QCADesigner: A Rapid Design and Simulation Tool for Quantum-Dot Cellular Automata

Konrad Walus, Timothy J. Dysart, Graham A. Jullien, Arief R. Budiman

Abstract— This paper describes a project to create a novel design and simulation tool for quantum-dot cellular automata (QCA), namely QCADesigner. QCA logic and circuit designers require a rapid and accurate simulation and design layout tool to determine the functionality of QCA circuits. QCADesigner gives the designer the ability to quickly layout a QCA design by providing an extensive set of CAD tools. As well, several simulation engines facilitate rapid and accurate simulation. This tool has already been used to design full-adders, barrel shifters, random-access memories, etc. These verified layouts provide motivation to continue efforts toward final implementation of QCA circuits.

***Index Terms*—quantum cellular automata, QCADesigner, simulation, design.**

I. Introduction

QCADesigner is the product of an ongoing effort to create a rapid and accurate simulation and layout tool for quantum-dot cellular automata (QCA). QCADesigner is capable of simulating complex QCA circuits on most standard platforms. Other published work describes some important QCA circuits in detail [16][17]. In this work, the various simulation engines available, and their relative benefits, are discussed. Additionally, an example of a random-access memory (RAM) designed and verified exclusively with QCADesigner is included. Section II is a brief summary of the QCADesigner project. In Section III, the three included simulation engines are considered. Section IV describes the signal clocking within QCADesigner. Section V outlines an example in which a RAM is designed and verified exclusively using QCADesigner. Section VI describes planned future additions. Section VII is a brief summary of other tools. Lastly, there is a brief section regarding the current state of QCADesigner and where it can be found. For those unfamiliar with QCA, [1][13] offer a thorough overview of the QCA architecture.

Manuscript received August 1, 2003. This work was supported in part by the financial support of research grants from the Natural Sciences and Engineering Research Council of Canada, Micronet R&D (a Network of Centres of Excellence), iCORE (Alberta), and workstations and tools from the Canadian Microelectronics Corporation and the Jet Propulsion Laboratory.

K. Walus is with the ATIPS Laboratory, University of Calgary, Department of Electrical and Computer Engineering, 2500 University Drive NW, Calgary, AB, Canada T2N 1N4 Phone: 403-210-9650, Fax: 403-282-6855 (e-mail: walus@atips.ca).

T. Dysart is with University of Notre Dame, 384 Fitzpatrick Hall, Notre Dame, Indiana, US 46556 (e-mail: tdysart@nd.edu).

G.A. Jullien is with the University of Calgary, Department of Electrical and Computer Engineering, 2500 University Drive NW, Calgary, AB, Canada T2N 1N4 (e-mail: jullien@atips.ca).

R.A. Budiman is with the University of Calgary, Department of Mechanical Engineering, 2500 University Drive NW, Calgary, AB, Canada T2N 1N4 (e-mail: budiman@enme.ucalgary.ca).

II. QCADesigner

Initially developed at the ATIPS Laboratory, University of Calgary, QCADesigner has attracted some important new developers, including top researchers from the University of Notre Dame. The project is written in C/C++ and employs a wide range of open-source software such as the GTK graphics library, and is maintained under the GNU public license for open source software. Developing the project in this manner enables it to be compiled and used on a wide range of systems. The objective of the project is to create an easy to use simulation and layout tool available freely to the research community via the Internet. 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.

The current version of QCADesigner has three different simulation engines included. The first is a digital logic simulator, which considers cells to be either null or fully polarized. The second is a nonlinear approximation engine, which uses the nonlinear cell-to-cell response function to iteratively determine the stable state of the cells within a design. The third uses a two-state Hamiltonian to form an approximation of the full quantum mechanical model of such a system.

One of the main problems in implementing more accurate simulations is the lack of experimental data for QCA systems with a large number of cells. However, several small QCA systems have been developed as proof-of-concept experiments [2]-[5]. As a result, the objective of this effort is to provide motivation for further research into implementing such devices and developing a dialog between circuit designers and implementers. This dialog will provide circuit designers with the knowledge necessary to design circuits that can be built while focusing the efforts of implementers on building

specific systems. Future versions of QCADesigner will include simulation engines built on the results gained from experimentation.

III. SIMULATION ENGINES

Currently, QCADesigner has three distinct simulation engines available. Each of the three engines has a different and important set of benefits and drawbacks. Additionally, each simulation engine can perform an exhaustive verification of the system or a set of user-selected vectors

A. Digital Simulation Engine

The digital simulation engine is a binary logic simulator within QCADesigner. This engine considers each cell to be in one of three states: null, logical one, or logical zero. With these three states and the appropriate clocking zone information for each cell, a design can be quickly simulated to ensure that for a given set of inputs, the correct set of outputs will be produced. This allows the logic designer to determine if the structure that has been laid out in QCADesigner corresponds to the desired logic function.

The simulator functions by first assigning values to the inputs of the design. Then, for each change in the clock, only cells about to switch are considered. The system clocking is considered in more detail in Section IV. Cells in the release and relax states are given a null value. Any cells about to switch, are processed by assigning them values based on the QCA interaction rules and the polarization of cells in their neighborhood. After all switching cells have been processed, their values are examined to ensure that none are null which ensures that the system will operate on all cells. Upon completing this check, the clock cycle is finished and the next cycle begins. This simulation will continue until all input combinations are exhausted and the last input vector has traversed the system. Since each cell can only receive input/output from its immediate neighbors in this engine, most QCA structures are able to be handled. However, there is an extremely small subset of designs that the simulator is not guaranteed to handle, and the reader is directed to the QCADesigner User's Guide [6] to examine this subset.

The advantage of this simulator is that a designer can quickly see if the logic functionality of a system corresponds to what is desired. Since no physical information outside of cell locations and orientation is needed by the simulator, it should remain an integral part of QCADesigner throughout its lifetime.

B. Nonlinear Approximation Simulation Engine

The nonlinear approximation simulation engine is built on a nonlinear approximation to the cell-to-cell response function. It has been shown that two cells have a nonlinear cell-to cell response function shown in figure 1 [14].

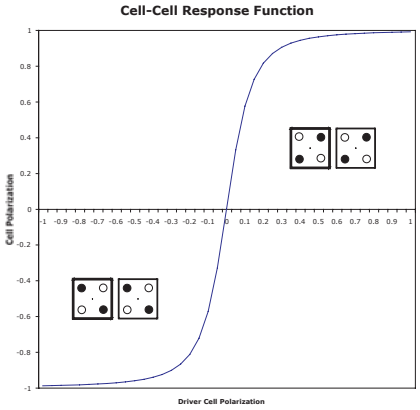

Fig. 1. Nonlinear cell-to-cell response function. The cell on the left is the input cell. The cell on the right is the output cell with its polarization determined by the above function.

This approximation excludes the quantum mechanical correlations between cells. The system is assumed to switch adiabatically, always remaining very close to the ground state. The polarization state of each of the cells is computed using

$$P_i = \frac{\frac{E_{i,j}^k}{2\gamma}\sum P_j}{\sqrt{1+\left(\frac{E_{i,j}^k}{2\gamma}\sum P_j\right)^2}} \quad (1)$$

Where P_i is the polarization state of the cell, and P_j is the polarization state of the neighboring cells. $E_{i,j}^k$ is the kink energy between cells; *i*, and *j* and represents the energy cost of oppositely polarized cells. γ is the tunneling potential and is used to clock the circuit, as described in [4]. 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 predetermined 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; but, as a result of 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 a design. For a more accurate simulation the two-state simulation engine is required.

C. Two-State Simulation Engine

To facilitate more accurate simulations we require a more advanced simulation engine. The two-state model assumes that the cell is a simple two state system, for which the

Hamiltonian is given by

$$H_i = \sum_j \begin{bmatrix} -\frac{1}{2} P_j E_{i,j}^k & -\gamma_j \\ -\gamma_j & \frac{1}{2} P_j E_{i,j}^k \end{bmatrix} \quad (2)$$

Using the Jacobi algorithm we are able to find the eigenvalues and vectors of the Hamiltonian.

$$H_i \psi_i = E_i \psi_i \quad (3)$$

Where H_i is the Hamiltonian given in (2). ψ_i is the state vector of the cell. E_i is the energy associated with the state ψ_i. The algorithm sorts each of the states, ψ_i, according to their respective energy, in ascending order. The first state in the sorted list is that which has the lowest energy. Our assumption is that the system remains very close to the ground state during computation. As a result, the state with the lowest energy is chosen and the cells polarization is set accordingly. The two state simulation engine computes the polarization of each cell in the design until the entire system has converged to a preset tolerance. Once the system has converged, the output values are recorded, new input values are set and the simulation is reiterated.

This method is less efficient in comparison to the nonlinear approximation, and as a result leads to longer simulation times. The advantage of this method is that the model on which it is based is far more accurate.

IV. QCADesigner Signal Clocking

In order to control the flow of information in a QCA circuit it has been shown that 4 clock signals are sufficient [7][8]. Each of the clock signals is shifted in phase by 90 degrees.

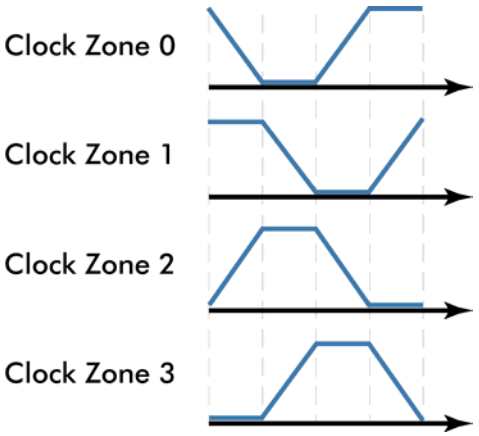

Fig. 2. The four available clock signals. Each signal is phase shifted by 90 degrees.

This allows information to be pumped through the circuit as a result of the successive latching and unlatching in cells attached to different clock cycles. For example, a wire, which is clocked from left to right with increasing clocking zones, will carry information in the same direction, i.e. from left to right. These four clocking zones are implemented in QCADesigner and each cell can be independently attached 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 the circuit has no control over the clocks. This means that information is transmitted through each cell and not retained. Each cell erases its own state every cycle of the clock. As a result, memory units must be created using loops of cells that continuously circulate the stored information as discussed in the next section.

V. QCA Random Access Memory (RAM)

Unlike CMOS memory, QCA has no equivalent for "static memory". Memory storage in this design is based on a circulating memory model similar to that proposed by the H-Memory architecture [11], which is a QCA implementation of a binary tree. Other designs have also recently been proposed [10]. The memory loop is divided into four consecutive clocking zones, each latching in succession. The stored memory continuously circulates in the loop until a write operation is performed at which time the memory is changed. Any read operation does not alter the memory.

Although serial structures, such as the proposed H-Memory, consume less clocking zones per bit, a parallel memory is more compatible with conventional architectures. Hence, this paper presents a layout of a conventional parallel memory architecture using QCA. The architecture is based on a simple 2D grid layout of memory cells. The rows of memory cells are addressed using a QCA decoder. The system diagram for the RAM is shown in figure 3.

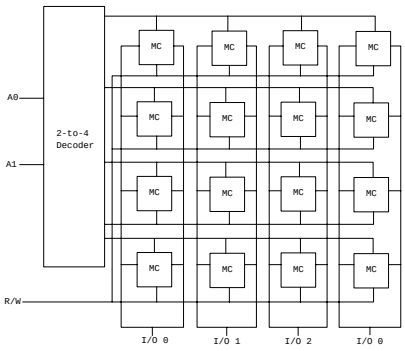

Fig. 3. System diagram of QCA RAM showing the architecture for a 4x4 memory.

Each of the rows of memory cells is addressable with the decoder. Once addressed, each memory cell in the RAM will be either written to or read from depending on which action is selected. The word size of the RAM is determined by the number of memory cells in each of the rows and can be increased by simply adding memory cells. To increase the total capacity of the RAM, the address space must also be increased. To do this we need to scale up the decoder as well

as add rows of memory cells to the array.

A. QCA MEMORY CELL

The proposed QCA memory cell is made of QCA logic gates, i.e., AND, OR and NOT gates. Moreover logic functions unique to the QCA technology such as the coplanar wire crossing, inversion chain [9] have been extensively exploited to ensure optimal design.

Each memory cell consists of 158 QCA cells. Since the design consists of a simple 2-dimensional grid structure, an n-bit memory would have *O(n)* cells. If we assume that cells are spaced 10nm apart, the design has a storage capacity of over 1.6 $Gbit/cm^2$. Further optimization could significantly increase this figure, possibly by an order of magnitude. One such optimization would be to store two or more bits per loop and maintain a parallel architecture.

The circuit schematic for the proposed QCA memory cell is shown in figure 4.

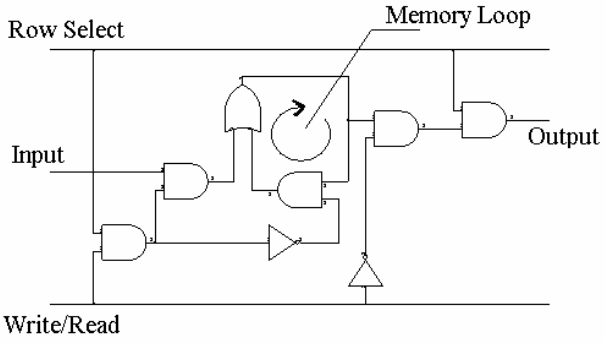

Fig. 4. QCA memory cell circuit schematic.

The memory value is constantly circulated inside the memory loop until the *Write/Read* and *Row Select* wires are polarized to 1, at which time the incoming input is fed into the memory loop and circulated. If the *Row Select* is polarized to 1 and the *Write/Read* is polarized to 0, the current memory value inside the loop is fed to the output. The circuit schematic shown is laid out to match that of the actual QCA layout. The final memory cell layout is shown in figure 5.

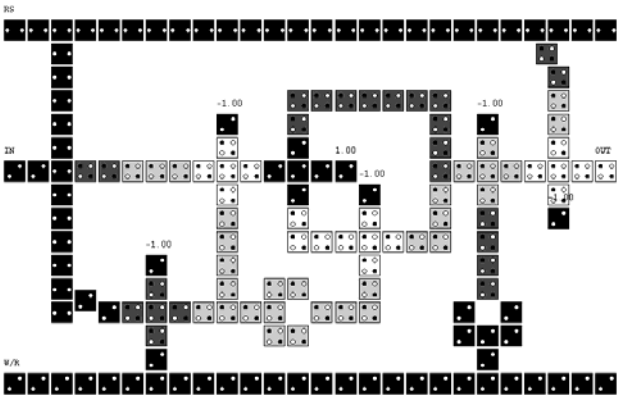

Fig. 5. Layout of QCA memory cell. The different shades of gray represent the different clocking zones to which cells are attached.

The memory loop in the center of the cell is clearly distinguished. Each shade of gray represents a different clocking zone. The darkest shaded cells are attached to clock 0, and the others are represented by successively lighter shades. The cells which have polarizations, shown directly above them, are fixed to that polarization. Fixed polarization cells are required since the fundamental logic gate in QCA is the majority gate, by fixing one of its inputs to either a 1 or 0 we can generate the standard AND, OR operations [9]. The control signals arrive at the memory cell from the right and the output signals propagate to the left. This memory cell is laid out such that it could easily be placed inside an array of other memory cells forming the complete RAM.

The overall use of cells in this unit could have been reduced by placing some of the interconnects closer together. Although this would reduce cell use, problems could arise from cross-talk between cell interconnects. In this design, we have used a separation of at least two cells between signal interconnect wherever the possibility for cross-talk exists; an example is the gap between the memory loop and some of the fixed polarization cells. There are some areas in the design where this was not necessary; e.g., interconnect attached to clocks 0 and clock 2. In this case, there is no problem with cross-talk because the two interconnects do not simultaneously transmit.

B. 1x4 QCA RAM

Using the memory cell described earlier and the QCA decoder, we can create a 2D addressable array. In Fig. 6, we show a 1x4 QCA RAM. The output from each of the cells is propagated downward through a serial OR array. The use of a serial OR array is necessary at this time since the equivalent of a tri-state buffer is not available in QCA technology.

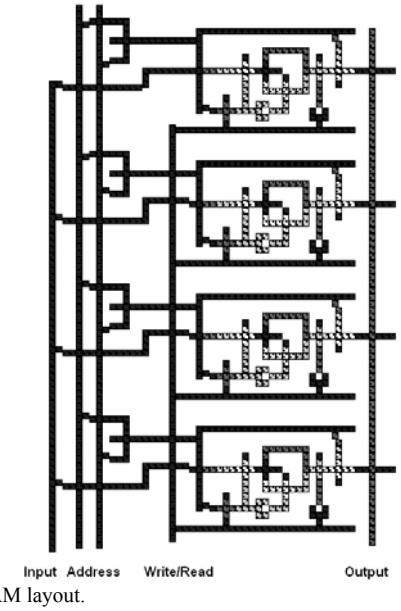

Fig. 6. 1x4 RAM layout.

Extending this design to larger word sizes involves adding memory cells to each row. Increasing the number of

addressable words requires scaling the decoder and adding more rows of memory cells. One of the challenges to using this design for larger memory sizes is that the delay is directly proportional to the row length since the *Row Select* signal has to propagate proportionately further to the outside memory cells.

VI. Future Work

Planned improvements for future versions include a design rule checker (DRC), hierarchical design capabilities, a fault tolerance simulation, and more accurate simulation and clocking models. Design rules, similar to the lambda rules of CMOS technology, are currently being developed at Notre Dame. After an initial set of these rules is available, a DRC will be written to verify that a design complies with all technology specific requirements. Hierarchical design abilities are a necessity for any CAD tool, and a rudimentary method of doing these is already available in QCADesigner. Additionally, a file format capable of easily handling hierarchical designs and a standard cell library is under development, with the XML standard being the leading candidate for specifying this file format. An additional requirement of this file format is to separate the architecture from the technology so that designs will be valid when manufacturing them with either metal-dot or molecular cells. The fault tolerance simulation engine will enable engineers to test the effects of fabrication and signal noise parameters not included in the ideal models. As the simulation methods and clocking models advance, they will be included in QCADesigner. One simulation engine that is currently highly desired is a more complete quantum mechanical simulation.

VII. Other Tools

QCADesigner is only one component of the QCA design tool set. Other "stand-alone" tools are being written that should be able to interact with QCADesigner, and as such, we will continue developing QCADesigner in a manner that allows for this growth in the tool set. An example tool that is currently under development is a fault modeling tool to examine the various effects of manufacturing defects within QCA systems that will aid in the development of defect tolerant QCA systems. Additionally, QCADesigner will be capable of handling new QCA technologies as they become viable for implementing systems. The capabilities already available and the inherent ability for growth within QCADesigner give us a robust tool for designing QCA circuits.

VIII. Conclusions

QCADesigner is a CAD tool designed specifically for QCA logic design and simulation. This tool allows users the ability to layout and verify any QCA systems that the user chooses. This functionality occurs due to the standard CAD features and various QCA specific simulation engines provided in the current version of QCADesigner. It is important to understand that although QCADesigner is currently in its infancy, great strides have been made in the development of QCA CAD tools. Additionally, a large amount of effort is being put forth to enhance QCADesigner and create a large, useful toolbox for QCA designers.

Lastly, the reader is reminded that QCADesigner is available at http://www.atips.ca/projects/qcadesigner and that a mailing list is available to keep users up to date with the most recent advances regarding this project [12].

Acknowledgment

The authors acknowledge the financial support of research grants from the Natural Sciences and Engineering Research Council of Canada, Micronet R&D (a Network of Centres of Excellence), iCORE (Alberta), workstations and tools from the Canadian Microelectronics Corporation, and the Jet Propulsion Laboratory.

References

[1] C. S. Lent, P. D. Tougaw, "A device architecture for computing with quantum dots", Proc. *IEEE*, vol. 85, no. 4, pp. 541 – 557, Apr. 1997.

[2] I. Amlani *et al*. "Experimental demonstration of a leadless quantum-dot cellular automata cell", *Appl. Phys. Lett*., vol. 77, no. 5, pp. 738-740, July 2000.

[3] A. Orlov *et al*., "Experimental demonstration of clocked single-electron switching in quantum-dot cellular automata", *Appl. Phys. Lett*., vol. 77, no. 2, pp. 295 – 297, July 2000.

[4] I. Amlani *et al*., "Digital logic using quantum-dot cellular automata", *Science*, vol. 284, pp. 289 – 291, Apr. 1999.

[5] A. Orlov *et al*., "Experimental demonstration of a binary wire for quantum-dot cellular automata", *Appl. Phys. Lett*., vol. 74, no. 19, pp. 2875 – 2877, May 1999.

[6] QCADesigner User's Guide, [Online] Available. http://www.atips.ca/projects/qcadesigner/downloads.html.

[7] G. Toth and C. S. Lent, "Quasiadiabatic switching for metal-island quantum-dot cellular automata", J. Appl. Phys., vol. 85, no. 5, pp. 2977 – 2984, Mar. 1999.

[8] K. Hennessy and C. S. Lent, "Clocking of molecular quantum-dot cellular automata", J. Vac. Sci. Technol. B., vol. 19, no. 5, pp. 1752 – 1755, Sept./Oct. 2001.

[9] Tougaw, P.D., Lent, C.S., "Logical devices implemented using quantum cellular automata", J. Appl. Phys., 75 (3) 1818, 1994

[10] D. Berzon, T J Fountain, "A Memory Design in QCA's using the SQUARES Formulism", Technical Report, University Collage London, UK, 1998

[11] S. Frost, A.F. Rodrigues, A.W. Janiszewski, R.T. Raush, P.M. Kogge, "Memory in motion: A study of storage structures in QCA", First Workshop on Non-Silicon Computing, 2002

[12] Walus, K. "ATIPS laboratory QCADesigner homepage", http://www.atips.ca/projects/qcadesigner, ATIPS Laboratory, University of Calgary, Calgary, AB, 2002

[13] W. Porod, "Quantum-dot devices and quantum-dot cellular automata", Inter. J. Bifurcation and Chaos, 7 (10) 2199, 1997

[14] G. Toth, "Correlation and coherence in quantum-dot cellular automata", Ph.D. Thesis, University of Notre Dame, 2000

[15] M. Governale, M. Maccuci, G. Iannaccone, C. Ungarelli, "Modeling and manufacturability assessment of bistable quantum-dot cells", J. Appl. Phys., 85 (5) 2962, 1999

[16] A. Vetteth, K. Walus, G. A. Jullien, V. S. Dimitrov, "Quantum dot cellular automata carry-look-ahead adder and barrel shifter", IEEE Emerging Telecommunications Technologies Conference, 2002

[17] A. Vetteth, K. Walus, G. A. Jullien, V. S. Dimitrov, "RAM design using quantum-dot cellular automata", Vol. 2, Pages 160-163, 2003 Nanotechnology Conference, February 23-27, 2003, San Francisco, CA.