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

Konrad Walus, *Student Member, IEEE*, Timothy J. Dysart, *Student Member, IEEE*, Graham A. Jullien, *Fellow, IEEE*, and R. Arief 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 a final implementation of QCA circuits.**

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

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 [1], [2]. In this paper, 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, [3] and [4] offer a thorough overview of the QCA architecture.

Manuscript received August 4, 2003; revised October 1, 2003. This work was supported in part by research grants from the Natural Sciences and Engineering Research Council of Canada, Micronet Research and Development (a Network of Centres of Excellence), iCORE (AB, Canada), workstations and tools from the Canadian Microelectronics Corporation and the Jet Propulsion Laboratory, Pasadena, CA. This paper was presented in part at the Second International Conference on Quantum Dots for Quantum Computing and Classical S Effects Circuits, 2003.

K. Walus and G. A. Jullien are with the ATIPS Laboratory, Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada T2N 1N4 (e-mail: walus@atips.ca; jullien@atips.ca).

T. J. Dysart is with the University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: tdysart@nd.edu).

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

Digital Object Identifier 10.1109/TNANO.2003.820815

II. QCADesigner

Initially developed at the ATIPS Laboratory, University of Calgary, Calgary, Canada, QCADesigner has attracted some important new developers, including top researchers from the University of Notre Dame, Notre Dame, IN. The project is written in C/C++ and employs a wide range of open-source software such as the GNU image manipulation program toolkit (GTK+) graphics library, and is maintained under the GNU's not Unix (GNU) public license (GPL) 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.

Included in the current version of QCADesigner are three different simulation engines. 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 [5]–[8]. 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 Users Guide [9] 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 Fig. 1[10].

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

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

where P_i is the polarization state of the cell, and P_j is the polarization state of the neighboring cells. $E^k_{i,j}$ 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 [7]. 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.

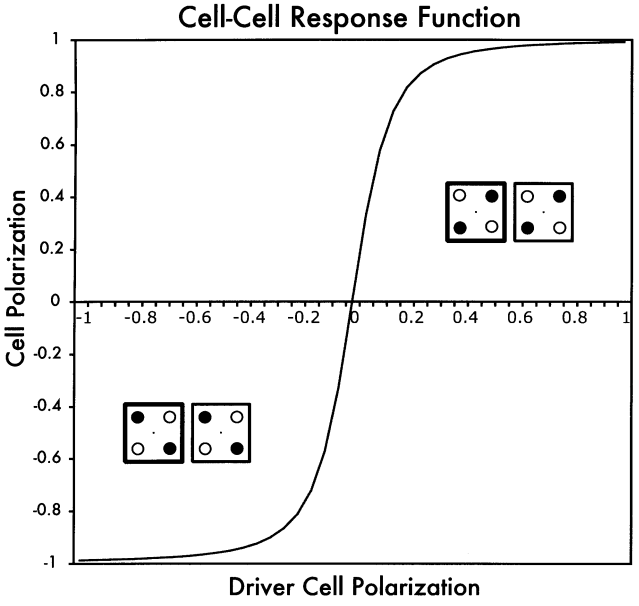


Fig. 1. Nonlinear cell-to-cell response function. The cell on the left is the input cell. The cell on the right is the output cell with its polarization determined by the above function.

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 this two-state system, it has been shown that the following Hamiltonian can be constructed [10]

$$H_i = \sum_j \begin{bmatrix} -\frac{1}{2}P_j E^k_{i,j} & -\gamma_i \\ -\gamma_i & \frac{1}{2}P_j E^k_{i,j} \end{bmatrix} \tag{2}$$

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. The summation is over all cells within an effective radius of cell i, and can be set prior to the simulation. Using the time-independent Schrödinger equation we are able to find the stationary states of the cell in the environment described by this Hamiltonian. QCADesigner uses

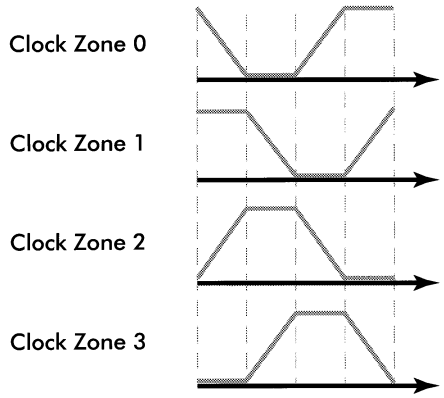


Fig. 2. The four available clock signals. Each signal is phase shifted by 90°.

the Jacobi algorithm to find the eigenvalues and eigenvectors of the Hamiltonian with

$$H_i\psi_i = E_i\psi_i \tag{3}$$

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. 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 cell 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. As well, we can include QCA dynamics for coherent systems by solving the time-dependent Shrödinger equation.

IV. 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 [11], [12]. Each of the clock signals is shifted in phase by 90°as shown in Fig. 2.

This allows information to be pumped through the circuit as a result of the successive latching and unlatching in cells connected 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 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 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 Section V.

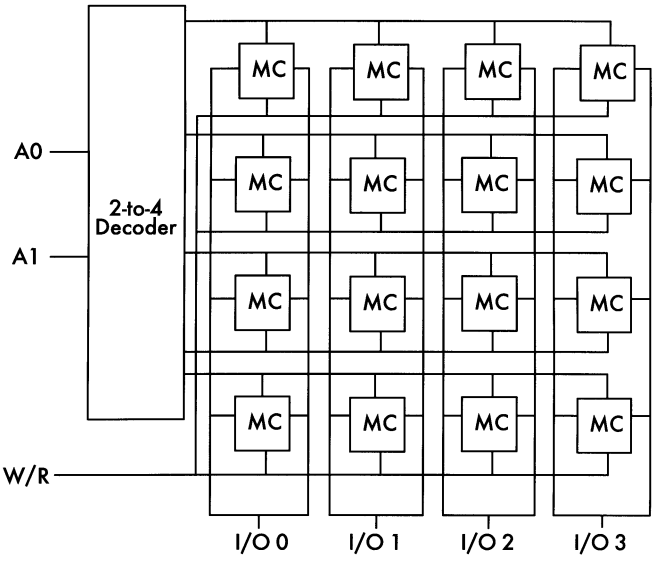


Fig. 3. System diagram of QCA RAM showing the architecture for a 4 × 4 memory.

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 [13], which is a QCA implementation of a binary tree. Other designs have also been proposed [14]. 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 fewer 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 two-dimensional (2-D) 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 Fig. 3.

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 and

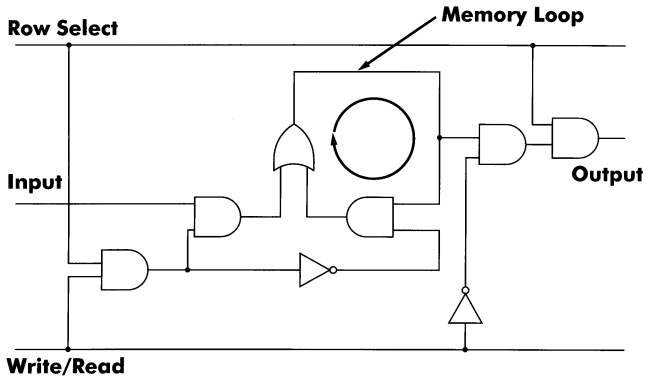


Fig. 4. QCA memory cell circuit schematic.

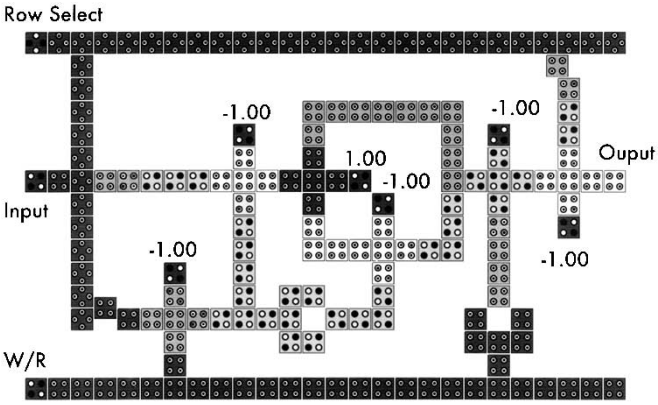


Fig. 5. Layout of QCA memory cell. The different shades represent the different clocking zones to which cells are attached.

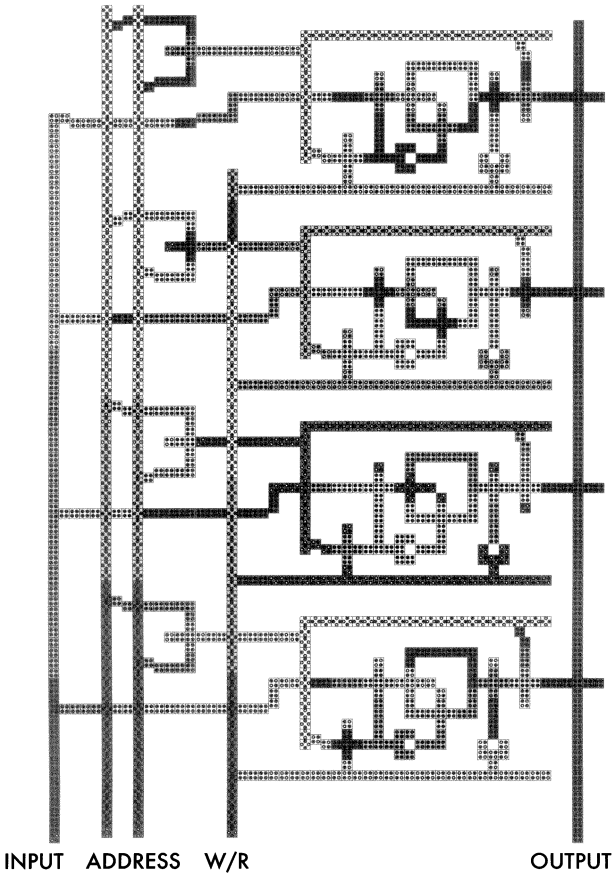


Fig. 6. 1×4 RAM layout.

inversion chain [15] have been extensively exploited to ensure optimal design.

Each memory cell consists of 158 QCA cells. Since the design consists of a simple 2-D grid structure, an n-bit memory would have $O(n)$ cells. If that cells are assumed to be 100 nm^2 in area, the design has a storage capacity of over 2.4 Gb/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 Fig. 4.

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 in Fig. 4 is laid out to match that of the actual QCA layout. The final memory cell layout is shown in Fig. 5.

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 the standard OR and AND functions are generated [15]. 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 crosstalk between cell interconnects. In this design, a separation of at least two cells between signal interconnects is used wherever the possibility for crosstalk 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 crosstalk because the two interconnects do not simultaneously transmit.

B. 1×4 QCA RAM

Using the memory cell described earlier and the QCA decoder, a 2-D addressable array is created. In Fig. 6, we show a 1×4 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 tristate buffer is not available in QCA technology.

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 [17]. 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 independent of the method used to manufacture them. 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 standalone tools are being written that should be able to interact with QCADesigner, and as such, the development QCADesigner will continue 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 [17]. 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 provides a robust tool for designing QCA circuits.

VIII. Conclusion

QCADesigner is a CAD tool designed specifically for QCA logic design and simulation. This tool allows users the ability to layout and verify a variety of QCA systems. 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 [9].

References

[1] A. Vetteth *et al.*, "Quantum dot cellular automata carry-look-ahead adder and barrel shifter," in *Proc. IEEE Emerging Telecommunications Technologies Conf.*, 2002.

[2] ——, "RAM design using quantum-dot cellular automata," in *Proc. 2003 Nanotechnology Conf.*, vol. 2, 2003, pp. 160–163.

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

[4] W. Porod, "Quantum-dot devices and quantum-dot cellular automata," *Int. J. Bifurcation Chaos*, vol. 7, no. 10, pp. 2199–2218, 1997.

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

[6] 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, 2000.

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

[8] 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, 1999.

[9] K. Walus. (2002) ATIPS Laboratory QCADesigner Homepage. ATIPS Laboratory, Univ. Calgary, Calgary, Canada. [Online]. Available: http://www.atips.ca/projects/qcadesigner

[10] G. Toth, "Correlation and Coherence in Quantum-Dot Cellular Automata," Ph.D. dissertation, Univ. Notre Dame, Notre Dame, IN, 2000.

[11] 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, 1999.

[12] 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, 2001.

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

[14] D. Berzon and T. J. Fountain, "A Memory Design in QCA's Using the SQUARES Formulism, Tech. Rep.," Univ. College London, London, U.K., 1998.

[15] P. D. Tougaw and C. S. Lent, "Logical devices implemented using quantum cellular automata," *J. Appl. Phys.*, vol. 75, no. 3, pp. 1818–1825, 1994.

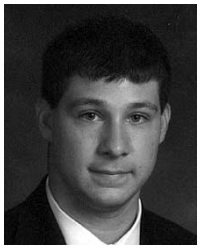
[16] M. Niemier, "The Effects of a New Technology on the Design, Organization, and Architectures of Computing Systems," Ph.D. dissertation, Univ. Notre Dame, Notre Dame, IN, 2003.

[17] T. J. Dysart and P. M. Kogge, "Strategy and prototype tool for doing fault-modeling in a nano-technology," in *Proc. IEEE 3rd Conf. Nanotechnology*, 2003.

Konrad Walus (S'02) received the electrical engineering degree from the University of Windsor, ON, Canada. He is currently working toward the Ph.D. degree in electrical engineering at the University of Calgary, AB, Canada.

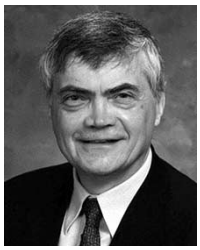
From 2000 to 2001, he was a Student Systems Engineer with AgentWare Systems Incorporated, Canada.

Mr.Walus is the recipient of several awards including the National Science and Engineering Council Scholarship, iCORE scholarship, Deans Research Award, and the Micralyne Microsystems Design Award. He has recently been named a finalist in the Alberta Science and Technology, Leaders of Tomorrow Award.



Timothy J. Dysart (S'03) received the B.S. degree in computer engineering from the University of Notre Dame, Notre Dame, IN, in 2002, where he is currently pursuing the M.S. degree in computer science and engineering.

His research interests include defect tolerant nanotechnologies (namely quantum-dot cellular automata), computer-aided design tools, and computer architecture.



Graham A. Jullien (F'02) received the Ph.D. degree in electrical engineering, from Aston University, Birmingham, U.K., in 1969.

From 1961 to 1966, he was a Student Engineer and Data Processing Engineer with English Electric Computers, U.K. From 1975 to 1976, he was a Visiting Senior Research Engineer with Central Research Laboratories of EMI Ltd., U.K. From 1969 to 2000, he was with the Department of Electrical and Computer Engineering, University of Windsor, ON, Canada, where he held the rank of University Professor, and was the Director of the VLSI Research Group. Since January 2001, he has been with the Department of Electrical and Computer Engineering, University of Calgary, AB, Canada, where he holds the iCORE Research Chair in Advanced Technology Information Processing Systems. He has published widely in the fields of digital signal processing, computer arithmetic, neural networks and VLSI systems, and teaches courses in related areas.

Dr. Jullien is a Member of the Board of Directors of the Canadian Microelectronics Corporation (CMC) and a Member of the Steering Committee and Board of Directors of the Micronet Network of Centers of Excellence. He has served on the technical committees of many international conferences, and currently serves on the Editorial Board of the *Journal of VLSI Signal Processing*. He is a past Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS. He hosted and was Program Cochair of the 11th IEEE Symposium on Computer Arithmetic, was Program Chair for the 8th Great Lakes Symposium on VLSI, and was Technical Program Chair for the 1999 Asilomar Conference on Signals, Systems and Computers. He was General Chair for the 2003 Asilomar Conference and was General Cochair of the International Workshop on System-on-Chip for Real-Time Systems, Calgary, AB, Canada, 2003.

R. Arief Budiman received the Ph.D. degree in materials science from the University of Toronto, ON, Canada, in 2002.

Currently, he is an Assistant Professor at the University of Calgary, AB, Canada. His research interests are self-assembly of nanostructures and their device applications.

Dr. Budiman received the Silver Graduate Student Award at the Materials Research Society 2001 Fall Symposium, and the GSI-Lumonics First Prize at the 2001 Canadian Association of Physicists Congress.