A Design of and Design Tools for a Novel Quantum Dot Based Microprocessor

Michael T. Niemier
University of Notre Dame
Department of Computer Science and Engineering
Notre Dame, IN 46545
mniemier@nd.edu

Michael J. Kontz
University of Notre Dame
Department of Computer Science and Engineering
Notre Dame, IN 46545
mkontz@nd.edu

Peter M. Kogge
University of Notre Dame
Department of Computer Science and Engineering
Notre Dame, IN 46545
kogge@cse.nd.edu

ABSTRACT

Despite the seemingly endless upwards spiral of modern VLSI technology, many experts are predicting a hard wall for CMOS in about a decade. Given this, researchers continue to look at alternative technologies, one of which is based on quantumdots, called quantumcellular automata (QCA). While the first such devices have beenfabricated, little is known about how to design complete systems of them. This paper summarizes one of the first such studies, namely an attempt to design a complete, albeit simple, CPU in the technology. To design a theoretical QCA microprocessor, two things must be accomplished. First a device model of the processor must be constructed (i.e. the schematic itself). Second, methods for simulatingand testing QCA designs must be developed. This paper summarizes the beginnings of a simple QCA microprocessor (namely, its dataflow) and a QCA design and simulation tool.

Categories and Subject Descriptors

B.2 [**Hardware**]: Arithmetic And Logic Structures; B.6 [**Hardware**]: Logic Design; B.7 [**Hardware**]: Integrated Circuits; C.1 [**Computer Systems Organization**]: Processor Architectures

General Terms

Nanotechnology, Quantum Cellular Automata

1. INTRODUCTION

Today, many integrated circuits are manufactured with 0.25 - 0.33 microns processes. As device sizes decrease to an order of 0.05 microns, physical limitations of conventional electronics including power consumption, interconnect, and lithography will become increasingly difficult to surmount [5]. In fact, one study indicates that as early as 2010, the physical limits of transistor sizing may be reached [1]. Thus, to continue the norms of doubling the number of devices in a processor every two years, and doubling the clock rate every three years, other technologies must be studied.

As an alternative to CMOS VLSI, researchers have proposed an approach to computing with quantum dots,the quantum cellular automata (QCA). QCA is based on the encoding of binary information in the charge configuration within quantum dot cells. Computation power is provided by the Coulombic interaction between QCA cells. No current flows between cells and no power or information is delivered to individual internal cells. Local interconnections between cells are provided by the physics of cell-to-cell interaction due to rearrangement of electron positions [6].

This paper will begin by summarizing the basics of the QCA technology. It will then discuss a dataflow for a completely QCA microprocessor. Next, the tool used in the design and simulation of the dataflow will be discussed. Finally, a section discussing how our design was tested will be included.

2. THE BASICS OF QCA

A high-level diagram of a four-dot QCA cell appears in Figure 1. Four quantum dots are positioned to form a square. Two mobile electrons exist in the cell and can move to different quantum dots in the QCA cell by means of electron tunneling. This tunneling is assumed to be completely controllable by potential barriers that can be raised and lowered between adjacent QCA cells by means of capacitive plates [6].

For an isolated cell there are two energetically minimal equivalent arrangements of the two electrons in the QCA cell, denoted cell polarization P = +1 and cell polarization P = -1. Cell polarization P = +1 represents a binary 1 while cell polarization P = -1 represents a binary 0. This concept is also illustrated graphically in Figure 1 [6].

3. QCA BUILDING BLOCKS

QCA cells perform computation by interacting coulombically with neighboring cells to influence each other's polarization. In the following subsections we review some simple, yet essential, QCA logical devices: a majority gate, QCA "wires", and more complex combinations of QCA cells [6].

3.1 The Majority Gate

DAC 2000, Los Angeles, California

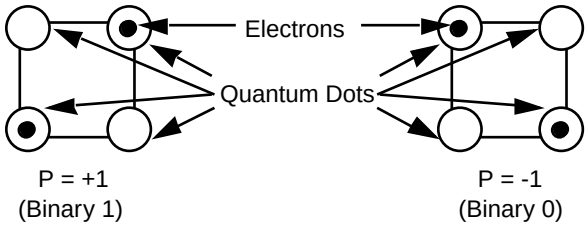

Figure 1: QCA cell polarizations and representations of binary 1 and binary 0.

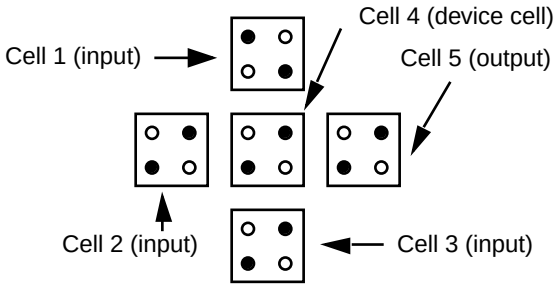

Figure 2: The fundamental QCA logical device - the majority gate.

The fundamental QCA logical circuit is the three-input majority gate that appears in Figure 2. Computation is performed with the majority gate by driving the device cell to its lowest energy state. This happens when it assumes the polarization of the majority of the three input cells. We define an input cell as one where the tunneling barriers have been raised to the point where the electrons are "trapped" in a polarization. The device cell will always assume the majority polarization because it is this polarization where electron repulsion between the electrons in the three input cells and the device cell will be at a minimum.

3.2 A 90-Degree QCA Wire

Figure 3 illustrates how a binary value propagates down the length of a QCA "wire". In this figure, the wire is a horizontal row of QCA cells. The binary signal propagates from left-to-right because of the Coulombic interactions between cells (i.e. Coulombic interaction causes Cell 2 to switch polarizations). In Figure 3, cell 1 has polarization P = -1 and cell 2 has polarization P = +1. A binary 0 (from polarization P = -1) will propagate down the length of the wire.

3.3 A 45-Degree QCA Wire

A QCA wire can also be comprised of cells oriented at 45-degrees as opposed to the 90-degree orientation discussed above. With the 45-degree orientation, as the binary value propagates down the length of the wire, it alternates between polarization P = +1 and polarization P = -1. A complemented or uncomplemented value can be ripped off the wire by placing a ripper cell at the proper location and considering the direction of signal propagation. The significant advantage of the 45-degree wire is that both a transmitted value and its complement can be obtained from a wire without the use of an explicit inverter!

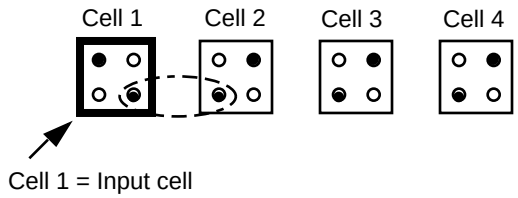

Figure 3: A QCA "wire".

3.4 Non-linear QCA Wires

Also, QCA cells do not have to be in a perfectly straight line to transmit binary signals correctly. Cells with a 90-degree orientation can be placed next to one another, but off center, and a binary value will still be transmitted successfully.

3.5 QCA Wires in the Plane

Finally, QCA wires possess the unique property that they are able to cross in the plane without the destruction of the value being transmitted on either wire. However, this property holds only if the QCA wires are of different orientations (i.e. one wire is a 45-degree wire and the other is a 90-degree wire).

3.6 QCA Logical Devices

To implement more complicated logical functions, a subset of simple logical gates is required. For example, it would be impossible to implement a multiplexor, decoder, or adder in QCA without a logical AND gate, OR gate, or inverter. It has been demonstrated that a value's complement can be obtained simply by ripping it off a 45-degree wire at the proper location. Implementing the logical AND and OR functions is also quite simple. The logical function for the majority gate is: $Y = AB + BC + AC$. The AND function can be implemented by setting one value (A, B, or C) this equation to a logical 0. Similarly, the OR function can be implemented by setting one value (A, B, or C) in equation to a logical 1. More complex logical circuits can then be constructed from AND and OR gates.

4. THE ROLE OF THE CLOCK IN QCA

The clock in QCA is multi-phased [4]. Individual QCA cells are not timed separately. However, an array of QCA cells can be divided into subarrays that offer the advantage of multi-phase clocking and pipelining. For each subarray, a single potential modulates the inter-dot barriers in all of the cells in one of the four clocking zones.

This clocking scheme allows one subarray to perform a certain calculation, have its state frozen by the raising of its interdot barriers, and have the output of that subarray act as the input to a successor array (i.e. clocking subarray 1 can act as input to clocking subarray 2). During the calculation phase, the successor array is kept in an unpolarized state so it does not influence the calculation. Each of the four clocking subarrays corresponds to a different clocking phase. Neighboring subarrays concurrently receive neighboring clocking phases.

During the first clock phase, the switch phase, QCA cells begin unpolarized and their interdot potential barriers are low. The barriers are then raised during this phase and

the QCA cells become polarized according to the state of their driver (i.e. their input cell). It is in this clock phase that the actual computation (or switching) occurs. By the end of the clock phase, barriers are high enough to suppress any electron tunneling and cell states are fixed. During the second clock phase, the hold phase, barriers are held high so the outputs of the subarray can be used as inputs to the next stage. In the third clock phase, the release phase, barriers are lowered and cells are allowed to relax to an unpolarized state. Finally, during the fourth clock phase, the relaxed phase, cell barriers remain lowered and cells remain in an unpolarized state [4].

5. A MICROPROCESSOR DATAFLOW

While there is still much work to be done, early physical results indicate that QCA could be a very viable alternative to CMOS. QCA wires and a QCA majority gate have been physically fabricated. Additionally, physical tests indicate that the devices do function as discussed above. However, the actual design of many of the circuits and devices required for a QCA microprocessor have not yet even been considered. To remedy this fact, we are designing and simulating a custom design of a microprocessor called Simple 12 entirely in QCA. A dataflow for the Simple 12 and a means for register/latch implementation will be discussed below.

5.1 Simple 12

The are three significant advantages in choosing Simple 12 as a base for generating an equivalent QCA design. First, the processor is simple. Simple 12 has 12-bit data words, an 8-bit addressable memory, and uses minimal hardware. It contains an ALU, accumulator, program counter, instruction register, and some control logic. Consequently, much of the physical layout can be performed by hand. Second, an actual processor will be designed with an instruction set that includes arithmetic loads, stores, and jumps. Therefore, solutions to the difficulties encountered in this design will apply to even more complex processors. Third, we have completed and fabricated a two micron CMOS Simple 12. Thus, it will be possible to make comparisons to an existing design in a technology on which we are trying to improve.

5.2 The Simple 12 Dataflow

A picture of a complete first-cut design of the QCA Simple 12 dataflow appears in Figure 4. This design is broken down into three subcomponents. One block represents the addition/subtraction unit of the ALU. Another "contains" the logic unit of the dataflow. Finally, the third contains intermediate signal generation logic. Each block is discussed in a separate subsection below.

5.2.1 The Addition/Subtraction Unit

The addition/subtraction unit is based on the full adder design (majority gate based) proposed by Lent, et. al. [4]. In our final design, the logic proposed by Lent, et. al. is used but its layout is different. It can easily be seen that by using majority gates, the adder that is produced is significantly different from a "normal" or conventional full adder. This majority gate single-bit full adder requires five majority gates and three inverters. However, because of the properties of 45 degree wires (i.e. a signal or its complement can be obtained from the wire), no explicit inverters are required in the QCA design.

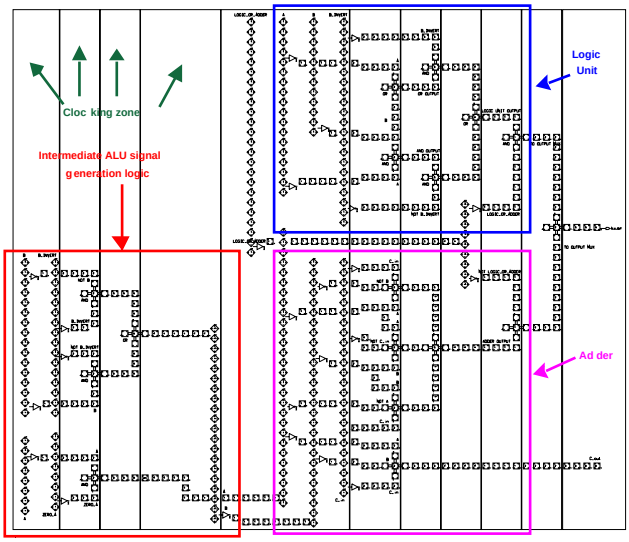

Figure 4: 1st cut of the QCA Simple 12 Dataflow.

5.2.2 The Logic Unit

To successfully execute the complete Simple 12 instruction set, the dataflow must be able to generate the following outputs: A+B, A-B, A AND B, A OR B, B, B+1, and 0. The logic unit of the dataflow will generate the outputs: A AND B, A OR B, B, and 0. (The output of the logic unit is then multiplexed with the output of the adder unit and one output from the dataflow is generated). The logic unit consists only of a majority gate with an input cell anchored so it performs the AND operation, a majority gate with an input cell anchored so it performs the OR operation, and a 2x1 multiplexor to select between the output of the AND and OR gate.

5.2.3 Intermediate Signal Generation

In Section 5.2.2, it was indicated that the logic unit had to generate the following outputs: A AND B, A OR B, B, and 0. One mechanism for generating B would be to OR every bit of B with a logical 1. However, to perform this operation, the other input to the logic unit, A, must be set to 1. In this case, the intermediate signal generation logic will perform such an operation on the input A based on the desired ALU operation and the correct bit will be sent to the logic unit. 0 is generated in a similar manner. This unit will also assist with adder operations (i.e. A-B).

5.3 The QCA Simple 12 ALU - Design Problems and Solutions

The first-cut of the QCA ALU has three significant problems. First, the clocking zones have different widths. If their widths are not uniform, the clock rate will be limited to the width of the largest clocking zone which will be slower than for a narrower clocking zone. A second difficulty with the design appearing in Figure 4 is that there is a large number of QCA cells per clocking phase. If too many cells are included in a clocking phase, the clock rate will deteriorate.

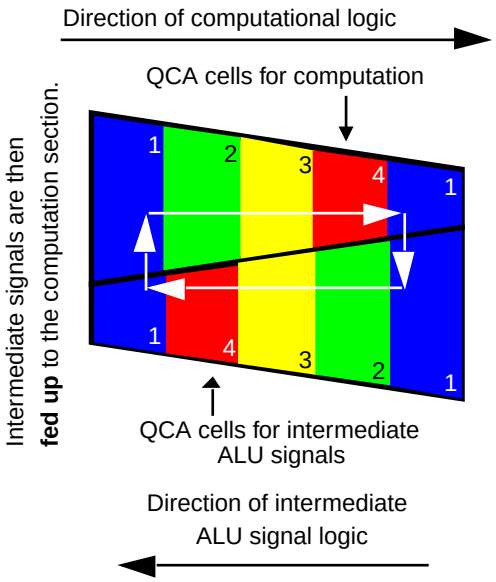

Figure 5: A description of trapezoidal clocking.

Also, for arrays of cells on the order of 10^3, there will be a tendency for the system to settle in an excited state rather than a ground state because of thermodynamic effects [3]. Finally, and most importantly, no physical feedback exists in this design which is all but essential in most useful microprocessors and finite state machines.

Solving the problems of clocking zone widths and the number of cells "placed" in each clocking zone is simple. To solve the problem of clocking zone widths, the schematic only needs to be adjusted to make them equal. The adiabatic clock rate can now increase. Furthermore, it is also possible to reduce the number of cells per clocking zone simply by adjusting the design and layout of the ALU. However, feedback still does not exist in this schematic. This problem and a means for implementing latches/registers will be discussed in the next subsection.

5.4 A More Efficient and Correct Design

By examining Figure 4, it is easy to see that other problems besides clocking zone width, an inappropriate number of cells per clocking phase, and the lack of physical feedback exist. One such problem is a significant amount of wasted area because of the logic required to generate intermediate ALU data. To remedy this problem, "trapezoidal clocking" and floorplanning will be introduced. In Figure 5, QCA logic to generate intermediate ALU data is not placed in front of the computational logic but rather, below it. Instead of leaving large gaps - or areas with no logic (like those appearing in Figure 4), "trapezoids" containing computational logic and intermediate signal generation logic can be fit together to minimize wasted area. The shaded regions in Figure 5 represent clocking zones. Thus, the computational and intermediate signal generation logic would still be divided up into clocking zones as depicted in Figure 4.

Trapezoidal clocking does not only provide a means for minimizing total area. It can also be used to implement a feedback path in QCA circuits. In Figure 5, the four clocking phases are each given a number (1, 2, 3, and 4) and a shade. These correspond to the four different clock phases that were discussed in Section 4. If the top "trapezoid" is computational logic, data can be fed back to the input (assumed to be clocking zone 1 at the far left) after "switching" in clocking zone 1 at the far right. Arrows illustrate the feedback path through the numbered clocking zones. It can be easily seen that the clocking phases are traversed in the proper order (i.e. in the order 1, 2, 3, 4). Furthermore, a signal can start a given point and a path exists to return to that point - the definition of feedback.

The clocking zone arrangement illustrated in Figure 5 can be extended to allow efficient and easy wire routing for control and data signals. Also, by arranging QCA cells in such a path, a wire can be "latched" indefinitely.

6. Q-BART

The logic of early versions of QCA components such as the logic unit or intermediate signal generation logic was simply checked "by-hand" with truth tables. Additionally, designs were created simply by using symbols to represent 45 of 90-degree QCA cells. No functionality could be attached to them. As designs grew larger, this lack of design automation and testing became more and more of a problem. To solve this problem, an architectural simulator, Q-BART (Quantum Based Architecture Rules Tool), was developed.

6.1 Q-BART's Graphical User Interface

As evidenced by discussions in Sections 2 and 5 of this paper, QCA designs are somewhat rigid. What this means is that QCA cells must be in a fairly straight, vertical or horizontal line to form a wire, majority gate, etc. (The only exception is an off-center wire and in the design in Figure 4, cells are only off-center by half of one cell). Additionally, when ripping a value off of or onto a 45-degree wire, a 90-degree cell must be placed exactly between two 45-degree cells. Because of such constraints, a grid structure was devised as a means for "laying out" QCA cells.

Different types of QCA devices (i.e. majority gates, 45-degree wires, 90-degree cells, etc.) can be added to the grid (and hence the design) by highlighting a grid square and pressing the appropriate device button. For example, to add a majority gate, the user should highlight the square where the device cell should be and then press the majority-gate button. A majority gate will then be added. Devices are represented by coloring cells appropriately (i.e. a 90-degree cell is represented by a dark blue 'X' while the cells that are part of the majority gate are red). Additionally, some coloring is done to help the user identify devices. A ripper cell is not colored dark blue, but rather another color to help the user identify what actually happens when these QCA cells interact. It is worth noting that in terms of the engine, the colored ripper cell would not be treated as such, but rather just as a 90-degree cell. This is done to simplify interaction rules and will be discussed further in the next subsection. It is also worth noting that the GUI can be and is used to display the results of a simulation. For instance, after a design has been entered, it can be simulated. During simulation all X's will change to 0 or 1s based on the interaction rules for types of QCA cells.

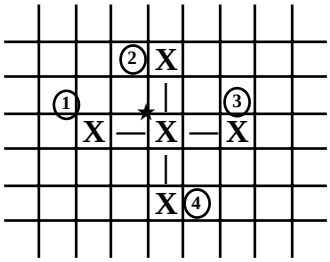

Figure 6: A graphical illustration of potential straight-adjacent 90 degree cell interactions.

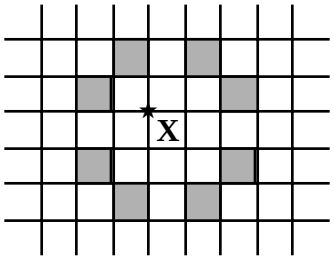

Figure 7: A graphical illustration of potential off-center 90-degree cell interactions.

6.2 Q-BART's Engine

To verify the logical correctness of a design, only a simple propagation QCA simulator is required. Such a simulator does not take into effect the effects of clocking zones in the design. A simulation simply starts at the inputs of the design. Each input cell is placed on a queue and is assigned timestep 0. Cells adjacent to the input cells are compared against the design/interaction rules of the engine. If a design rule indicates that the two cells can interact (i.e. a 90-degree cell can interact with an adjacent 90-degree cell), then the adjacent cell is placed on the queue. When cells that were initially on the queue are processed, the timestep will be incremented and the cells placed on the queue during the previous timestep will be tested for possible interactions. This process will continue until the queue is empty. This process gives the user the appearance that all changes during a given timestep occur simultaneously. It is worth mentioning that the next thing that will be added to the simulator is the ability to simulate with clocking zones.

The next sub-subsections discuss the design/interaction rules for the engine in detail.

6.2.1 A 90-Degree Cell Interacting with a 90-Degree Cell

All possible situations of this case are represented by Figure 6. All cells are assumed to be 90-degree cells. For this example, assume that the center cell just changed. Consequently, if the center 90-degree cell changes, locations 1, 2, 3, and 4 must be checked for existing 90-degree cells. If such cells exist, they will get the data associated with the cell that changed with one exception. If a cell in location 1, 2, 3, or 4 changed in the timestep just before cell $\star$ did, then it will not change. Why? Because cell $\star$ changed in response to this change.

6.2.2 A 45-Degree Cell Interacting with a 45-Degree Cell

This case is identical to the case 6.2.1. However, cells will get the complement of the value associated with the cell that changed.

6.2.3 A 90-Degree Cell Interacting with an Off-center 90-Degree Cell

All possible situations of this case are represented by Figure 7. Assume that all shaded cells are possible 90-degree cells. For this example, assume that the center cell just changed.

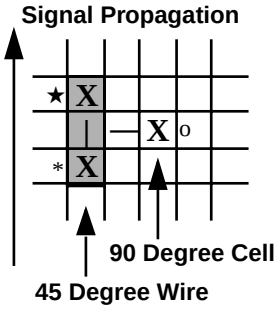

Figure 8: A graphical illustration of ripping a value off of a 45 degree wire to a 90-degree cell.

Consequently, if a 90-degree cell changes, shaded locations must be checked for existing 90-degree cells. If such cells exist, they will get the data associated with the cell that changed with one exception. If a cell in a shaded location changed in the timestep just before cell $\star$ did, then it will not change. Why? Because cell $\star$ changed in response to this change.

6.2.4 A 90-Degree Cell Getting a Value from a 45-Degree wire

One possible situation is illustrated in Figure 8. In Figure 8, cell "o" will receive the complement of the data that is associated with cell * (just like the "next" 45-degree cell of the 45-degree wire). If the signal propagation along the 45 degree wire was in the opposite direction (i.e. in the "up" direction), cell "o" would receive the data associated with cell * (NOTE: NOT the complement. This is because of electron positions within cells). As mentioned, each case is not illustrated here. However, the same design/interaction rule will apply in all cases.

6.2.5 An Input Cell of a Majority Gate Interacting with a Device Cell of a Majority Gate

This case is nearly identical to the case 6.2.1. However, unlike a simple cell in a 90-degree wire, the cell that can be influenced by the cell that changed (i.e. another 90-degree cell) does not just receive the data associated with the cell that changed. Here, the majority gate device cell should get the majority of the cells that surround it. In the simple propagation simulator, the simulator will simply wait until all inputs arrive for the device cell to change. However, in future versions of the simulator additional cell states will be introduced in an effort to mimic the lack of definitiveness

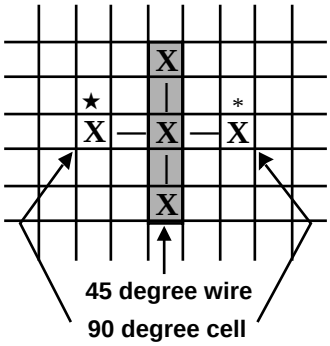

Figure 9: Situation for a crossover.

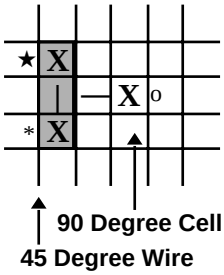

Figure 10: Ripping a value onto a 45-degree wire.

in a device cell if all three inputs have not arrived. Still, it is worth noting that the simple propagation simulator will nearly emulate the functionality of a design with clocking zones as a majority gate device cell usually is present just after the start of a clocking zone border. Consequently, all input wire lengths are nearly identical.

6.2.6 A Device Cell of a Majority Gate Interacting with a 90-Degree Cell

This case is identical to the case 6.2.1. The device cell will simply give its data to the 90-degree output cell of the majority gate.

6.2.7 A Crossover

One possible case is illustrated in Figure 9. In Figure 9, if cell $\star$ changes (a 90-degree cell), a check is made to see if a cell that is a 45-degree cell is directly in line with it. If it is, a check must be made for a 90-cell (cell *), on the other side of it. If cell * does exist, then it will receive the data associated with cell $\star$. Cell * will then be put on the queue.

6.2.8 Ripping a value from a 90-Degree Cell to a 45-Degree Cell

One case is illustrated in Figure 10. In this case, cell * will get the data associated with cell "o" and cell $\star$ will get the complement of this data. The other three cases are determined by electron positions.

7. SIMULATION RESULTS

Once the QCA dataflow has been entered into Q-BART, verifying its logical correctness is very simple. As mentioned, Q-BART allows designs to be constructed and simulated. Thus, all inputs and control signals in the design only need to have values assigned to them and the simulator can then be run.

To verify the functionality of the QCA Simple 12 dataflow, all operations that the dataflow must perform were tested with all possible combinations of A, B, (and in some cases C-in) inputs. These results were then compared against a logical schematic of the same design that was simulated using Mentor Graphics tools. In all cases, the results were identical indicating a theoretically correct QCA design. Enhanced designs can be tested in the same manner.

8. CONCLUSIONS

It is now worthwhile to consider the dimensions associated with the QCA technology and compare them with conventional CMOS. This is possible because prior to designing the QCA Simple 12, a CMOS Simple 12 had been designed and fabricated (at a 2 micron process). Given current projections for early fabrication runs, the expected distance between quantum dots is 10 nm. Furthermore, the diameter of a quantum dot is also 10 nm. The distance between centers of adjacent cells is on the order of 42 nm as there must be a slightly larger separation between electrons of neighboring cells [2].

Area measurements were taken for this QCA design and the equivalent area was determined for the CMOS implementation (with the CMOS numbers scaled to a 0.07 micron process - about the end of the CMOS curve). It was determined that QCA offers at least an order of magnitude area density increase over the equivalent CMOS design. Additionally, scientists are currently developing means for scaling the size of a QCA cell by a factor of 10. If these molecular dots are in fact implemented, potential density gains are three orders of magnitude.

At present the tools are in place to design and simulate more complex QCA designs. Future work will involve a control and memory interface unit and clocked QCA designs.

9. REFERENCES

[1] In *The National Technology Roadmap for Semiconductors*. Semiconductor Industry Association, 1997.

[2] G. H. Bernstein, G. Bazan, M. Chen, C. S. Lent, J. L. Merz, A. O. Orlov, W. Porod, G. L. Snider, and P. D. Tougaw. Practical issues in the realization of quantum-dot cellular automata. *Superlattices and Microstructures*, 20:447–459, 1996.

[3] D. Berzon and T. Fountain. Unpublished.

[4] C. S. Lent and P. D. Tougaw. A device architecture for computing with quantum dots. *Proceedings of the IEEE*, 85:541, 1997.

[5] J. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice Hall Electronics, New Jersey, 1996.

[6] P. Tougaw and C. Lent. Logical devices implemented using quantum cellular automata. *Journal of Applied Physics*, 75:1818, 1994.