

Testing of Quantum Cellular Automata

Mehdi B. Tahoori, Jing Huang, Mariam Momenzadeh, and Fabrizio Lombardi

Abstract—There has been considerable research on quantum dot cellular automata (QCA) as a new computing scheme in the nanoscale regimes. The basic logic element of this technology is the majority voter. In this paper, a detailed simulation-based characterization of QCA defects and study of their effects at logic level are presented. Testing of these QCA devices at logic level is investigated and compared with conventional CMOS-based designs. Unique testing features of designs based on this technology are presented and interesting properties have been identified. A testing technique is presented; it requires only a constant number of test vectors to achieve 100% fault coverage with respect to the fault list of the original design. A design-for-test scheme is also presented, which results in the generation of a reduced test set at 100% fault coverage.

Index Terms—Defect characterization, design-for-testability, nanotechnology, quantum dot cellular automata (QCA), testing.

I. INTRODUCTION

OVER THE last few decades, the exponential scaling in feature sizes and the increase in processing power have been successfully achieved by conventional lithography-based very large scale integration (VLSI) technology. However, this faces serious challenges due to the fundamental physical limits of CMOS technology such as ultra-thin gate oxides, short channel effects, doping fluctuations, and the increasingly difficult and expensive lithography in nanoscale regimes.

There has been extensive research in recent years at the nanoscale to supersede conventional CMOS technology. It is anticipated that these technologies can achieve a density of 10^{12} devices/cm² and operate at terahertz frequencies.

One of the fundamental issues in the testing community is the radical shift in computation and fabrication technology and its effect on the test flow. Under these circumstances, do test generation and design-for-test becomes even intractable? Since the manufacturing process for nanodevices is ill defined, it is extremely difficult to address manufacturing testing problems. However, it would be inappropriate to ignore testing of these devices until manufacturing is realizable. This paper tries to address this issue for one of the proposed trends in the nanometer era.

Among these new devices, quantum dot cellular automata (QCA) not only gives a solution at nanoscale, but it also offers a new method of computation and information transformation. In terms of feature size, it is projected that a QCA cell of a few

nanometers in size can be fabricated through molecular implementation by a self-assembly process.

The unique feature of QCA-based designs is that logic states are not stored in voltage levels as in conventional electronics, but they are represented by the position of individual electrons. For QCA, the cells must be aligned precisely at nanoscales to provide correct functionality, thus, proper testing of these devices for manufacturing defects and misalignment plays a major role for quality of QCA-based circuits. The basic logic element in this technology is the majority voter (MV). Since the basic logic elements of QCA-based designs are different from conventional CMOS designs, they need different testing schemes. One of the interesting features of this technology is that it is possible to investigate some of the manufacturing issues (especially defects at nanoscale) by quantum-mechanics simulation of these devices.

In this paper, the defect characterization of these devices has been extensively studied; effects of defects are investigated at the logic level. Testing of QCA is also compared with testing of conventional CMOS implementations of these logic devices. The approach proposed in this paper is based on simulating different manufacturing misalignments, investigating their effects at the logic level, and identifying the test vectors for detection of all faults. Different fabrication schemes of an MV at the cell level are performed; these different implementations are compared in terms of defect tolerance and testability.

As test sets generated based on the stuck-at fault model are quite acceptable for testing conventional CMOS-based designs, this model does not fully capture the behavior of most of the prevalent defects in the CMOS fabrication process. Thus, it is possible to investigate the effectiveness of stuck-at test sets for QCA defects even though QCA defect mechanisms cannot be modeled by the stuck-at model. This aspect is investigated in detail in this paper.

Defect injection is also exploited to study the behavior of QCA-based circuits in the presence of defects and to measure the effectiveness of different test sets for detecting them. Unique testing properties of this technology have been identified in this study. Constant testability (C testability) of QCA designs based on MVs is investigated. An efficient test generation approach has been proposed. A design-for-test scheme is introduced to improve testability of these designs.

The remainder of this paper is organized as follows. In Section II, a review of QCA is presented. In Section III, testing of QCA-based design at the logic level is discussed. In Section IV, the defect characterization of QCA is presented. In Section V, test-set generation and coverage are discussed. Design for testability issues of QCA are presented in Section VI. Finally, Section VII concludes this paper.

Manuscript received March 24, 2004; revised May 11, 2004.

The authors are with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: mtahoori@ece.neu.edu; hjing@ece.neu.edu; mmomenza@ece.neu.edu; lombardi@ece.neu.edu).

Digital Object Identifier 10.1109/TNANO.2004.834169

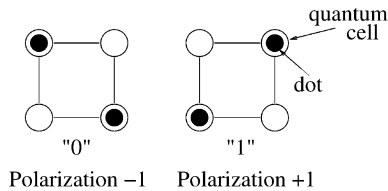


Fig. 1. QCA cell polarization.

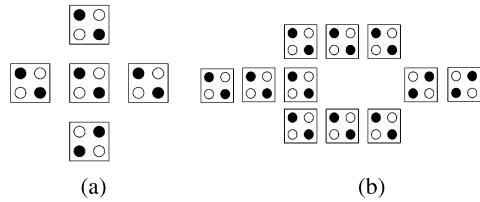


Fig. 2. QCA devices.

II. REVIEW OF QCA

QCA is a novel device that stores logic states not as voltage levels, but rather based on the position of individual electrons. A quantum cell can be viewed as a set of four charge containers or “dots” positioned at the corners of a square. The cell contains two extra mobile electrons, which can quantum mechanically tunnel between dots, but not cells. The electrons are forced to the corner positions by Coulomb repulsion. The two possible polarization states represent logic “0” and logic “1,” as shown in Fig. 1.

Unlike conventional logic in which information is transferred from one place to another by electrical current, QCA does so by the Coulomb interaction that connects the state of one cell to the state of its neighbors. This results in a technology in which information transfer (interconnection) is the same as information transformation (logic manipulation). Power dissipation in QCA circuits is ultra low compared with conventional CMOS circuits [1]–[3].

The basic logic gate in QCA is the MV. The MV with logic function $MV(A, B, C) = AB + AC + BC$ can be realized by only five QCA cells (compared to a CMOS implementation, which requires 16 transistors), as shown in Fig. 2. The logic AND and logic OR functions can be implemented from an MV by setting one input (the programming input) permanently to 0 and 1, respectively. The inverter can be seen in Fig. 2; unlike conventional CMOS, in which the inverter is the simplest block, in QCA, this device consumes a substantial area, as given by the number of QCA cells. There are other ways to obtain inverting function in QCA, for example, two cells in a diagonal orientation exhibit an inverting behavior. However, the inverter in Fig. 2 is preferable in QCA design, as it is more robust and more fault tolerant with respect to cell displacement faults. The binary wire and the inverter chain as interconnects are shown in Fig. 3.

The concept of clocking for QCAs has been introduced in [4]; clocking consists of four zones. QCA memories based on recursive H structures have been proposed in [5]. Therefore, sequential, as well as combinational designs can be realized using QCA. Some designs based on QCA [including carry look-ahead adder, barrel shifter, microprocessors, and field-programmable

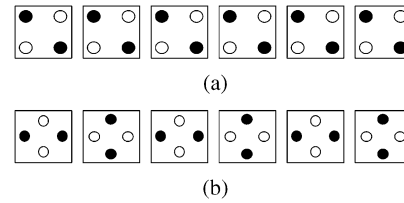


Fig. 3. Binary wire.

gate arrays (FPGAs)] have been proposed in the technical literature [6]–[9].

Currently, microsized QCA devices have been fabricated with metal cells, which operates at 50 mk [1], [10]. In [1], an experimental demonstration of a basic QCA cell has been presented. The device under study is composed of four metal dots, connected with tunnel junctions and capacitors. These experiments have confirmed that switching of a single electron in a double dot cell can control the position of a single electron in another double dot. In [11], basic logic circuits made of these cells has been demonstrated. It is predicted that molecular scale (~ 2 nm) will yield QCA operations at room temperature. Sequential circuits has also been fabricated using metal tunnel junction technology; the operation of a QCA latch and a two-bit shift register have been demonstrated in [12]. Recent research have focused on fabricating molecular-sized QCA, in which higher speed and room-temperature operation can be realistically expected. Reference [13] proposes some possible realizations of molecular QCA. It describes the progress for making QCA molecules and details the surface attachment chemistry that is compatible with QCA. An initial analysis of a simple molecular system that acts as a molecular QCA cell is presented in [14]; redox sites act as dots, while tunneling is provided by bridging ligands. A theoretical demonstration of a molecular QCA MV is also provided in [14].

The fault-tolerant properties of the MV under some manufacturing misalignments have been reported by researchers at the National Aeronautics and Science Administration (NASA), Washington, DC [15]–[17]. Based on this simulation-based study, a fault-tolerant MV block has been proposed. It has been shown that MV is more vulnerable to misalignment in the vertical direction than in the horizontal direction. A misalignment (at least equal to half a cell width in the vertical direction) causes the MV to malfunction. This confirms that a complete test of designs based on MVs is extremely needed.

III. LOGIC-LEVEL TESTING

A. *C* Testability of MV-Based Designs

Here, *C* testability of a one-dimensional (1-D) array of MVs is discussed. We present a 100% stuck-at fault test set for a chain of n MVs, as shown in Fig. 4. A 100% single stuck-at fault (SSF) test set for a single MV has a minimal length of 4, such as $\{010\ 011\ 100\ 101\}$, $\{001\ 011\ 100\ 101\}$, $\{001\ 010\ 101\ 110\}$.

As shown in Fig. 4, n MVs are concatenated into a 1-D (1-D or linear) chain. A is the primary data input and B_i and C_i are the control inputs. By applying $B_i C_i = 01$ for all MVs in the chain and set the primary input A to 0, any stuck-at-1 fault on

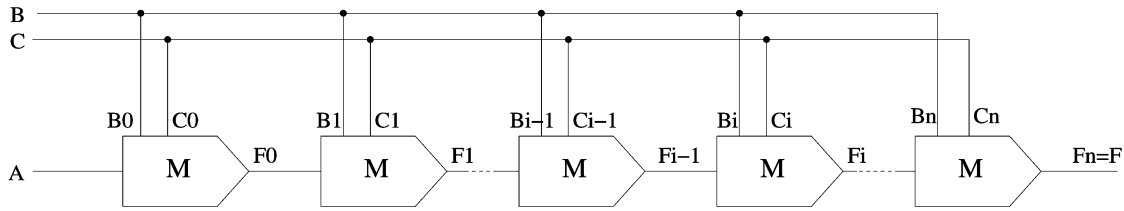
Fig. 4. Chain of n MVs.

TABLE I
DETECTING SSF IN MAJORITY VOTER CHAIN

SSF	test vectors (A B C)			
	001	010	101	110
A/0			x	x
A/1	x	x		
Bi/0				x
Bi/1	x			
Ci/0			x	
Ci/1		x		
Fi/0			x	x
Fi/1	x	x		

F_i and B_i will be detected. This occurs because the MV chain is effectively converted to a chain of OR gates with inputs F_i and B_i . As the test vector for these inputs are 00, any stuck-at-1 fault will be detected. Similarly, by applying $A = 1$, any stuck-at-0 fault on F_i or C_i will be detected (a 1-D chain of AND gates with inputs F_i and C_i). To detect B_i stuck-at-0 or C_i stuck-at-1 faults, we need more vectors: $B_i C_i = 01$ and setting A to 1 and 0, respectively.

A/1 (A stuck-at-1) can be detected by vectors 001 or 010. A is set to 0 to sensitize the fault, $BC = 01$ or 10 will propagate the fault to F . Similarly, **A/0** is detected by 101 and 110. **Bi/1** is detected by 001. In this case, the faulty gate is the MV i with inputs F_{i-1}, B_i, C_i and output F_i . B is set to 0 to sensitize the fault. As $BC = 01$, then A will be propagated to F_{i-1} (i.e., $F_{i-1} = A = 0$). Thus, the stuck-at fault at B_i can be propagate to F_i because the other inputs of the gate are 0 and 1. Also, $BC = 01$, thus, the faulty value will be propagated to the primary output F . Similarly, **Bi/0** is detected by 110. **Fi/1** is detected by vectors 001 or 010. Since $BC = 01, 10, A = 0$ if it is propagated to F_i to sensitize the fault. The faulty value will then propagate down the chain and reach the primary output F . Similarly, **Fi/0** is detected by 101 and 110. The SSF detected by each test vector is shown in Table I.

Hence, any 1-D chain of MVs independently of its length n can be tested for all stuck-at faults by only four vectors, i.e., it is C testable. This can be generalized to a two-dimensional (2-D) network of MVs. Note that, in the 1-D chain, 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 by getting the values of the control inputs.

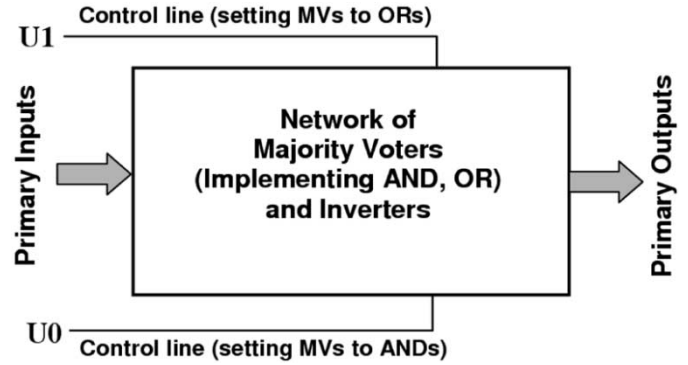


Fig. 5. QCA implementation of logic networks using MVs (implementing AND and OR) and inverters.

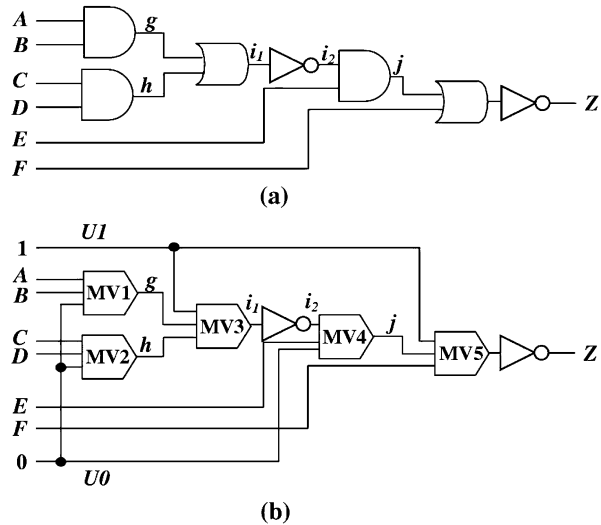


Fig. 6. (a) Simple AND-OR logic. (b) MV-based implementation.

B. Stuck-at Test Properties

The overall structure of a QCA implementation for (combinational) logic designs is shown in Fig. 5. The block consists of an interconnection of MVs and inverters. There are two system-level control lines, i.e., U_0 and U_1 , which are connected to MVs. U_0 is connected to logic "0" and sets some MVs to the AND function, whereas U_1 is connected to logic "1" and sets the other MVs to the OR function. A simple example is shown in Fig. 6. These control lines provide additional controllability because these lines can be seen as extra input lines during testing time. This unique feature of QCA can be exploited to achieve a higher test coverage and quality. However, no design-for-testability scheme comes for free. Adding the

global control lines may also generate additional wire crossings in the design. As will be shown below, coplanar wire crossing is very vulnerable to defects and, thus, difficult to manufacture. There has been some research in QCA placement and routing problems [18].

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 implementations.

Consider an MV with input lines A, B , and C and output line Z (where $Z = AB + AC + BC$).

Property 1: Consider an MV with input values a, b , and c (for lines A, B , and C , respectively) and output z . If all inputs are flipped, $abc \rightarrow a'b'c'$, then the output will also be flipped, i.e., $z \rightarrow z'$ (where A' is the complement of A).

Note that this is not the case for other logic functions such as AND, NOR, etc. For example, consider a three input AND gate with inputs 100 and output 0. If the inputs are flipped to 011, then the output will remain 0.

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

Property 3: Consider an MV with input pattern abc (for lines A, B , and C , respectively). The stuck-at- v fault on any input or output line of the voter is detectable (the fault effect appears at the output line) by abc if and only if the stuck-at- v' fault on that line is detectable by $a'b'c'$.

Proof: Consider l stuck-at- v fault. If l is an input line, consider l to be A without loss of generality. The fault is detected if and only if the value of a is v' and the other inputs (b and c) have opposite values. As a result, a' is v and b' and c' have opposite values. Hence, $a'b'c'$ detects the stuck-at- v' fault for l .

Again, this property does not hold for other logic functions. As an example, consider a two-input AND gate with test vector 11, which detects stuck-at-0 at both the top and bottom 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 at any inputs and/or the output of the MV, then property 3 still holds.

The interesting property of MVs is that the above properties hold for any arbitrary network of MVs and inverters.

Property 5: Consider an arbitrary network of MVs and inverters with primary input vector V . If all bits of V are flipped, $V \rightarrow V'$, all nodes in the network will be flipped.

Proof: The proof is based on induction on the level (distance) of each MV in the network from the primary inputs by forming a topological order of the MVs in the network. The step of induction is property 2.

Property 6: Consider an arbitrary network of MVs and inverters with primary input vector V . For any node n in the network, n stuck-at- u is detected by V if and only if n stuck-at- u' is detected by V' .

Proof: The proof is similar to the proof of property 5. The step of induction is property 4.

Properties 5 and 6 are very interesting and have proven unique features of a network of MVs and inverters. Based on prop-

erty 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, a 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: just one fault per each node because, if a vector V detects one stuck-at fault on that node, 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) into half.

To generate tests for detecting stuck-at faults in a network of MVs and inverters, conventional (combinational) ATPG tools can be exploited. The network of MVs and inverters is first transformed into a hierarchical gate-level netlist. Each MV is replaced by a hierarchical cell implementing the majority function. We only consider pin faults on the inputs of these hierarchical cells, which correspond to the inputs of MVs. As explained above, only half of the pin faults must be considered for test generation.

IV. DEFECT CHARACTERIZATION

Here, the robustness of the QCA MVs and binary wires, as well as for additional QCA circuits, is investigated. The basic functionality of a QCA device is based on the Coulombic interaction among neighboring QCA cells (depending on the accuracy and geometry of its implementation). Various configurations of QCA devices have been studied using the QCADesigner¹ v1.20 simulation tool. For accuracy, the bistable model is employed. This is a quantum mechanical engine using the Jacobi algorithm to calculate the eigenvalues/vectors of the Hamiltonian matrix.

A. Defect and Failure Modes

To perform a defect characterization of QCA devices and circuits and study their effects at the logic level, appropriate defect mechanisms and models must be considered, which: 1) can be simulated using the available simulation tool and 2) can be realistic to reflect manufacturing and fabrication defects. According to [19], in the present stage of QCA manufacturing, defects are possible in both the synthesis and deposition phases. Manufacturing defects may cause a cell to have missing or extra dots or/and electrons. This will be fatal to the correct operation of the cell. However, defects are much more likely to occur in the deposition phase than in the synthesis phase, thus resulting in a cell misplacement. A missing dot (or additional dot) is very unlikely due to the ease of purification of small inorganic molecules [19]. Moreover, electrochemical measurements for the Creutz-Taube ion (a model QCA cell) show that less than one molecule in 10^5 are in the incorrect charge state, yet placing the individual cells in specific locations during the deposition phase is diffi-

¹QCADesigner is the product of an ongoing collaboration between the Advanced Technology Information Processing Systems (ATIPS) Laboratory, University of Calgary, Calgary, AB, Canada, and the University of Notre Dame, Notre Dame, IN.

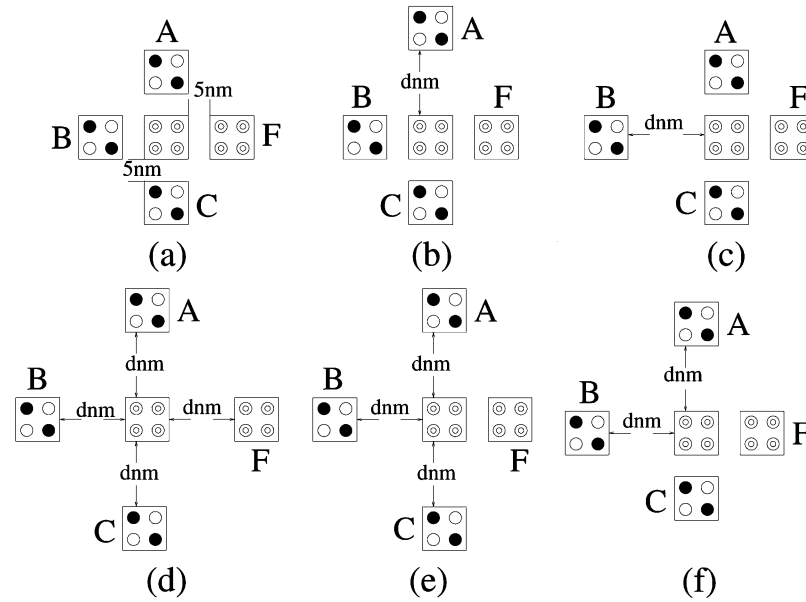


Fig. 7. Displacement in MV. (a) Fault free. (b) Displace A. (c) Displace B. (d) Displace all inputs and output. (e) Displace all inputs. (f) Displace A and B.

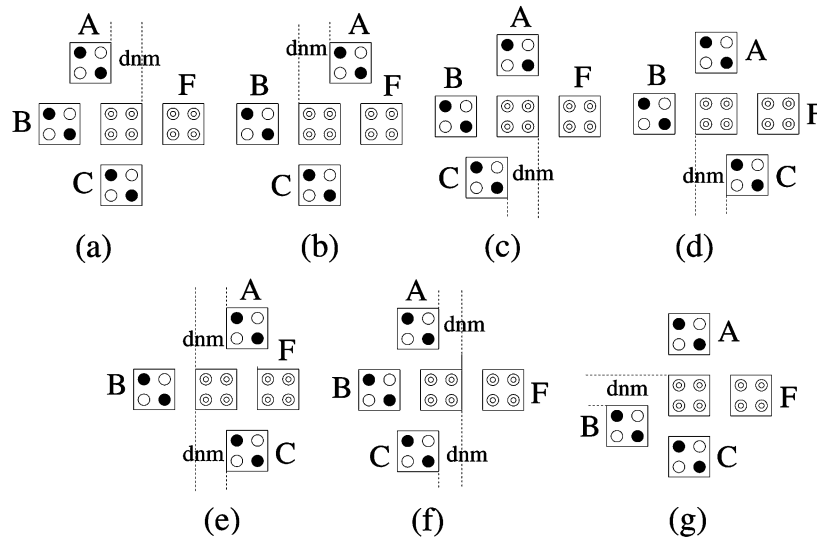


Fig. 8. Misalignment in MV. (a) A misalignment. (b) A misalignment. (c) C misalignment. (d) C misalignment. (e) A, C misalignment. (f) A, C misalignment. (g) B misalignment.

cult. Various types of cell misplacement faults may occur such as cell misalignment, rotated cells, etc. Therefore, in this paper, we assume that all the cells are perfectly manufactured and operate correctly and study the effects of the following types of cell misplacement faults.

- A *cell displacement* is a defect in which the defective cell is misplaced from its original direction. Several cell displacement defects are shown in Fig. 7.
- In a *cell misalignment* defect, the direction of the defective cell is not properly aligned. Some examples of cell misalignments are shown in Fig. 8.
- In a *cell omission* defect, a particular cell is missing as compared to the original (defect-free) arrangement. The electron missing defect in which the defective cells have no electrons can be modeled by this type of defects.

In this study, the following defects are considered and simulated for QCA devices: all possible combinations of displacement of cells with respect to the central cell under different distances, misalignment of cells in different directions. For QCA circuits, cell omission defects are also simulated.

B. MV Defect Analysis

A defect-free MV, which has a 5-nm dot size, a 20 nm \times 20 nm cell size, with a 5-nm cell distance is considered, as shown in Fig. 7(a). Different defects in the MV, including cell displacement and misalignment, have been considered and simulated. The results for cell displacement and misalignment are shown in Tables II and III, respectively. Only faulty entries are shown in the tables in the form of (fault-free/faulty) values.

TABLE II
RESULTS FOR DISPLACEMENT IN MAJORITY VOTER

displace cell A: fig 7.b			
$d \leq 15nm$ Normal Operation		$d \geq 20nm$, F=B	
displace cell B: fig 7.c			
$d \leq 40nm$		$d \geq 45nm$	
Normal Operation		A B C	F
		001	Z (no polarization)
		011	Z (no polarization)
		100	Z (no polarization)
		110	Z (no polarization)
displace all input/output cells: fig 7.d			
$d \leq 10$ or $30 \leq d \leq 40nm$		$15 \leq d \leq 25nm$	
Normal Operation		A B C	F
$d \geq 45nm$		010	0/1
F=Z (no polarization)		101	1/0
displace all input cells: fig 7.e			
$d \leq 15$ or $d = 40nm$		$d \geq 45nm$	
Normal Operation		F=Z (no polarization)	
$20 \leq d \leq 25$ or $d = 35nm$		$d = 30nm$	
A B C	F	A B C	F
010	0/1	000	0/1
101	1/0	010	0/1
		101	1/0
		111	1/0
displace cells A and B: fig 7.f			
$d \leq 5nm$ Normal Operation		$d \geq 10nm$, F=C	

The data shows that, in most cases, the horizontal input cell (i.e., cell B) is dominant; this cell seems to have a bigger impact on the center cell than A and C. For misalignment, any single cell misalignment greater or equal to half a cell causes malfunction (i.e., fault at logic level occurs). This is conceivable as a half-cell shift results in inversion in QCA. In some cases, the fault margin is smaller. A comparison between misalignment and displacement defects illustrates that the misalignment defects have more catastrophic effects on the functionality of an MV with the same defective distance as the displacement defect.

C. Defects and Faults in a Full Adder

A QCA implementation of a full adder using three MVs and two inverters is shown in Fig. 9. The corresponding QCA layout is shown in Fig. 10, which contains 145 cells. The cells are $18 \text{ nm} \times 18 \text{ nm}$ with a dot size of 5 nm . 40 different single-cell omission defects have been simulated for this circuit.

1) *Defects in Wires and Inverter Chain:* Removing a single cell from a binary wire does not affect its functionality at the logic level, although it may result in some delay faults. However, a single cell omission in a wire implemented as an inverter chain results in an unwanted complementation at the output of the chain. Those binary wires, which change direction in the

TABLE III
RESULTS FOR MISALIGNMENT IN MAJORITY VOTER

move A toward west: fig 8.a			
$d \leq 5nm$ Normal Operation		$d \geq 10nm$, F=B	
move A toward east: fig 8.b			
$5 \leq d \leq 15nm$		$d = 20$ or $d = 30nm$	
A B C	F	Normal Operation	
001	0/1		
010	0/1	$d = 25nm$	
101	1/0	F=A	
110	1/0		
move C toward west: fig 8.c			
$d \leq 5nm$		$d \geq 10nm$	
Normal Operation		F=B	
move C toward east: fig 8.d			
$5 \leq d \leq 15nm$		$d = 20$ or $d = 30nm$	
A B C	F	Normal Operation	
010	0/1		
011	1/0	$d = 25nm$	
100	0/1	F=C	
101	1/0		
move A,C toward west: fig 8.e			
$d \geq 5nm$ F=B			
move A,C toward east: fig 8.f			
$d = 5, 20, d \geq 30nm$		$10nm \leq d \leq 15nm$	
F=B		A B C	F
$d = 25nm$		000	0/1
Normal Operation		010	0/1
		101	1/0
		111	1/0
move B toward south/north: fig 8.g			
$d \leq 5nm$		$d \geq 45nm$	
Normal Operation		A B C	F
		001	0/1
		011	1/0
		100	0/1
		110	1/0

layout (e.g., L shape) are very sensitive to defects on the corner cells. A cell omission defect at the corner cell is equivalent to an unwanted complementation fault at logic level.

2) *Defects in Wire Crossing:* In a QCA implementation, two wires (horizontal and vertical) can cross each other in the same layer (*coplanar wire crossing*). In this case, one of them is implemented as a binary wire, while the other one is implemented as an inverter chain (i.e., the cells in the other wire are rotated). In the fault-free case, the wires are unaffected by each other and can carry different signal values.

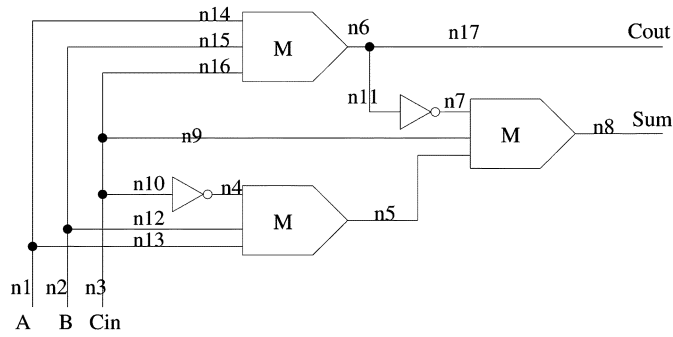


Fig. 9. 1-bit QCA full adder.

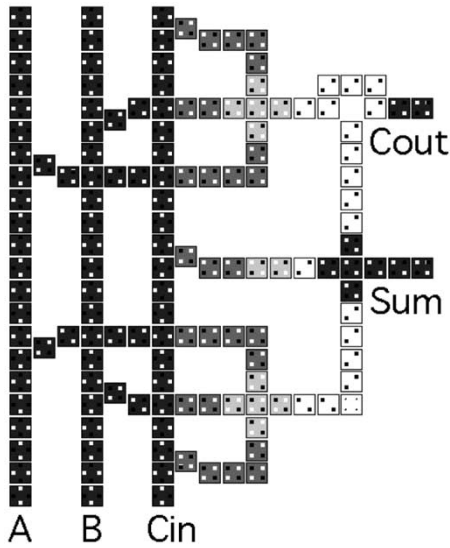


Fig. 10. 1-bit QCA full-adder layout.

However, this structure is very vulnerable to cell omission defects at or near to the cross point. The cell omission defect at the cross point results in an unwanted complementation on the inverter chain and the binary wire is *dominated* by the faulty value of the inverter chain (dominating bridging fault). Cell omission defects for the cells adjacent to the cross point have similar effects, i.e., the value of the binary wire is dominated by the faulty value of the inverter chain.

3) *Defects in the MV*: The results of defects in an MV of the full adder is consistent with the defect characterization results for a single MV: the horizontal input has more impact on the output than the vertical inputs. A cell omission defect on the horizontal input cell does not affect the functionality. However, a cell omission defect on any of the vertical inputs causes the output to be dominated only by the horizontal input, i.e., the output is shorted to the horizontal input.

The cell omission defect on the center cell of an MV with vertical input values a and b and horizontal input c changes the function to be the majority of a' , b' , and c . This can be interpreted as unwanted complementation faults on both vertical inputs.

V. TEST SET COVERAGE AND FAULT MODEL

Despite the fact that stuck-at faults do not accurately model all defects found in current CMOS technology, such a model

seems to still be the most effective in terms of detection [20]. Therefore, although from simulation results QCA defects do not behave like stuck-at faults, it is still interesting and appropriate to evaluate the effectiveness of different stuck-at test sets for the simulated defects on a single MV. The main results of this evaluation are as follows.

- In all simulations, *super exhaustive* input patterns (i.e., all possible input transitions) are used. The results show that there is no sequence-dependent behavior at the logic level, i.e., none of the manufacturing misalignments introduces a state dependency at logic level.
- Except for a single case (i.e., the displacement of all inputs and output cells), faults are detected using a subset of some 100% stuck-at fault test sets. Note that not all of these 100% stuck-at test sets are equal.
- A particular 100% two-detect stuck-at test set (each fault is detected by two vectors) can detect all manufacturing defects, except for one case, i.e., the simultaneous displacement of the top and left inputs.
- Moreover, a particular 100% single stuck-at test set (001, 010, 011, 101) can detect all simulated defects.

The results for the full-adder circuit shows that none of the defects behave as stuck-at faults at logic level. However, cell omission defects in wires implemented as inverter chains mainly result in *unwanted complementation* faults in which an additional inverter is present in the faulty wire. Cell omission defects at corner cells of the binary wires exhibit similar behavior.

We have also considered stuck-at test sets for the full adder (Fig. 9) and computed the corresponding defect coverage with respect to cell omission defects. Note that, for a full adder, any two vectors $\{(a, b, c), (a', b', c')\}$ will result in 100% stuck-at coverage for pin faults, e.g., $\{(010), (101)\}$. However, this test set can detect only 17 out of 28 cell omission defects (more over 12 of 40 simulated cell omission defects do not affect its functionality). By considering all internal nodes ($n1-n17$ in Fig. 9), $\{000\ 001\ 011\ 100\ 101\}$ is a 100% single stuck-at test set. This test set can detect all 28 detectable defects. This shows that the specific QCA implementation must be considered for test generation to achieve high defect coverage.

VI. QCA DESIGN FOR TESTABILITY

A. Test Set for MVs

Consider the simple AND–OR structure shown in Fig. 11(a) and a possible implementation using MVs in Fig. 11(b). Note that there is no built-in power supply or ground lines in quantum-dot-based designs. There are two extra inputs connected to logic “1” and logic “0” to connect some selected inputs of MV to implement AND and OR logic functions. We refer to these inputs as the control lines. The input line of the MV, which is connected to a control line, is called the control input (the control line is a fanout stem and the control inputs of MVs are fanout branches connected to the control line). The other inputs are called noncontrol inputs.

The exhaustive testing of the circuit in Fig. 11(a) needs all eight combinations of the three inputs. The minimum test set with 100% SSF coverage for this circuit contains four vectors.

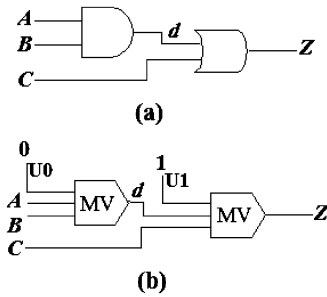


Fig. 11. (a) AND-OR circuit. (b) Implementation by MVs.

These vectors are $ABC = (010, 100, 101, 110)$, The fault list is $A/1$ (A stuck-at 1), $A/0, B/1, B/0, C/1, C/0, d/1, d/0, Z/1$, and $Z/0$. However, 100% stuck-at coverage for the same fault list contains only two vectors for the implementation using MVs, shown in Fig. 11(b). These vectors are $(ABCU_0U_1) = (11100, 00011)$. In the first test vector, both control inputs U_0 and U_1 are connected to 0. This vector detects $A/0, B/0, C/0, d/0$, and $Z/0$. The second input connects all control inputs to 1 and sets the primary inputs A, B , and C to 0. Therefore, the MVs implement OR functions. This vector detects all stuck-at-1 faults, namely, $A/1, B/1, C/1, d/1$, and $Z/1$. This reduced test set is achievable due to the specific features of the MV network and extra controllability offered by the control inputs.

Note that any 100% stuck-at coverage test set for the original circuit of Fig. 11(a) detects no stuck-at faults on the control lines of MVs, namely, $U_0/1, U_0/0, U_1/1, U_1/0$. This includes all test sets generated prior to mapping the design into MVs, and the above pair of vectors. Testing of a control line of MV for stuck-at faults requires that the two other inputs of MV 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 noncontrol inputs of each MV have different values, then all stuck-at faults on the control lines can be detected by only two test vectors.

In the above example, the following two vectors must be added to the test set to detect control line faults: $(ABCU_0U_1) = (10011, 01100)$. The first (second) vector detects stuck-at-0 (stuck-at-1) faults on the control lines. Note that the noncontrolling inputs of each MV have opposite values in each test vector.

Now consider a more complex example, as shown in Fig. 12(a), with a possible implementation by QCA MVs in Fig. 12(b). This network requires at least seven test vectors for 100% SSF coverage. However, the circuit in Fig. 12(b) requires only two vectors to achieve 100% fault coverage for the same fault list (all stuck-at faults on nodes $A, B, C, D, E, F, g, h, i, j, Z$). These two vectors are $(ABCDEFU_0U_1) = (0000011, 1111100)$. In this case, testing for stuck-at faults on control lines cannot be accomplished by two test vectors, as in the previous example. It is not possible to simultaneously set the noncontrol inputs of MV1, MV2, and MV3 to opposite values (i.e., AB, CD , and gh)

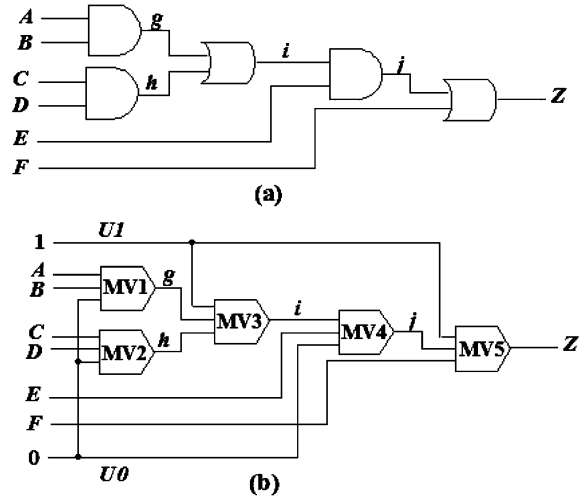


Fig. 12. (a) Network of AND-OR (b) implementation by MVs.

because the control inputs of MV1 and MV2 are connected to the same control line. This results in more than two test vectors for detecting stuck-at faults on all control inputs and control lines.

B. General Test Set for a Network of MVs

Consider a logic network (composed only of AND and OR gates), which is implemented using QCA MVs. The general case in which NOT gates also used (universal set) will be covered in Section VI-C. The QCA network has two additional control inputs other than the primary inputs. One control input, which is connected as one of the three inputs to all MVs implements the OR function, and is connected to "1." The other control input, which is feeding all MVs implementing the AND function, is connected to "0." Only two test vectors are needed for the MV implementation to detect all SSFs in the fault list of the original design.

The first test vector sets all primary inputs to "0" and two control inputs to "1." The second test vector sets all primary inputs to "1" and the control inputs to "0".

Lemma: The first test vector, as described above, detects all stuck-at-1 faults, while the second vector detects all stuck-at-0 faults in the fault list of the original design.

Proof: Since, in the first vector, the control inputs are set to "1," then all MVs implement an OR function. Therefore, by applying "0" on all primary inputs, any stuck-at-1 fault on any node produces an incorrect "1" on the primary output(s) and the fault will be detected. The second vector is the dual of the first one, in which all MVs implement an AND function. By applying a "1" on all primary inputs, an incorrect "0" on any node will produce an incorrect 0 at the primary output and the fault will be detected. Q.E.D. This implies that MV-based circuits are also C testable.

Note that these two test vectors guarantee 100% stuck-at fault coverage in the fault list of the original design. However, they do not detect stuck-at faults on the control inputs (these inputs do not exist in the original network and only exist in the MV-based implementation). To detect stuck-at faults on the control lines

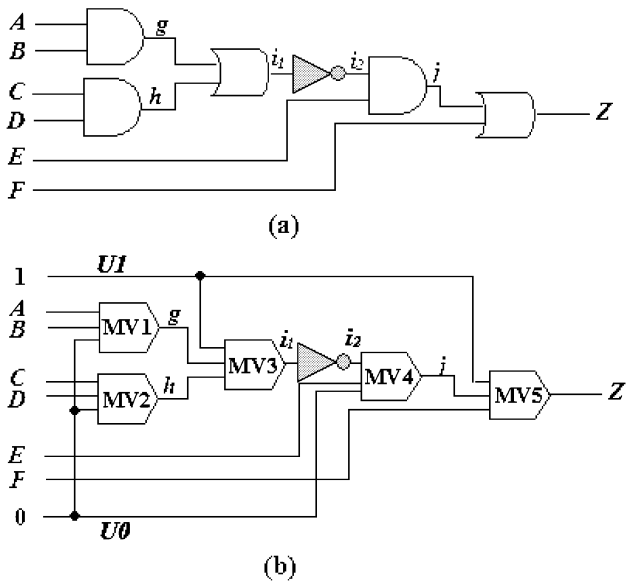


Fig. 13. Effect of inverters on masking.

and control inputs, conventional (combinational) ATPG tools can be used to generate tests for these faults. 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. We only consider the pin faults on the inputs of these hierarchical cells (corresponding to the control inputs of MVs).

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 also the number of test vectors.

C. Design for Testability

In Section VI-B, test generation for logic networks composed of AND and OR functions and implemented by MV QCAs were considered. Since the universal logic contains inverters, as well as AND and OR, this general case is considered here.

Inside the logic network, inverters prevent propagation of all faults by the proposed two test vectors. As a result, some faults remain undetected and additional test vectors are needed to detect them. This is shown through an example. The circuit shown in Fig. 13 is the same as Fig. 12, except an inverter is added on node i . The test vector $(ABCDEFU_0U_1) = (0000011)$, which detects all stuck-at-1 faults on all nodes in Fig. 12, cannot detect $E/1$ and $F/1$ (it detects $j/0$ and $Z/0$ instead of $j/1$ and $Z/1$). Moreover, unlike Fig. 12, the presence of some stuck-at faults (for multiple faults) prevents the detection by this vector, e.g., $g/1$ and $E/1$.

To use the test vector pair presented in Section VI-B, DeMorgan's law can be exploited to change the structure of the design such that all inversions are pushed back to the primary input level. It is always possible to transform a general logic network of AND, OR, and NOT functions (universal logic) into an equiv-

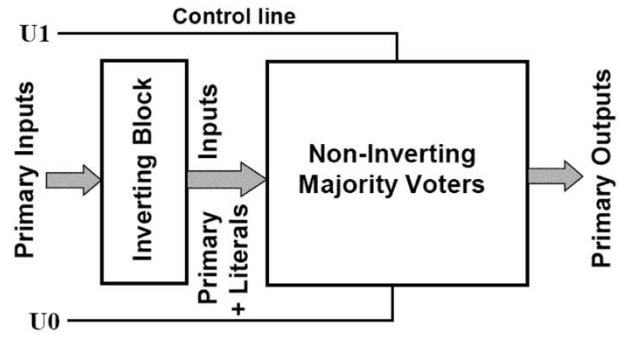


Fig. 14. Architecture with separate blocks for inverters and MVs.

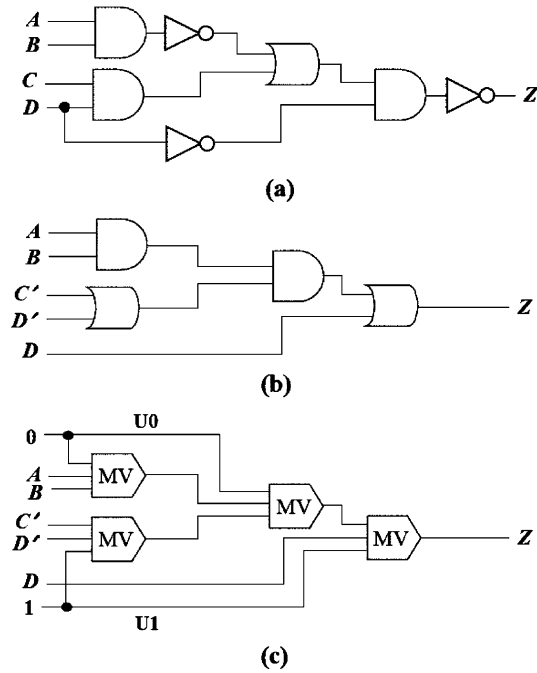


Fig. 15. (a) Logic network of AND, OR, and NOT. (b) Equivalent network of AND, OR and literals. (c) Implementation by MVs.

alent 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 in Section VI-B can be still exploited. The structure of these designs is shown in Fig. 14. The design is partitioned into two blocks: the first block is the inverting block, which generates the literals, followed by a block of MVs, which implement the AND-OR network. Note that the inverting block can be very efficiently implemented in QCA using only inverter chains (Fig. 3) because the number of cells and the area of the inverter chain are the same as for a binary wire. Therefore, 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). An example of this transformation is shown in Fig. 15. Fig. 15(a) shows an example of a logic network of AND, OR, and NOT functions. In Fig. 15(b), the equivalent logic is shown such that the inverters are pushed back into the input level. The corresponding MV implementation is shown in Fig. 15(c).

VII. CONCLUSION

QCA are novel devices that are promising in the era of nanoscale computing. In this paper, testing of QCA-based designs has been investigated. A detailed defect characterization of QCA basic logic devices and some representative circuits has been presented. As shown in this paper, the QCA behavior of defects at the logic level (i.e., faults) are not similar to those in a conventional CMOS fabrication process. For example, an *unwanted complementation* fault at logic level has been observed for a considerable number of cases of *cell omission* defects. Hence, appropriate fault models for QCA must be developed and used for test generation.

The effectiveness of different stuck-at test sets for detecting QCA defects has been studied. Our results show that, to achieve a high defect coverage, the specific QCA implementations of each function must be considered for test generation. Some interesting and unique properties of QCA implementations of logic networks have been investigated. As shown in this paper, a network of MVs (and inverters) has unique testing properties: any (V, V', V) test set achieves 100% toggle fault coverage and V detects n stuck-at- u if and only if V' detects n stuck-at- u' . C testability of QCA designs has been investigated. MV-based designs offer C testability in both 1-D and 2-D structures. A design-for-test scheme has been presented based on a change in the structure of the design; the design is partitioned into a block of inverters and a block of MVs. This structure results in reduced test generation and test length.

ACKNOWLEDGMENT

The authors would like to thank K. Walus, ATIPS Laboratory, University of Calgary, Calgary, AB, Canada, and Prof. M. Lieberman, Department of Chemistry and Biochemistry, University of Notre Dame, Notre Dame, IN, for advice and help with this study. The authors also wish to thank the reviewers of this paper for their insightful comments.

REFERENCES

- [1] A. O. Orlov, I. Amlani, G. H. Bernstein, C. S. Lent, and G. L. Snider, "Realization of a functional cell for quantum-dot cellular automata," *Science*, vol. 277, pp. 928–930, 1997.
- [2] P. Tougaw and C. Lent, "Logical devices implemented using quantum cellular automata," *J. Appl. Phys.*, vol. 75, no. 3, pp. 1818–1825, 1994.
- [3] P. D. Tougaw and C. S. Lent, "Dynamic behavior of quantum cellular automata," *J. Appl. Phys.*, vol. 80, no. 8, pp. 4722–4736, 1996.
- [4] K. Hennessy and C. S. Lent, "Clocking of molecular quantum-dot cellular automata," *J. Vac. Sci. Technol.*, vol. 19, no. 5, pp. 1752–1755, 2001.
- [5] S. E. Frost, A. F. Rodrigues, A. W. Janiszewski, R. Rausch, and P. M. Kogge, "Memory in motion: A study of storage structures in QCA," presented at the 1st Non-Silicon Computation Workshop, 2002.
- [6] M. T. Niemier, A. F. Rodrigues, and P. Kogge, "A potentially implementable FPGA for quantum dot cellular automata," presented at the 1st Non-Silicon Computation Workshop, 2002.
- [7] M. Niemier and P. M. Kogge, "Logic-in-wire: Using quantum dots to implement a microprocessor," presented at the Electronics, Circuits, and Systems Int. Conf., 1999.
- [8] V. S. Dimitrov, G. A. Jullien, and K. Walus, "Quantum-dot cellular automata carry-look-ahead adder and barrel shifter," presented at the IEEE Emerging Telecommunications Technologies Conf., 2002.

- [9] K. Walus, A. Vetteth, G. A. Jullien, and V. S. Dimitrov, "RAM design using quantum-dot cellular automata," in *Nanotechnology Conf. and Trade Show*, vol. 2, 2003, pp. 160–163.
- [10] I. Amlani, A. O. Orlov, G. L. Snider, and C. S. Lent, "Demonstration of a six-dot quantum cellular automata system," *Appl. Phys. Lett.*, vol. 72, pp. 2179–2181, 1998.
- [11] I. Amlani, A. Orlov, G. Toth, C. Lent, G. Bernstein, and G. L. Snider, "Digital logic gate using quantum-dot cellular automata," *Science*, vol. 284, pp. 289–291, 1999.
- [12] R. Kummamuru, A. Orlov, C. Lent, G. Bernstein, and G. Snider, "Clocked quantum-dot cellular automata circuits," in *Nanotechnology Conf. and Trade Show*, vol. 2, 2003, pp. 149–152.
- [13] M. Lieberman, S. Chellamma, B. Varughese, Y. Wang, C. Lent, G. Bernstein, G. Snider, and F. Peiris, "Quantum-dot cellular automata at a molecular scale," *Ann. New York Acad. Sci.*, vol. 960, pp. 225–239, 2002.
- [14] C. Lent, B. Isaksen, and M. Lieberman, "Molecular quantum-dot cellular automata," *J. Amer. Chem. Soc.*, vol. 125, pp. 1056–1063, 2003.
- [15] C. D. Armstrong, W. M. Humphreys, and A. Fijany, "The design of fault tolerant quantum dot cellular automata based logic," presented at the 11th NASA VLSI Design Symp., 2003.
- [16] C. D. Armstrong and W. M. Humphreys, "The development of design tools for fault tolerant quantum dot cellular automata based logic," presented at the 2nd Int. Quantum Dots for Quantum Computing and Classical Size Effect Circuits Workshop, 2003.
- [17] A. Fijany and B. Toomarian, "Bouncing threads: Merging a new execution model into a nanotechnology memory," in *IEEE Computer Soc. Annu. VLSI Symp.*, vol. 3, 2001, pp. 27–37.
- [18] R. Ravichandran, N. Ladiwala, J. Nguyen, M. Niemier, and S. Kim, "Automatic cell placement for quantum-dot cellular automata," presented at the Great Lake VLSI Symp., 2004.
- [19] M. Lieberman, private communication.
- [20] E. McClusky and C. Tseng, "Stuck-fault tests vs. actual defects," in *Int. Test Conf.*, 2000, pp. 336–343.



Mehdi B. Tahoori received the B.S. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2000, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 2002 and 2003, respectively.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA. Prior to joining Northeastern University, he was involved with advanced computer-aided design (CAD) research with the Fujitsu Laboratories of America, during which time he worked on noise analysis for deep submicrometer mixed-signal designs. He is also involved with test, defect, and fault tolerance issues for emerging nanotechnologies. His research interests include very large scale integration (VLSI) testing, VLSI CAD, and fault tolerant computing.



Jing Huang received the B.S. degree in electronics engineering from Fudan University, Shanghai, China, in 2001, and is currently working toward the Ph.D. degree in electrical engineering and computer science at Northeastern University, Boston, MA.

She is currently a Research Assistant with the Electrical and Computer Engineering Department, Northeastern University. From 1999 to 2001, she was a Research Assistant with the Computer Aided Test Laboratory, Electrical Engineering Department, Fudan University. From 2001 to 2002, she was a Hardware Design Engineer with Huawei Technologies Inc., Shenzhen, China. Her research interests include testing, design for testability and fault tolerance of VLSI, reconfigurable systems, and nanotechnologies.



Mariam Momenzadeh was born in Tehran, Iran, in 1976. She received the B.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 1999, the M.Sc. degree in computer engineering and science from the University of Connecticut, Storrs, in 2003, and is currently working toward the Ph.D. degree in computer engineering at Northeastern University, Boston, MA.

Her research interests lie in design, testing, and fault tolerance issues in digital systems and

nanotechnology.



Fabrizio Lombardi received the B.Sc. degree (with honors) in electronic engineering from the University of Essex, Essex, U.K., in 1977, the Master in Microwaves and Modern Optics, Diploma in Microwave Engineering degrees from University College London, London, U.K., both in 1978, and the Ph.D. degree from the University of London, London, U.K., in 1982.

In 1977, he joined the Microwave Research Unit, University College London. He is currently the Holder of the International Test Conference (ITC)

Endowed Professorship at Northeastern University, Boston, MA. From 1998 to 2004, he was Chair of the Department of Electrical and Computer Engineering, Northeastern University. Prior to joining Northeastern University he was a faculty member with Texas Tech University, University of Colorado at Boulder, and Texas A&M University. His research interests are testing and design of digital systems, quantum computing, Automatic Test Equipment systems, configurable/network computing, defect tolerance, and CAD VLSI. He has authored and coauthored numerous publications in these areas and has edited six books.

Dr. Lombardi was an associate editor of the IEEE TRANSACTIONS ON COMPUTERS (1996–2000). He was a Distinguished Visitor of the IEEE Computer Society (IEEE CS) (1990–1993). Since 2000, he has been the Associate Editor-In-Chief of IEEE TRANSACTIONS ON COMPUTERS. He is also currently an associate editor of the *IEEE Design and Test Magazine* and a Distinguished Visitor of the IEEE CS. He is also the chair of the Committee on Nanotechnology Devices and Systems of the Test Technology Technical Council of the IEEE.