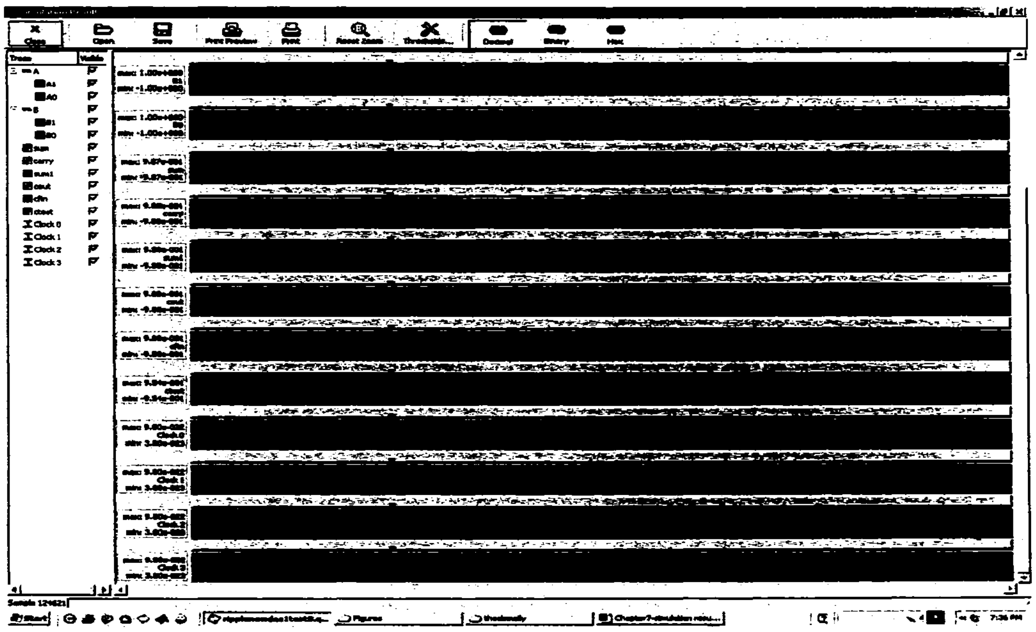

Figure 65 : 2-bit modular design test for s-a-0 detecting fault in CTEST line.

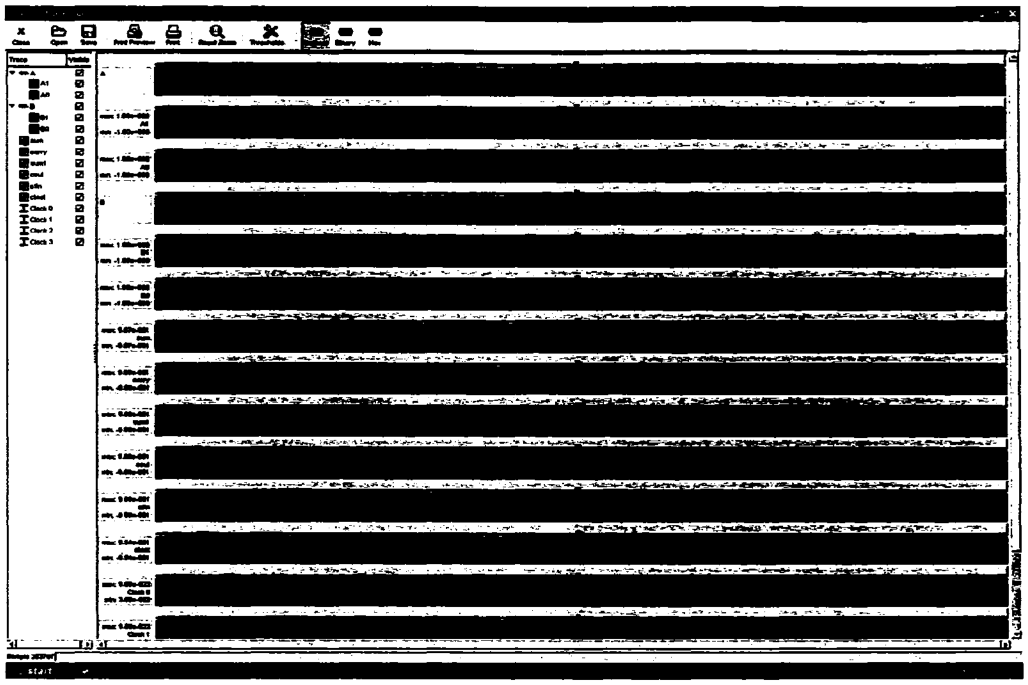

Figure 66: 2-bit modular design fault testing for s-a-1 detecting no fault for a fault free case.

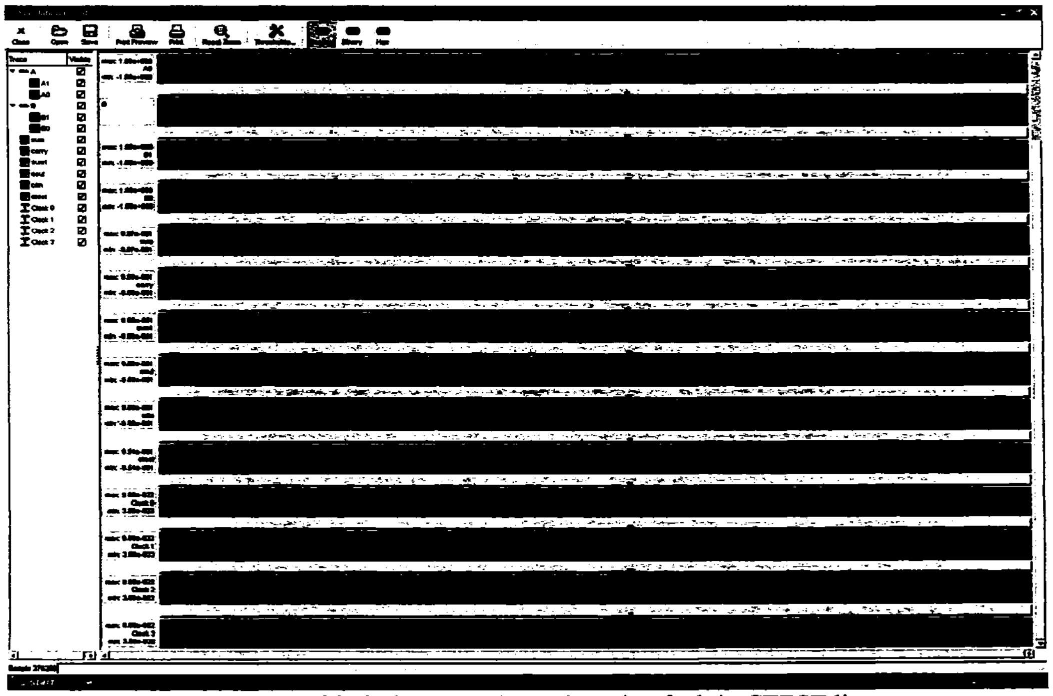

Figure 67: 2-bit design 1 s-a-1 test detecting fault in CTEST line.

Chapter 8: Conclusion and Future Work

Quantum Dot Cellular Automata (QCA) is a novel paradigm of computing, which has been proposed as a replacement for CMOS circuits. This thesis presents the introductory theory of QCA, its structure, working, implementation, testing, unique features, AND-OR logic, simulation of defects resulting in functional errors and hence application of fault model. The existing situation has been analyzed, and some lacks in testing aspects have been pointed out. Afterwards, in this work a new strategy for designing QCA logic exhaustively testable for s-a-v fault is presented. The first design uses $2n$ ($n \equiv$ primary inputs) '*Test Enable*' MVs, and is tested with two 4-bit vectors regardless of complexity of design and input size. Second design employs n majority voters for the same purpose, resulting in lesser MVs at the price of increased vector length. Application specific conditions would decide which design becomes optimal.

The use of ATPG for s-a-v faults at control inputs of the non-inverting circuit remains simple. The data path frequently becomes very complex; nevertheless the proposed technique uses only two short lengthened test vectors.

8.1. Issues in Future QCA Design

As a final concluding remark, some issues that have been addressed or can be considered in future works regarding QCA design for testability is succumbed here.

Application of single-stuck-at-value fault model: Though individual occurring defects may not act like any single s-a-v fault, test set for this fault model could still be able to attain high fault coverage for any such circuits. Indication is also there that a combinational circuit does not change into a sequential one, so this fault model should suffice for testing.

Fault detection definition: In conventional s-a-v model application, it is said that a fault is detected if at least one output is complemented from its fault free value (i.e. we get a '1' instead of '0' or vice versa). But as seen in some cases, the output may not carry any logic value, a proposition is made to change the definition for fault detection to: Fault is detected if one fails to detect the correct response (i.e. instead of detecting a fault by observing a '1' instead of '0', a fault is said to be detected as the fault free response '0' has not been obtained).

Delay fault application: Delay testing may be applied to find defects that do not change the functionality but do degrade circuit performance by incurring delay.

Literal input AND-OR logic: It is beneficial to employ QCA design using the literal input AND-OR logic approach for reduced test set. In applying the method, the proposed two '*Test Enable*' schemes can be adopted, the better of which, may depend on distinct case constraints.

ATPG tool: ATPG tool for testing faults in the control inputs using the unique features of QCA logic should give excellent result. Satisfiability (SAT) is a good alternative assuming the gate replacement faults in control lines.

Standardization of implementation: Different implementation can be used for the same logic, resulting in different test set. This might become a problem in large scale industrial basis, so further research is needed to hint some way of standardizing basic logic blocks and bringing the industry to a consensus for a particular design adoption for at least the basic logic blocks.

8.2. Future Work

The proposed solutions are mainly concentrated on data path faults testing. A versatile scheme is required for testing of control lines. As the faults in control lines exhibit gate replacement behavior, in this testing approach a standard ATPG base or SAT [70] based formulation is planned for future work.

Area minimization is also an important task. Exploiting the scaling issues presented in [59] the spacing between wires can be obtained, which will create an area efficient design. This will also reduce the propagation delay in QCA circuit.

In this work, only combinational circuits are considered. In future, the effectiveness of the proposed scheme to test sequential circuits in QCA will be investigated. At this stage no or very few work has been done in this regard. The only study on defects characterization of QCA sequential circuits [57] shows that molecular-based model of this type of devices and circuits includes single additional and missing cell defects, and they are mostly evidenced at logic level by stuck-at faults. The effectiveness of s-a-v fault test sets can be further investigated.

Finally, the implementation of standard quantum gates like the NOT, the controlled NOT and a three-bit quantum gate using QCA have been demonstrated in [4]. The fault model of these gates remained under investigation, and needs attention in future to pave way for quantum computing.

Reference

[1]. J. D. Plummer, M. Deal, P. B. Griffin, "Silicon VLSI Technology: Fundamentals, Practice and Modeling"; Prentice Hall, Inc. 2000.

[2]. M. B. Tahoori, M. Momenzadeh, J. Huang, F. Lombardi, "Defects and Faults in Quantum Cellular Automata at Nano Scale``; Proceedings of the 22nd IEEE VLSI Test Symposium (VTS 2004), 2004.

[3]. C. S. Lent, P. D. Tougaw, W. Porod, G. H. Bernstein, "Quantum cellular automata"; Nanotechnology, Vol.4, pp.49-57, 1993.

[4]. G. Toth, J. Timler, C. S. Lent, "Quantum Computing with Quantum-dot Cellular Automata using Coherence Vector Formalism"; Proceedings of the IEEE International Workshop on Computational Electronics (IWCE-6), Osaka, 1998.

[5]. C.S. Lent, B. Isaksen, M. Lieberman, "Molecular Quantum-Dot Cellular Automata"; Journal of the American Chemical Society, vol. 125(4), pp. 1056-1063, 2003.

[6]. H. Qi, S. Sharma, Z. Li, G.L. Snider, A.O. Orlov, C.S. Lent, T. P. Fehiner, "Molecular Quantum Cellular Automata Cells: Electric Field Driven Switching of a Silicon Surface Bound Array of Vertically Oriented Two-Dot Molecular QCA"; Journal of the Am. Chem. Society, (JACS Articles) vol. 125, no. 49, pp. 15250-15259, 2003.

[7]. J. Huang, M. Momenzadeh, M. B. Tahoori, F. Lombardi, "Defect Characterization for Scaling of QCA Devices"; Proceedings of the 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'04), 2004.

[8]. M. B. Tahoori, J. Huang, M. Momenzadeh, F. Lombardi, "Testing of Quantum Cellular Automata"; IEEE Transactions on Nanotechnology, Vol.3, No.4, December 2004.

[9]. P. D. Tougaw, C. S. Lent, "Logical devices implemented using quantum cellular automata"; Journal of Applied Physics, Vol.75, No.3, pp.1818-1824, February 1994.

[10]. E. McClusky, C. Tseng, "Stuck-fault tests vs. actual defects"; International Test Conference, pp.336-343, 2000.

[11]. A. Fijany, B. N. Toomarian, "New Design for Quantum Dots Cellular Automata to Obtain Fault Tolerant Logic Gates"; Journal of Nanoparticles Research, Vol.3, pp.27-37, February 2001.

[12]. M. B. Tahoori, F. Lombardi, "Testing of Quantum Dot Cellular Automata Based Designs"; Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'04) 2004.

[13]. http://www.qcadesigner.ca

[14]. G. H. Bernstein, "Quantum-dot Cellular Automata by Electric and Magnetic Field Coupling": Custom Integrated Circuits Conference, IEEE, 2003.

[15]. A. Aviram, M. A. Ratner, "Molecular Rectifiers"; Chemical Physics Letters, Vol.29, No.2, pp.277-283, 1974.

[16]. T. P. Smith' "Quantum dots: electrons in a new dimension": SPIE Vol.1284 Nanostructures and microstructure Correlation with Physical Properties of Semiconductors, pp.12-19, 1990.

[17]. D. K. Ferry, W. Porod, "Interconnections and architecture for ensembles of microstructures"; Superlattices and Microstructures, Vol.2, No.1, pp.41-44, 1986.

[18]. T. A. Fulton, G. 1. Dolan . "Observation of single-electron charging effects in small tunnel junctions,''; Physical Review Letters, Vol.59, No.1, pp.109-112, 1987.

[19]. G. Bezan, A. O. Orlov, G. L. Snider, G. H. Bernstein, "Charge detector realization for AlGaAs/GaAs quantum-dot cellular automata"; J. Voc. Sci. Technol. B, Vol.14, No.6, pp.4046-4050, 1996.

[20]. C. S. Lent and P. D. Tougaw, "A Device Architecture for Computing with Quantum Dots"; Proceedings of the IEEE, vol.85 no.4 April 1997.

[21]. T. Tanamoto, R. Katoh, Y. Naruse, "A novel quantum cellular automata logic with loop structures"; Journal of Applied Physic, Vol.33, pp.1502–1505, October 1994

[22]. J. C. Lusth, D. J. Jackson, "Graph theoretic approach to quantum cellular design and analysis", Journal of Applied Physics, Vol.79, No.4, pp.2097, February 1996.

[23]. M. Chen, W. Porod, "Design of gate-confined quantum dot structures in the few-electron regime"; Journal of Applied Physics, Vol.78, pp.1050–1057, 1995.

[24]. T. Fountain, unpublished.

[25]. C. S. Lent, B. Isaksen, "Clocked Molecular Quantum-Dot Cellular Automata"; IEEE Transactions on Electron Devices, Vol.50, No.9, September 2003.

[26]. G. H. Bernstein, "Quantum-dot Cellular Automata: Computing by Field Polarization"; Custom Integrated Circuits Conference, Proceedings of the IEEE 2003, pp. 223-229, 21-24 September 2003.

[27]. G. L. Snider, A. O. Orlov, I. Amlani, G. H. Bernsteing, C. S. Lent, J. L. Merz, W. Porod, "Quantum-Dot Cellular Automata"; Microprocesses and Nanotechnology Conference, 1999. 1999 International Digest of Papers. Microprocesses and Nanotechnology '99. pp.90-91, July 1999.

[28]. K. K. Yadavalli, A. O. Orlov, R. K. Kummamuru, C. S. Lent, G. H. Bernstein, G. L. Snider, "Fanout in Quantum Dot Cellular Automata"; 63rd Device Research Conference, Santa Barbara, CA, June 2005.

[29]. M. Lui, C. S. Lent, "High-speed metallic quantum-dot cellular automata"; Third IEEE Conference on Nanotechnology. IEEE-NANO 2003. Vol.2, pp. 465-468, 12-14 August 2003.

[30]. L. Bonci, M. Gattobigio, G. lannaccone, M. Macucci, "Simulation of time evolution of clocked and nonclocked quantum cellular automaton circuits"; Journal of Applied Physics, Vol.92, pp.3169, 2002.

[31]. D. A. Antonelli, A. B. Kahang, D. Z. Chen, P. M. Kogge, T. J. Dysart, R. C. Murphy, X. S. Hu, M. T. Niemier, "Quantum-Dot Cellular Automata (QCA)

Circuit Partitioning: Problem Modeling and Solutions"; Proc.ACM/IEEE Design Automation Conf., June 2004, 363-368, June, 2004.

[32]. J. Liang, J. C. Lusth, "3-Dimensional Configuration to Promote Timely Settling of Quantum-dot Cellular Automata"; Proceedings of the Third IEEE Conference on Nanotechnology, San Francisco, CA, August 2003.

[33]. S. E. Frost, A. F. Rodrigues, C. A. Giefer, Peter M. Kogge, "Bouncing Threads: Merging a New Execution Model into a Nanotechnology Memory"; Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'03), 2003.

[34]. E. P. Blair, C. S. Lent, "An Architecture for Molecular Computing using Quantum-Dot Cellular Automata"; Third IEEE Conference on Nanotechnology, 2003. IEEE-NANO 2003, Vol.1, pp.402-405, 12-14 August 2003.

[35]. K. Walus, G. A. Jullien, V. S. Dimitrov, "Computer Arithmetic Structures for Quantum Cellular Automata"; Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, 2003, Vol.2, pp.1435-1439, 9-12 November 2003.

[36]. R. Zhang, K. Walus, W. Wang, G. A. Jullien, "Performance Comparison of Quantum-dot Cellular Automata Adders"; IEEE International Symposium on Circuits and Systems, Vol.3, pp. 2522-2526, 23-26 May 2005.

[37]. W. Wang, K. Walus, G. A. Jullien, "Quantum-Dot Cellular Automata Adders"; Third IEEE Conference on Nanotechnology, IEEE-NANO 2003, Vol.1, pp.461-464, 12-14 August 2003.

[38]. H. Cho, E. E. Swartzlander, Jr., "Pipelined Carry Lookahead Adder Design in Quantum-dot Cellular Automata"; Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers, 2005, pp. 1191-1195, October 28 - November 1, 2005.

[39]. K. Walus, G. Schulhof, G. A. Jullien, R. Zhang W. Wang, "Circuit Design Based on Majority Gates for Applications with Quantum-Dot Cellular Automata"; Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004, Vol.2, pp. 1354-1357, 7-10 November 2004.

[40]. R. Zhang, K. Walus, W. Wang, G. A. Jullien, "A Majority Reduction Technique for Adder Structures in Quantum-dot Cellular"; Applications of Digital Image

Processing XXVII. Edited by Tescher, Andrew G. Proceedings of the SPIE, Vol.5559, pp. 91-100, 2004.

[41]. J. R. Janulis, P. D. Tougaw, S. C. Henderson, E. W. Johnson, "Serial Bit-Stream Analysis Using Quantum-Dot Cellular Automata"; IEEE Transactions on Nanotechnology, Vol.3, No.1, March 2004.

[42]. W. J. Townsend, J. A. Abraham, "Complex Gate Implementations for Quantum Dot Cellular Automata"; 4th IEEE Conference on Nanotechnology, 2004.

[43]. K. Walus, G. Schulhof, G. A. Jullien, "High Level Exploration of Quantum-Dot Cellular Automata (QCA)"; IEEE Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, November 7-10, 2004.

[44]. V. Vankamamidi, M. Ottavi, F. Lombardi, "A Line-Based Parallel Memory for QCA Implementation"; IEEE Transactions on Nanotechnology, Vol.4, No.6, November 2005.

[45]. C. R. Graunke, D. I. Wheeler, D. Tougaw, J. D. Will, "Implementation of a Crossbar Network Using Quantum-Dot Cellular Automata"; IEEE Transactions on Nanotechnology, Vol.4, No.4, July 2005.

[46]. M. T. Niemier, R. Ravichandran, P. M. Kogge, "Using Circuits and Systems-Level Research to Drive Nanotechnology"; Proceedings of the IEEE International Conference on Computer Design (ICCD'04), 2004.

[47]. S. C. Henderson, E. W. Johnson, J. R. Janulis, P. D. Tougaw, "Incorporating Standard CMOS Design Process Methodologies into the QCA Logic Design Process"; IEEE Transactions on Nanotechnology, Vol.3, No.1, March 2004.

[48]. R. Tang, F. Zhang, Y. B. Kim, "QCA-Based Nano Circuits Design"; IEEE International Symposium on Circuits and Systems, 2005, Vol.3, pp.2527-2530, 23-26 May 2005.

[49]. D Berzon, T. J. Fountain, "A Memory Design in QCAs using the SQUARES Formalism"; Proceedings of the Ninth Great Lakes Symposium on VLSI, p.166, March 04-06, 1999.

[50]. J. Huang, M. Momenzadeh, L. Schiano, F. Lombardi, "Simulation-based Design of Modular QCA"; Proceedings of 2005 5th IEEE Conference on Nanotechnology, July 2005.

[51]. K. Kim, K. Wu, R. Karri, "Towards Designing Robust QCA Architectures in the Presence of Sneak Noise Paths"; Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05), 2005.

[52]. K. Walus, T. J. Dysart, G. A. Jullien, R. A. Budiman, "QCADesigner: A Rapid Design and Simulation Tool for Quantum-Dot Cellular Automata"; IEEE Transactions on Nanotechnology, Vol.3, No.1, March 2004.

[53]. K. Walus, V. Dimitrov, G.A. Jullien, W.C. Miller, "QCADesigner: A CAD Tool for an Emerging Nano-Technology"; Micronet Annual Workshop 2003.

[54]. R. K. Kummamuru, A. O. Orlov, R. Ramasubramaniam, C. S. Lent, G. H. Bernstein, G. L. Snider, "Operation of a Quantum-Dot Cellular Automata (QCA) Shift Register and Analysis of Errors"; IEEE Transactions on Electron Devices, Vol.50, No.9, September 2003.

[55]. T. J. Dysart, P. M. Kogge, "Strategy and Prototype Tool for Doing Fault Modeling in a Nano-technology"; IEEE Nano Conference, San Francisco, CA, August 12-14, 2003.

[56]. M. Momenzadeh, M. Ottavi, F. Lombardi, "Modeling QCA Defects at Molecular-level in Combinational Circuits"; Proceedings of the 2005 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05), 2005.

[57]. M. Momenzadeh, J. Huang F. Lombardi, "Defect Characterization and Tolerance of QCA Sequential Devices and Circuits"; Proceedings of the 2005 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05), 2005.

[58]. T. Wei, K. Wu, R. Karri, A. Orailoglu, "Fault Tolerant Quantum Cellular Array (QCA) Design using Triple Modular Redundancy with Shifted Operands": Asia and South Pacific Design Automation Conference (ASP-DAC 2005), Shanghai, China, pp 1192-1195, January 18-21, 2005.

[59]. M. Momenzadeh, J. Huang, M. B. Tahoori, F. Lombardi, "On the Evaluation of Scaling of QCA Devices in the Presence of Defects at Manufacturing"; IEEE Transactions on Nanotechnology, Vol.4, No.6, November 2005.

[60]. M. Momenzadeh, J. Huang, M. B. Tahoori, F. Lombardi, " Characterization, Test, and Logic Synthesis of And-Or-Inverter (AOI) Gate Design for QCA Implementation"; IEEE Transactions on Computer-Aided Desing of Integrated Circuits and Systems, Vol.24, No.12, December 2005.

[61]. J. C. Lusth, "Balancing QCA Logic Gates under Image Charge neutralization"; IEEE NANO, WA3: Quantum Cellular Automata, August 2002.

[62]. J. C. Lusth, "Symmetric versus Asymeric Charge Neutraliazation in Quantum-Dot Cellular Automata"; IEEE NANO 2001, T1.1: Quantum Dots and Devices I, October 2001.

[63]. M. L. Bushnell, V. D. Agrawal, "Essentials of Electronic Testing for Digital, Memory, and Mixed-signal VLSI CIrcuits".

[64]. G. Toth, C.S. Lent, "Quasiadiabatic switching for metal-island quantum-dot cellular automata"; Journal of Applied Physics, Vol.85, No.5, pp.2977-2984, 1999.

[65]. W. Porod, "Quantum-dot Devices and Quantum-dot cellular automata"; International Journal of Bifurcation and Chaos, Vol.7, No.10, pp2199-2218, 1997.

[66]. G. Toth, C. S. Lent, "Quantum computing with quantum-dot cellular automata"; Physical Review B, Vol.63, pp.1-9, 2000.

[67]. C. S. Lent, "Molecular Electronics: Bypassing the Transistor Paradigm"; Science, Vol.288, pp.1597-1599, 2000.

[68]. R. P. Cowburn, M. E. Welland, "Room Temperature Magnetic Quantum Dot Cellular Automata"; science, Vol.287, pp.1466-1468, 2000.

[69]. M. C. B. Parish, "Modelling of Physical Constraints on Bistable Magnetic Quantum Cellular Automata"; PhD thesis, University of London, UK, 2003.

[70]. K. Radecka, Z. Zilic, "Verification by Error Modeling: Using Testing Methods in Hardware Verification"; Kluwer Academic Publishers, 2001.