Test Generation for Combinational Quantum Cellular Automata (QCA) Circuits

Pallav Gupta, Niraj K. Jha, and Loganathan Lingappan
Dept. of Electrical Engineering
Princeton University
Princeton, NJ 08544
{pgupta|jha|llingapp}@ee.princeton.edu

***Abstract*— In this paper, we present a test generation framework for testing of quantum cellular automata (QCA) circuits. QCA is a nanotechnology that has attracted significant recent attention and shows immense promise as a viable future technology. This work is motivated by the fact that the stuck-at fault test set of a circuit is not guaranteed to detect all defects that can occur in its QCA implementation. We show how to generate additional test vectors to supplement the stuck-at fault test set to guarantee that all simulated defects in the QCA gates get detected. Since nanotechnologies will be dominated by interconnects, we also target bridging faults on QCA interconnects. The efficacy of our framework is established through its application to QCA implementations of MCNC benchmarks that use majority gates as primitives.**

I. Introduction

For two decades, bulk complementary metal-oxide semiconductor (CMOS) technology has consistently provided the required dimension scaling for implementing high-density, high-speed, and low-power very large scale integrated (VLSI) systems. However, such an aggressive scaling is accompanied by many problems. These problems are, to name a few, high leakage current, high power density levels, and very high lithography costs. It is predicted that these issues will result in the conclusion of the CMOS revolution within the next 10-15 years [1]. The short time remaining exerts the need for active research in novel nanoscale technologies to help determine viable alternatives.

One possible replacement technology candidate that has been proposed is quantum cellular automata (QCA) [2]–[5] in which logic states, rather than being encoded as voltage levels, as in conventional CMOS technology, are represented by the configuration of an electron pair confined within a quantum-dot cell. QCA promises small feature size and ultra-low power consumption. It is believed that a QCA cell of a few nanometers can be fabricated through molecular implementation by a self-assembly process [6]. If this does hold true, then it is anticipated that QCA can achieve densities of 10^{12} devices/cm^2 and operate at THz frequencies [7].

Since its initial proposal, QCA has attracted significant attention. Consequently, various researchers have addressed different computer-aided design (CAD) problems for QCA. In [8], the authors have developed a majority logic synthesis tool for QCA called MALS. Majority logic synthesis has also been addressed in [9]. In the SQUARES formalism [10], some basic QCA components are described that can be connected arbitrarily. This allows a designer to lay out a QCA circuit systematically with standardized components. Models to minimize layout area and dead space are presented in [11]. These address the main shortcomings of the SQUARES formalism. A tool called QCADesigner for manual layout of QCA circuits has been presented in [12]. This tool also offers various simulation engines, each offering a trade-off between speed and accuracy, to simulate the layout for functional correctness. The authors of [7], [13] characterize in detail the types of defects that are most likely to occur in the manufacturing of QCA circuits.

The main question facing the testing community is whether the conventional CMOS test flow will have to be radically altered in order to accommodate testing of nanotechnologies, such as QCA. In testing of CMOS circuits, test generation is usually done using the single stuck-at fault (SSF) model. In this model, a line in a circuit is assumed to be permanently stuck-at-one (SA1) or stuck-at-zero (SA0). It has been observed that the SSF test set can detect a large proportion (80-85%) of all manufacturing defects in CMOS even though the SSF model does not fully capture the behavior of all such defects [14]. For those defects that remain uncovered, additional test generation is performed using other fault models. To test for defects on the interconnect, the bridging fault model is commonly used in which a pair of lines is assumed to be shorted. A similar approach is also applicable to testing of QCA circuits, as we will see later.

This work addresses the problem of automatic test pattern generation (ATPG) for combinational QCA circuits. Even though fault modeling has been addressed for QCA circuits before [7], [13], we are unaware of any other work that has addressed the general ATPG problem before. The main contributions of this work are as follows:

- This is the first comprehensive test generation methodology for combinational QCA circuits.
- Based on the QCA defect characterization results in [7], [13], we show how to supplement an SSF test set with additional vectors so that we can guarantee that every testable defect in modeled gates in a QCA circuit can actually be detected.
- Our test generation methodology also targets bridging faults on QCA interconnects.

The remainder of this paper is organized as follows. In Section II, we present background material. We explain, with a motivational example, why a test generation methodology for QCA is needed in Section III. Section IV describes our test generation methodology in detail. We discuss experimental results in Section V and conclude the paper in Section VI.

II. Preliminaries

In this section, we provide some background material that will be helpful in understanding the remainder of this paper.

A. A QCA cell

A QCA cell, shown in Fig. 1, contains four quantum dots positioned at the corner of a square, and two electrons that can move to any quantum dot within the cell through electron tunneling [2]. Due to Coulombic interactions, only two configurations of the electron pair exist that are stable. Assigning a polarization, P, of -1 and $+1$ to distinguish between these two configurations leads to a binary logic system.

B. QCA Gates

The fundamental logic gate in QCA is the majority gate. A three-input majority gate, M, is logic high if two or more of its inputs are logic high. That is,

$$M(A, B, C) = AB + AC + BC. \quad (1)$$

From this point onwards, a three-input majority gate will be simply referred to as a majority gate.

Computation in a QCA majority gate, as shown in Fig. 2(a), is performed by driving the device cell to its lowest energy state. This is achieved when the device cell assumes the polarization of the majority of the three input cells. The reason why the device

Acknowledgments: This work was supported by NSF under Grant No. CCF-0429745.

3-9810801-0-6/DATE06 © 2006 EDAA

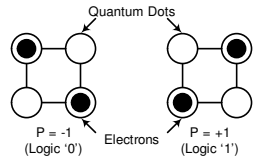


Fig. 1. A QCA cell. The logic states are encoded in the electron pair configuration.

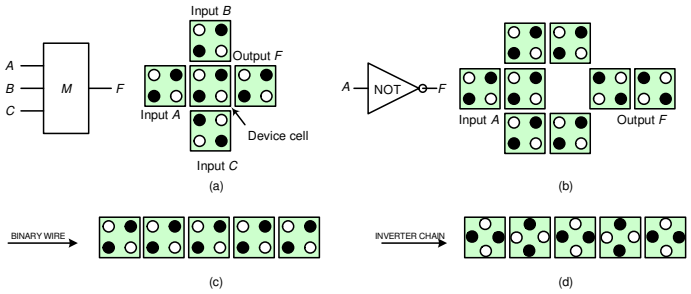


Fig. 2. A QCA (a) majority gate, (b) an inverter, (c) binary wire, and (d) inverter chain.

cell always assumes a majority polarization is because it is in this polarization state that the Coulombic repulsion between electrons in the inputs cells is minimized [2]. The polarization of the device cell is then transfered to the output cell.

The schematic diagrams of an inverter and interconnects are also shown in Fig. 2. There are several other ways of obtaining the inverting function in QCA. However, the inverter shown in Fig. 2(b) is preferred as it is very robust. In the QCA binary wire shown in Fig. 2(c), information propagates from left to right. An inverter chain, shown in Fig. 2(d), can be constructed if the QCA cells are rotated by 45°. Furthermore, it is possible to implement two-input AND and OR gates by permanently fixing the polarization of one of the input cells of a majority gate to -1 (logic 0) and $+1$ (logic 1), respectively. Finally, the majority gate and inverter constitute a functionally complete set (i.e., they can implement any arbitrary Boolean function).

C. QCA Defects

The types of defect that are likely to occur in the manufacturing of QCA devices have been investigated in [7], [13], [15] and are illustrated in Fig. 3. They can be categorized as follows:

1) In a *cell displacement* defect, the defective cell is displaced from its original position. For example, in Fig. 3(b) the cell with input B is displaced to the north by Δnm from its original position.
2) In a *cell misalignment* defect, the direction of the defective cell is not properly aligned. For example, in Fig. 3(c) the cell with input B is misaligned to the east by Δnm from its original position.
3) In a *cell omission* defect, the defective cell is missing as compared to the defect-free case. For example, in Fig. 3(d) the cell with input B is not present.

For a QCA majority gate, all possible combinations of displacement of cells with respect to the center cell under different distances and misalignments in different directions were simulated [13] using QCADesigner [12]. It was reported that these defects result in different logic values in the fault-free (defect-free) and faulty (defective) cases. In addition, it was mentioned that a particular SSF test set could detect all of the simulated defects [7]. However, a majority gate has nine possible minimal SSF test sets and the question as to which of these tests sets can detect all of the simulated defects was left unanswered. We will answer this question in this work when we discuss our ATPG framework.

For QCA interconnects, cell displacement and omission defects on binary wires and inverter chains were also considered in [15]. These

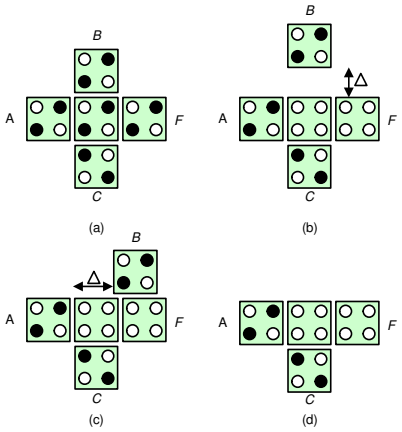


Fig. 3. (a) Defect-free majority gate, (b) displacement defect, (c) misalignment defect, and (d) omission defect.

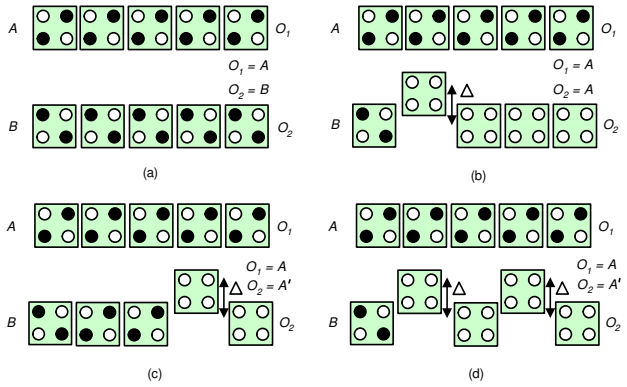


Fig. 4. (a) Fault-free binary wire and (b)-(d) binary wire with a bridging fault.

defects could be better modeled using a dominant fault model in which the output of the dominated wire is determined by the logic value on the dominant wire. There are many possible scenarios that could occur in the presence of a bridging fault and are illustrated in Fig. 4. In the first scenario shown in Fig. 4(b), the second cell is displaced to the north from its original position by Δnm. In this case, the dominated wire O_2 will have a logic value equal to that of the dominant wire O_1. However, if the fourth cell is displaced, as shown in Fig. 4(c), then O_2 will have a logic value equal to the complement of the logic value on O_1 (i.e., $O_2 = O_1'$). Finally, if multiple cells are displaced, as shown in Fig. 4(d), then O_2 will also equal O_1'.

It was also shown that binary wires are more defect-tolerant than inverter chains. Inverter chains are typically used to implement coplanar wire crossings. In such a wire crossing, a wire running along the horizontal or vertical direction is a binary wire, while the second wire is an inverter chain running in the perpendicular direction. However, it has been shown that circuits containing coplanar wire crossings are very likely to fail [16]. A multi-layer wire crossover scheme has been proposed to remedy this problem.

D. Test Generation Basics

Fault activation and fault effect propagation are the key elements of test generation. Fault activation is the process of creating a fault effect at the output of the faulty circuit element. Fault effect propagation is the process of assigning values to circuit lines such that the fault effect propagates from the output of the faulty circuit element to a primary output of the circuit [14].

Given a test vector T, the process of fault simulation of a circuit C derives all the faults in C detected by T. Two faults f_i and f_j in a circuit C are said to be equivalent if the corresponding faulty versions of the circuit, C_{f_i} and C_{f_j}, respectively, have identical input-output

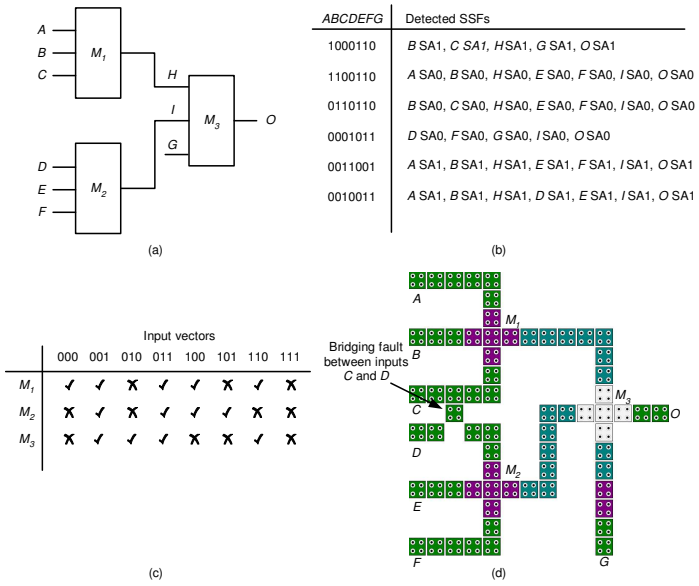


Fig. 5. (a) Example majority circuit, (b) minimal SSF test set, (c) input vectors received by each majority gate, and (d) QCA layout of the circuit with a bridging fault between inputs C and D.

logic behavior. Consequently, one of the two faults can be dropped from the fault list. Fault f_i is said to dominate f_j if the set of vectors that detects f_i is a superset of the set of vectors that detects f_j. Consequently, f_i can be dropped from the fault list [14].

E. SAT-based Test Generation

Boolean satisfiability (SAT) based formulations are commonly used for test generation [14]. In SAT, a satisfying assignment on the variables of a Boolean function, $g(x_1, x_2, \ldots, x_n)$, is sought such that g evaluates to 1. If no such assignment can be found, then g is said to be unsatisfiable. Typically, g is represented in conjunctive normal form (CNF) which is a conjunction of disjunctive clauses. A disjunctive clause is a disjunction of one or more literals where a literal is a variable or its complement.

SAT-based test generation [17] is performed by constructing the CNF of the fault-free and faulty circuits. If a SAT solver can find a satisfying assignment on the variables of the CNF formed by concatenating the CNF of the fault-free and faulty circuits, the values assigned to the inputs of the circuit form a test vector for that particular fault. If an assignment cannot be found, the fault is proved to be redundant and is untestable.

III. Motivational Example

In this section, we present an example to motivate the need for our testing methodology for QCA circuits.

Consider the majority circuit shown in Fig. 5(a) which contains three majority gates (M_1-M_3), seven primary inputs (A-G), and one primary output (O). Since there are ten lines in this circuit, there are 20 SSFs. However, we only need to target SA0/SA1 faults at the primary inputs in this circuit to guarantee detection of all 20 SSFs, as we will see in Section IV. Since there are seven primary inputs, there are 14 SSFs that need to be targeted during test generation. Fig. 5(b) shows a minimal SSF test set for the circuit. FSIM [18] was extended to perform fault simulation on the majority circuit using the generated test set. For each test vector in Fig. 5(b), all the SSFs detected by it are also shown.

Given the test set in Fig. 5(b), Fig. 5(c) shows the input vectors that the individual majority gates of the circuit receive in the fault-free case. For example, it can be seen that M_2 receives input vectors 001, 011, 100, and 101. Now, consider fault I SA0 which is detected by three test vectors, namely 1100110, 0110110, and 0001011. Given the application of vectors 1100110 and 0110110, M_3 will receive input vector 110 in the fault-free case. If vector 0001011 is applied, M_3 will receive input vector 011. In all these cases, the expected fault-free logical value at output O is logic 1. However, in the presence of I SA0, output O become logic 0, thus detecting the fault.

It can be seen from Fig. 5(c) that M_1 receives input vectors 000, 001, 011, 100 and 110. Even though these vectors form a *complete* SSF test set for a majority gate, they are *not* a complete test set for detecting *all* simulated defects that were presented in [7], [13], [15]. QCADesigner [12] was used to perform defect simulation and confirm this assertion. In fact, five defects (three misalignments and two displacements on the QCA cells in M_1) cannot be detected by these input vectors. If we add vector 0100110 to our original test set, then M_1 will also receive input vector 010. Note that the effect of the last four of the above five vectors is also propagated from the output of M_1 to output O. This is a now a complete test set for all simulated defects in M_1. Gates M_2 and M_3 already receive input vectors which form a complete defect test set and hence, require no additional test vectors (the effect of these vectors is also propagated to output O).

Fig. 5(d) shows a possible QCA layout for the circuit in Fig. 5(a) (different shades in QCA cells indicate different phases of a four-phase clock that is typically used for QCA circuits [6]). Consider the QCA cell displaced from the binary wire of input D so that there exists a bridging fault between inputs C and D. In this case, C dominates D. Furthermore, it is unknown whether the bridging fault will result in $D = C$ or $D = C'$. To be able to detect this defect, we need vectors to test for two of four possible conditions. Condition D SA0 with $C = 0$ or D SA1 with $C = 1$ must be tested and condition D SA1 with $C = 0$ or D SA0 with $C = 1$ must be tested. We will explain how we obtained these conditions in the next section. The first and second conditions can be tested by vectors 0001011 and 0010011 that are already present in our test set. However, we currently have no vector that can test either of the latter two conditions. Therefore, additional test generation is required and vectors 0000011 and 0011011 are derived as tests for the third and fourth conditions, respectively. Adding either of these two vectors is sufficient as we only need to satisfy either the third or fourth condition for fault detection. The bridging fault between C and D, when C is dominant and D is dominated, can now be completely tested for all simulated defects.

The above example illustrates that the SSF test set of a circuit cannot guarantee the detection of all simulated defects in the QCA majority gates. Therefore, test generation will be required to cover the defects not covered by the SSF test set. In addition, test generation will also be needed to cover bridging faults on QCA interconnects.

IV. ATPG for QCA Circuits

In this section, we present our ATPG methodology for combinational QCA circuits.

A. Fault Collapsing for Majority Gates

In general, there are no fault equivalence relationships between the inputs and outputs of a majority gate. However, an output SA0 (SA1) fault dominates an input SA0 (SA1) fault. Consider the first case. A vector is a test for an input SA0 fault in a majority gate if and only if the fault-free and faulty values at the output are 1 and 0, respectively. This is because when testing for an input SA0, the other two inputs must have values 0 and 1, respectively, or vice versa in order to enable the propagation of the fault through the gate. This is evident from the definition of a majority gate given in Equation (1). Thus, any vector that detects an input SA0 fault will also detect an output SA0 fault. A similar situation holds for an input SA1 fault.

The above observation leads to the following theorems for testing of irredundant majority circuits.

Theorem 1: In an irredundant, fanout-free combinational majority circuit, C, any test set, $\mathcal{V}$, that detects all SSFs on the primary inputs detects all SSFs in C.

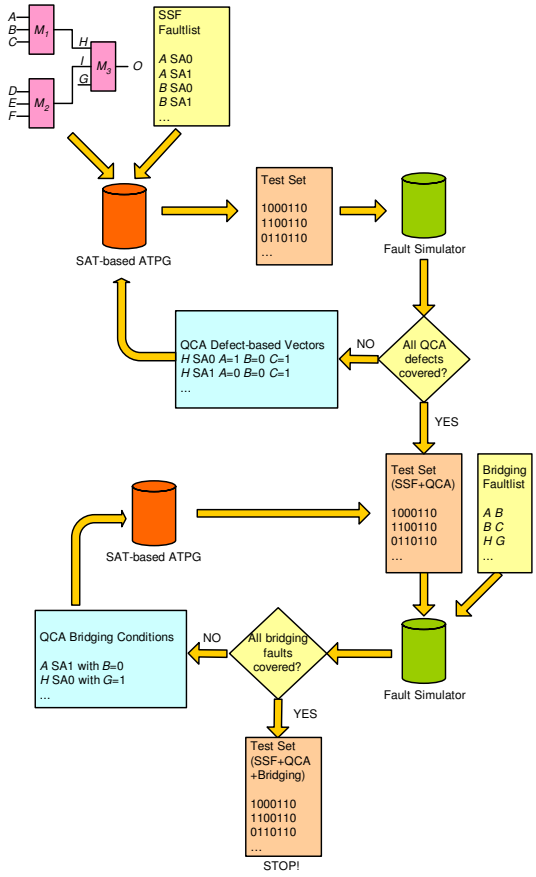


Fig. 6. A high-level overview of our ATPG approach for combinational QCA circuits.

Proof: The proof is based on fault dominance and has been omitted due to space limitations. ■

Theorem 2: In an irredundant combinational majority circuit, C, any test set, $\mathcal{V}$, that detects all SSFs on the primary inputs and fanout branches detects all SSFs in C.

Proof: This proof is also based on fault dominance and has been omitted due to space limitations. ■

B. High-level Overview

Fig. 6 gives a high-level overview of our approach. The input to our methodology is a logic-level circuit description consisting of majority gates and inverters only and the output is a test set which can detect all simulated defects in the gates and interconnects in the corresponding circuit. Initially, we perform SAT-based test generation for the logic-level circuit by targeting all SSFs on the primary inputs and fanout branches of the circuit. The targeted faults are derived based on the fault collapsing theorems for majority logic presented in the previous section. SAT-based test generation requires the CNF of a majority gate which is given below (A, B, and C are its inputs and F its output):

$$M_{CNF} = (\bar{A}+\bar{B}+F)(\bar{A}+\bar{C}+F)(\bar{B}+\bar{C}+F) \\ (A+B+\bar{F})(A+C+\bar{F})(B+C+\bar{F}). \quad (2)$$

Next, fault simulation on the logic-level circuit is performed using the generated test set. The fault simulator keeps track of all the input vectors that are received by every majority gate in the circuit. If there exists a majority gate(s) which does not receive a 100% defect test set, additional test generation is performed by using SAT-based ATPG to justify and propagate the effects of the extra vectors. Once all the defects have been covered, a test set containing the SSF test set and additional QCA vectors is obtained. This new test set and a bridging fault list is given to the fault simulator to determine whether all bridging faults are covered or not. The fault simulator generates a list of those defects that remain uncovered. We supply this QCA bridging fault list for further test generation to the SAT-based ATPG engine. If the designer has the QCA layout information available, the bridging fault list can be constructed based on adjacent wires in the layout. If the QCA layout information is not available, the designer can supply all possible wire pairs assuming the worst-case scenario. Once all bridging faults have been covered, the final test set is provided as output.

C. Targeting QCA Defects in a Majority Gate

For a majority gate, there are nine minimal SSF test sets that contain four test vectors each. The minimal test sets were applied to a QCA majority gate and all defects described in [7], [13], [15] were simulated using QCADesigner [12]. Fig. 7 shows our results for this experiment. Of the nine minimal test sets, three test sets were unable to detect all the simulated defects.

Consider the majority gate M in Fig. 8 that is embedded in a larger network. Suppose that after SSF test generation for the entire circuit, it is determined that M receives input vectors 010, 011, 100 and 101. According to Fig. 7, this is not a complete defect test set as three defects remain uncovered. We can make it complete by ensuring that M also receives either 001 or 110 as its input vector. Thus, the condition given to the SAT-based ATPG would be:

$$F \text{ SA1 with } A{=}0\ B{=}0\ C{=}1$$

or

$$F \text{ SA0 with } A{=}1\ B{=}1\ C{=}0.$$

Given the above conditions, the CNF of the faulty gate would be:

$$M_{CNF_F} = (\bar{A}+\bar{B}+F)(\bar{A}+\bar{C}+F)(\bar{B}+\bar{C}+F) \\ (A+B+\bar{F})(A+C+\bar{F})(B+C+\bar{F}) \\ (\bar{F})(\bar{A})(\bar{B})(C). \quad (3)$$

or

$$M_{CNF_F} = (\bar{A}+\bar{B}+F)(\bar{A}+\bar{C}+F)(\bar{B}+\bar{C}+F) \\ (A+B+\bar{F})(A+C+\bar{F})(B+C+\bar{F}) \\ (F)(A)(B)(\bar{C}). \quad (4)$$

If a test vector can be generated for either fault, then M would be completely testable for all simulated defects.

D. Targeting QCA Defects on Interconnects

It is very important to test for defects on QCA interconnects because it is predicted that interconnects will consume the bulk of chip area in future nanotechnologies [19]. As shown in Figs. 4(b)-4(d), a displacement of the QCA cells on a wire will result in a bridging fault between the two wires. Using QCADesigner [12], it was verified that such defects can be modeled using the dominant bridging fault model [7]. However, depending upon the displacement distance, the lower wire will assume either the upper wire's logic value or its complement.

Fig. 9 shows the possible scenarios that can result if a bridging fault is present between two wires. The scenario that will occur depends upon the displacement distance Δ of the defective cell and/or the number of cells that get displaced. It also shows the conditions that must be satisfied in order to detect the particular scenario. In the first scenario in Fig. 9(a), the lower wire's logic value is equal to that of the upper wire while in Fig. 9(b), the lower wire's logic value is equal to the complement of that of the upper wire. If the first scenario occurs, then a vector is required to test for one of the two conditions shown in Fig. 9(a). Similarly, if the second scenario occurs, a vector is required to test for either of the two conditions shown in Fig. 9(b). In reality, it will not be known which scenario occurred in the presence of the defect because Δ will not be known beforehand. Consequently, we need to have vectors that test for both scenarios to be able to have a test that can completely detect this bridging fault.

SSF Test Set	100% Defect Coverage?	SSF Test Set	100% Defect Coverage?	SSF Test Set	100% Defect Coverage?
{001, 010, 011, 101}	✓	{001, 100, 101, 110}	✓	{010, 011, 100, 101}	✗ 3 Uncovered Defects
{001, 011, 100, 101}	✓	{010, 100, 101, 110}	✓	{001, 010, 011, 110}	✓
{001, 010, 101, 110}	✗ 1 Uncovered Defect	{001, 011, 100, 110}	✗ 5 Uncovered Defects	{010, 011, 100, 110}	✓

Fig. 7. Minimal SSF test sets for a majority gate. Some of these test sets, however, are not a 100% defect test set for a QCA majority gate.

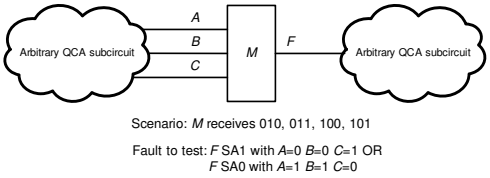


Fig. 8. Testing of defects in a QCA majority gate.

Scenario 1: Δ is such that $B = A$

Fault-free A B	Faulty A B	Equivalent Condition
0 0	0 0	-
0 1	0 0	B SA0 with A=0
1 0	1 1	B SA1 with A=1
1 1	1 1	-

Note: Assumption is that B is dominated by A.

(a)

Scenario 2: Δ is such that $B = A'$

Fault-free A B	Faulty A B	Equivalent Condition
0 0	0 1	B SA1 with A=0
0 1	0 1	-
1 0	1 0	-
1 1	1 0	B SA0 with A=1

Note: Assumption is that B is dominated by A.

(b)

Fig. 9. Modeling of bridging faults in a QCA binary wire.

As an example, consider once again the majority gate in Fig. 8. Assume there is a bridging fault between inputs A and B and it is not known whether the defective cell is on wire A or wire B. In addition, Δ is unknown. In order to test for this fault, four conditions need to be satisfied (the first two result from A dominating B, while the last two result from B dominating A). The conditions are as follows:

$$\begin{array}{lll} B \text{ SA0 with } A\text{=0} & \text{or} & B \text{ SA1 with } A\text{=1} \\ B \text{ SA1 with } A\text{=0} & \text{or} & B \text{ SA0 with } A\text{=1} \\ A \text{ SA0 with } B\text{=0} & \text{or} & A \text{ SA1 with } B\text{=1} \\ A \text{ SA1 with } B\text{=0} & \text{or} & A \text{ SA0 with } B\text{=1}. \end{array}$$

The CNFs for the above conditions are,

$$\begin{array}{lll} (CNF_B)(B)(\bar{A}) & \text{or} & (CNF_B)(\bar{B})(A) \\ (CNF_B)(\bar{B})(\bar{A}) & \text{or} & (CNF_B)(B)(A) \\ (CNF_A)(A)(\bar{B}) & \text{or} & (CNF_A)(\bar{A})(B) \\ (CNF_A)(\bar{A})(\bar{B}) & \text{or} & (CNF_A)(A)(B), \end{array}$$

where CNF_A and CNF_B are the CNFs of the gates with output A and B, respectively. If the test set contains vectors that can satisfy the above conditions, then this bridging fault can be detected. Otherwise, the fault is not completely testable.

Given a QCA circuit with n lines, there are at most $n(n-1)$ possible bridging faults involving two wires. This is based on the assumption that layout information is not available. Consequently, $2n(n-1)$ conditions will need to be satisfied in order to obtain a test set for all bridging faults. However, it should be noted that at least $n(n-1)$ (i.e., 50%) of these conditions will already be satisfied, given any complete SSF test set. This is because, given a pair of wires, when testing for a SA0/SA1 fault on one wire the other line must have a value of 0 or 1. In this work, we assume that the QCA layout of the circuit is available and the designer knows *a priori* which pairs of wires to test for bridging faults.

V. Experimental Results

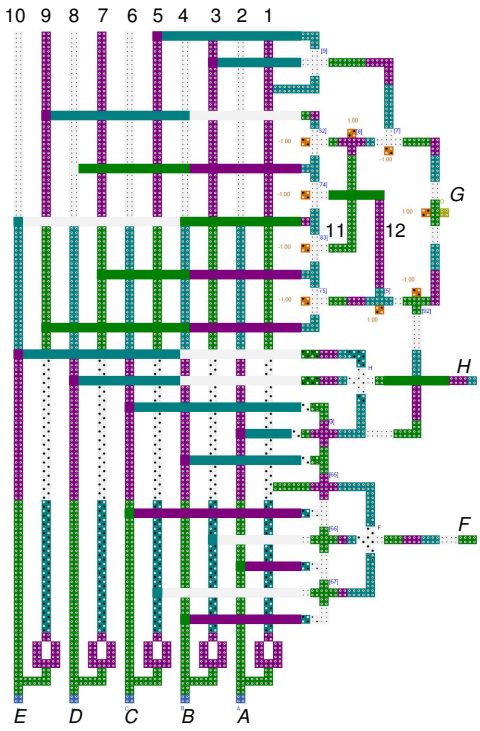
In this section, we present our experimental results on benchmarks from the MCNC benchmark suite [20]. Our methodology was implemented by extending the fault simulator FSIM [18] as a QCA defect and bridging fault simulator. The modifications amounted to approximately $1,000$ lines of C code. All of the benchmarks were first synthesized into multi-level majority networks using the logic synthesis tool MALS [8]. All experiments were conducted on a Dell PowerEdge 600SC server featuring a Pentium IV 1.6GHz processor, 512KB cache, 1GB RAM, and running Fedora Core 4.

TABLE I
Defects on QCA Majority Gates

Benchmark	No. of PI	No. of PO	No. of majority gates	SSF test set size	Additional QCA vectors	Time (s)
9symml	9	1	214	129	0	1.4
alu2	10	6	356	165	0	7.5
apex6	135	99	701	161	108	49.9
b9	41	21	137	55	24	2.2
cht	47	36	120	17	7	1.8
cm150a	21	1	46	35	0	0.1
cm151a	12	2	42	18	0	0.1
cm152a	11	1	21	18	0	0.1
cm162a	14	5	46	22	0	0.2
cm163a	16	5	42	24	0	0.2
cm82a	5	3	16	12	0	< 0.1
cm85a	11	3	34	26	0	0.1
cmb	16	4	44	39	0	0.2
count	35	16	144	46	30	2.1
cu	14	11	46	31	0	0.2
example2	85	66	259	86	41	7.5
frg1	28	3	111	90	68	0.8
i2	201	1	209	213	204	3.5
i5	133	66	132	26	6	2.5
i6	138	67	381	29	10	17.1
i7	199	67	506	36	12	27.9
i8	133	81	913	104	74	72.4
i9	88	63	559	49	25	22.6
mux	21	1	46	38	0	0.1
pcle	19	9	67	23	0	0.5
pcler8	27	17	90	41	25	0.9
ttt2	24	21	154	43	0	1.9
unreg	36	16	84	19	5	0.9
x2	10	7	42	23	0	0.2
x3	135	99	701	16	103	49.5
z4ml	7	4	27	14	1	< 0.1

Table I shows the number of primary inputs and outputs, circuit gate count, size of the SSF test set, number of additional vectors needed to cover all defects in majority gates, and CPU time for test generation. Note that the additional vectors cover all defects in majority gates only. They do not include vectors that may be needed to cover bridging faults between interconnects. The SSF test set was generated using the SAT-based ATPG engine in SIS [21]. In most of the smaller benchmarks, it can be seen that no additional vectors are required. This is probably because of the presence of fewer reconvergent fanouts, making it easier to satisfy the additional conditions. However, for larger benchmarks the opposite seems to be true, as they require additional vectors. For example, for the *apex6* benchmark, an additional 108 vectors were generated.

Rather than giving a table of results for test generation for QCA bridging faults, we present results for an example which was manually laid out. The QCA layout for the *cm82a* benchmark is shown in Fig. 10. The layout uses two layers in which the crossover wires run on different layers. The wire crossover scheme is described in [16]. The layout was implemented using QCADesigner [12] and the circuit was simulated for functional correctness. The circuit has



Possible wire pairs to test for bridging faults:

1 2	7 8	8 10	2 3	8 9	6 8	6 3
3 4	9 10	6 2	4 5	3 5	6 4	2 4
5 6	3 1	4 1	6 7	1 9	1 6	11 12

Fig. 10. QCA layout of the *cm82a* benchmark and wire pairs that need to be tested for possible bridging faults.

5 inputs, 3 outputs, 5 inverters, and 16 majority gates. Note that two-input AND and OR gates are implemented by fixing one of the inputs of a majority gate to logic 0 and logic 1, respectively. Based on the layout information, Fig. 10 also shows the wire pairs that need to be targeted for bridging fault detection. Given the bridging fault list and the SSF test set for this benchmark from Table I, we fault-simulated this circuit and determined that additional test generation was necessary to cover some undetected bridging faults. The conditions that needed to be satisfied were:

2 SA1 with 1=0	or	2 SA0 with 1=1
4 SA1 with 3=0	or	4 SA0 with 3=1
6 SA1 with 5=0	or	6 SA0 with 5=1
8 SA1 with 7=0	or	8 SA0 with 7=1
10 SA1 with 9=0	or	10 SA0 with 9=1.

It turns out that all of the above conditions were unsatisfiable. This can be explained by looking at Fig. 10. It can be seen that wire 2 carries complement values to wire 1, wire 4 carries complement values to wire 3, and so on. Thus, testing for a SA1 (SA0) fault on a line with the condition that the other line have a value of 0 (1) is not possible. Hence, for this particular example, the SSF test set will detect all the detectable bridging faults. For larger circuits, however, one can expect the need for additional test vectors to cover defects on QCA interconnects.

VI. Conclusions

In this paper, we presented the first comprehensive testing methodology for combinational QCA circuits. First, we developed some fault collapsing theorems for majority circuits. Based on the fault list derived from our fault collapsing theorems, we generated an SSF test set using a SAT-based solver. With an example, we showed that although an SSF test set detects every SSF in the circuit, it is not guaranteed to detect all the simulated QCA defects in the majority gates. Consequently, further test generation is required and we showed how to generate additional test vectors that would guarantee that all defects in the majority gates are covered.

Interconnects are likely to consume the bulk of chip area in nanotechnologies such as QCA. Therefore, we also analyzed bridging faults in QCA circuits. Based upon fault modeling results presented previously, we showed the necessary conditions that must be satisfied to detect a bridging fault between a pair of QCA wires. For a pair of wires, only two of the four conditions (one from each scenario) need to be satisfied. Multiple bridging faults are very likely to occur between interconnects, and it would be interesting to extend this work to consider them. However, it would be interesting to see if the test set derived using our methodology can detect a large proportion, if not all, of the multiple bridging faults in a given circuit. In order to implement the above methodology, a Boolean SSF simulator was modified to handle QCA defect and interconnect simulation.

References

[1] "International Technology Roadmap for Semiconductors." http://public.itrs.net

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

[3] 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, no. 17, pp. 2179–2181, Apr. 1998.

[4] I. Amlani, A. O. Orlov, G. Toth, C. S. Lent, G. Bernstein, and G. L. Snider, "Digital logic gate using quantum-dot cellular automata," *Science*, vol. 284, no. 5412, pp. 289–291, Apr. 1999.

[5] M. Lieberman, S. Chellamma, B. Varughese, Y. Wang, C. S. Lent, G. Bernstein, G. L. Snider, and F. Peiris, "Quantum-dot cellular automata at a molecular scale," *Ann. New York Acad. Sci.*, vol. 960, pp. 225–239, 2002.

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

[7] M. B. Tahoori, J. Huang, M. Momenzadeh, and F. Lombardi, "Testing of quantum cellular automata," *IEEE Trans. Nanotechnol.*, vol. 3, no. 4, pp. 432–442, Dec. 2004.

[8] R. Zhang, P. Gupta, and N. K. Jha, "Synthesis of majority and minority networks and its applications to QCA, TPL and SET based nanotechnologies," in *Proc. Int. Conf. VLSI Design*, Jan. 2005, pp. 229–234.

[9] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "A method of majority logic reduction for quantum cellular automata," *IEEE Trans. Nanotechnol.*, vol. 3, no. 4, pp. 443–450, Dec. 2004.

[10] D. Berzon and T. J. Fountain, "A memory design in QCAs using the SQUARES formalism," in *Proc. ACM Great Lakes Symp. VLSI*, Mar. 1999, pp. 166–169.

[11] N. Gergel, S. Craft, and J. Lach, "Modeling QCA for area minimization in logic synthesis," in *Proc. ACM Great Lakes Symp. VLSI*, Apr. 2003, pp. 60–63.

[12] K. Walus, T. Dysart, G. A. Jullien, and R. A. Budiman, "QCADesigner: A rapid design and simulation tool for quantum-dot cellular automata," *IEEE Trans. Nanotechnol.*, vol. 3, no. 1, pp. 26–31, Mar. 2004.

[13] M. B. Tahoori, M. Momenzadeh, J. Huang, and F. Lombardi, "Defects and faults in quantum cellular automata at nanoscale," in *Proc. VLSI Test Symp.*, Apr. 2004, pp. 291–296.

[14] N. Jha and S. Gupta, *Testing of Digital Systems*. Cambridge, United Kingdom: Cambridge University Press, 2003.

[15] M. Momenzadeh, M. B. Tahoori, J. Huang, and F. Lombardi, "Quantum cellular automata: New defects and faults for new devices," in *Proc. Int. Parallel and Distributed Processing Symp.*, Apr. 2004, pp. 207–214.

[16] K. Walus, G. Schulhof, and G. A. Jullien, "High level exploration of quantum-dot cellular automata," in *Proc. Conf. Signals, Systems, and Computers*, Nov. 2004, pp. 7–10.

[17] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Trans. Computer-Aided Design*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

[18] H. K. Lee and D. S. Ha, "An efficient, forward fault simulation algorithm based on the parallel pattern single fault propagation," in *Proc. Int. Test Conf.*, Oct. 1991, pp. 946–955.

[19] P. Beckett and A. Jennings, "Towards nanocomputer architecture," in *Proc. Asia-Pacific Comp. Sys. Arch. Conf.*, Jan. 2002, pp. 141–150.

[20] R. Lisanke, "Logic synthesis and optimization benchmarks," Microelectronics Center of North Carolina, Tech. Rep., 1988.

[21] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Sequential circuit design using synthesis and optimization," in *Proc. Int. Conf. Computer Design*, Oct. 1992, pp. 328–333.