J Electron Test (2009) 25:39–54
DOI 10.1007/s10836-008-5078-y

Detecting Multiple Faults in One-Dimensional Arrays of Reversible QCA Gates

Xiaojun Ma · Jing Huang · Cecilia Metra · Fabrizio Lombardi

Received: 15 November 2007 / Accepted: 11 June 2008 / Published online: 17 December 2008

Abstract Reversible logic design is a well-known paradigm in digital computation. In this paper, quantum-dot cellular automata (QCA) is investigated for testable implementations of reversible logic in array systems. Testability of 1D arrays consisting of reversible QCA gates is investigated for multiple faulty modules. It has been shown that fault masking is possible in the presence of multiple faults without additional lines for controllability and observability. A technique for achieving C-testability of a 1D array is introduced by adding lines for observability. By adding lines for controllability, as well as observability, the array may be fully tested with a smaller number of test patterns. Different cases of arrays made of QCA reversible gates are presented to illustrate the applicability of the proposed testing method.

Responsible Editors: C. Bolchini and Y.-B. Kim

This manuscript is an extended version of a paper presented at the 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems.

X. Ma (✉) · J. Huang · F. Lombardi
ECE Department, Northeastern University,
Boston, MA 02115, USA
e-mail: xma@ece.neu.edu

J. Huang
e-mail: hjing@ece.neu.edu

F. Lombardi
e-mail: lombardi@ece.neu.edu

C. Metra
EE Department, University of Bologna, Bologna, Italy
e-mail: cmetra@deis.unibo.it

Keywords QCA · Nanotechnology · Reversible computing · Emerging technology · Array testing

1 Introduction

Nanotechnology has been the focus of extensive research in recent years. Various novel computing systems have been proposed to supersede the projected limitations of CMOS at the end of the roadmap [4]. One of the most pressing hurdles in the development of innovative computation paradigms and systems is energy dissipation [8]. Landauer [7] pursued the relation between energy dissipation and computing at logic level and found the thermodynamic limit of computation. *Reversible computing* has then be proposed to avoid this limit. Reversible computation is accomplished at logic level by establishing a one-to-one onto mapping between the input and output states of the circuit [2]. This *bijective property* was initially investigated by Landauer who proved that the lower bound of heat dissipation for every bit of information lost in computing is $kTln2$ joules, where k is Boltzmann's constant and T is the temperature. Moreover, dissipation can be avoided if computation is carried out without losing any information [2]. Due to the bijective property, testing of reversible logic is also generally simpler than conventional irreversible logic [11]. Reversible logic gates are *information lossless*, i.e. the information output of a reversible circuit is maximized. Therefore according to [1], the probability of fault detection is maximized too. Reversible logic is inherently easier to test because the one-to-one onto property improves the controllability as well as the observability of the circuit [11].

An extensive literature exists on reversible computing [2, 13] and different gates as primitives for reversible logic computation have been proposed. In most cases, an elegant mathematical analysis of these gates has been provided as a characterization by which reversible computing (mostly at logic level) can be accomplished independently of the technology employed. However, little work has been reported on the capabilities of emerging technologies to perform reversible computation.

Quantum-dot cellular automata (QCA) is a promising emerging technology that relies on novel design concepts [9]. A QCA cell and four basic QCA logic devices are shown in Fig. 1. A QCA cell consists of four charge containers (denoted by *dots*) located at the corners of a square. Two extra mobile electrons can move in the cell by tunneling between dots. Due to Coulombic repulsion, the cell is a bi-state device with the two states shown in Fig. 1a. QCA operates by the Coulombic interaction that connects the state of one cell to the state of its neighbors. Placement of QCA cells according to a specific pattern can correlate the ground state of the system to a logic function (mapping from the states of the input QCA cells to the states of the output QCA cells). The basic logic devices for QCA are the majority voter (MV) and inverter (INV). Binary wires and inverter chains are used as interconnects for QCA circuits.

Clocking is used to modulate the inter-dot tunneling barrier of QCA cells. Using an induced electric field mechanism for clocking, true power gain is possible. A four-phase clocking scheme for QCA also known as Landauer clocking has been proposed in [8]. As shown in Fig. 2, the four phases are RELAX, SWITCH, LOCK and RELEASE. During the RELAX phase, there is no inter-dot barrier and a cell remains unpolarized. During the SWITCH phase, the inter-dot barrier is slowly raised and a cell attains a definitive polarity under the influence of its neighbors. In the LOCK phase, barriers are high and a cell retains its polarity. Finally in the RELEASE phase, barriers are lowered and a cell loses its polarity. The clock signals are commonly supplied by CMOS wires buried under the QCA circuitry. A QCA circuit is partitioned into a number of clocking zones and all cells in the same zone are controlled by a common clock signal. Clocking zones of a QCA circuit are arranged in this periodic fashion, such that zones in the LOCK phase are followed by zones in the SWITCH, RELEASE and RELAX phases.

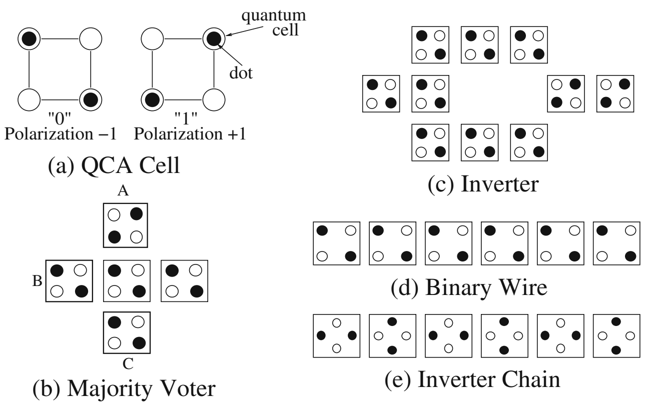


Fig. 1 QCA cell and basic QCA devices

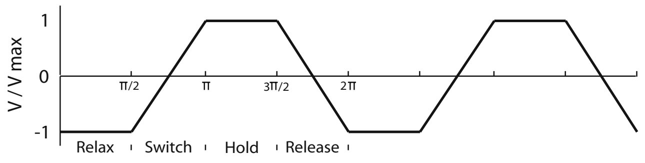


Fig. 2 Four-phased Landauer clocking

Recent developments in manufacturing have shown that the molecular implementation of QCA devices and circuits is a promising technology for QCA fabrication [6]. Unlike conventional logic circuits in which information is transferred by electrical current, QCA operates by the Coulombic interaction that connects the state of one cell to the state of its neighbors. Therefore, QCA has very low power consumption. It has been deemed as a promising technology for building reversible systems. A novel clocking scheme referred to as *Bennett clocking*, has been proposed for QCA in [8]. Bennett clocking can potentially offer a practical realization of reversible computing. It has been shown by direct calculation that with Bennett clocking, energy dissipation per switching event is much less than $kTln2$ for QCA circuits containing MV and fan-out [8].

Although it is possible in theory to design a system with a virtually zero energy dissipation by Bennett clocking, the main focus of this paper is to analyze the testability advantage of logically reversible QCA systems. In this paper, reversible circuits that are amenable to efficient testing due to their bijective property, are investigated under multiple faults. The one-to-one onto mapping of reversible logic improves both controllability and observability of a circuit. As applicable to *molecular QCA implementations*, array structures of basic units (referred to as modules) made of reversible gates are analyzed. Testing and constant testability (C-testability) have been extensively studied for VLSI Iterative Logic Array (ILA) [5, 15]. ILAs built with reversible logic gates have been studied in [3] by considering the testing benefits from the reversible logic function. In our work the testability of a one dimensional (1D) array of reversible QCA array is investigated, with the assumption of multiple faulty modules.

2 Preliminaries

The QCA gates used in this paper contain MV, INV, fanout and fixed polarization cells. These gates are therefore not strictly "reversible". However, the focus of this paper is not on designing a system with zero energy dissipation, but rather exploring the testability advantage of logically reversible array systems and the *one-to-one onto mapping* in logic design. In this paper, reversible logic gates are defined by their bijective property between inputs and outputs.

For molecular QCA, manufacturing defects can occur in both the *chemical synthesis* phase (in which the QCA cells are manufactured) and the *deposition* phase (in which the QCA cells are attached to a substrate). Defects are more likely to occur in the deposition phase than in the chemical synthesis phase, which result in perfectly manufactured cells being imperfectly placed in the substrate. As a result, two types of defects are considered, namely the *missing cell defect* and the *additional cell defect*. The former represents the case in which a cell fails to attach to the substrate, while the latter represents the case of an unwanted cell deposition. Thus, the *single missing/additional cell* defect is used as applicable defect model in our research. The fault characterization of QCA gates is based on this model.

The following conditions, notations and assumptions are valid hereafter as well as in previous work [10].

1. As applicable to molecular implementations, *missing/additional* QCA cell defects are considered in a gate. Only defects in the active devices (MV and INV) are considered since for interconnect results under this defect model have been reported in a previous manuscript [12]. A *single* QCA cell defect per gate is assumed.
2. The basic function of a reversible logic gate is a one-to-one onto mapping of input to output patterns. Let a_i represent the 3 bit pattern whose decimal value is i, e.g. $a_0 = 000$, $a_5 = 101$. Then, a reversible logic gate can be represented by a set of input-to-output mappings $a_i \rightarrow a_j$. Different defects can result in different faulty input-to-output mapping functions. Hereafter, we use the fault pattern FP$_i$ to denote the ith faulty mapping set from inputs to outputs. It is evident that the number of distinct output patterns from a gate can not be greater than the number of distinct input patterns applied to that gate.
3. Simulation is performed using QCADesigner v1.4 coherence vector engine [14]. All cells used in the simulations have size of $10nm \times 10nm$, dot size of $2.5nm$ and cell to cell distance of $2.5nm$.
4. Four phase clocking for adiabatic switching i(as described in Section 1) is assumed.

2.1 Reversible Gates in QCA

The QCA implementation of four reversible gates has been proposed in [10]. All four gates have three inputs and three outputs. These reversible gates have been characterized in the presence of a *single missing/additional cell* defect [10]. The simulator QCADesigner has been used to perform an exhaustive test (the test set consists of all 8 input test patterns) for defect characterization. Simulation has been conducted for each possible *single missing/additional cell* condition in each gate. The simulated output is compared with the fault-free output and different faulty patterns are recorded and categorized. The fault characterization method and results have been reported in [10] and will be used in this paper as basis for analyzing the C-testability of QCA reversible arrays.

– *The Fredkin Gate* [13] is one of the most studied reversible gate with 3 inputs and 3 outputs. It has the following three output functions (where u' denotes the complement of u):

$$\begin{cases} v = u \\ y1 = u'x2 + ux1 \\ y2 = u'x1 + ux2 \end{cases}$$

The QCA implementation of the Fredkin gate is shown in Fig. 3.
There are 14 fault patterns observed in the Fredkin Gate when one missing/additional cell defect is considered. The fault-free output pattern as well as the 14 faulty patterns are shown in Table 1. A minimal test set with cardinality of 3 has been established for the Fredkin gate [10] that detects a gate for all above test patterns. The test set is $< a_1, a_2, a_7 >$ $(< 001, 010, 111 >)$.
– *The Toffoli Gate* [13] has the output functions

$$\begin{cases} y1 = x1x2' + x1x3' + x1'x2x3 \\ y2 = x2 \\ y3 = x3 \end{cases}$$

The layout of the QCA implementation of the Toffoli gate is shown in Fig. 4.

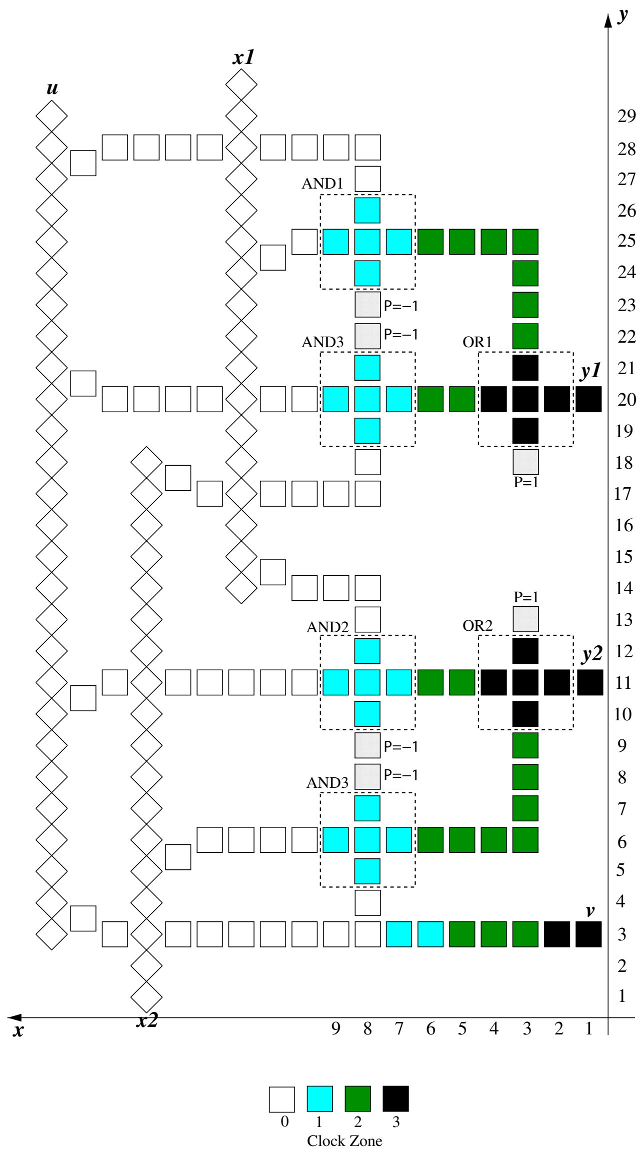


Fig. 3 QCA layout of the Fredkin gate

Under single missing/additional cell defect assumption, the QCA implementation of Toffoli Gate has 13 possible fault patterns as shown in Table 2. The test set a_3,a_4,a_7 ($< 011, 100, 111 >$) can detect all fault patterns.

– *The QCA1 gate* has been proposed in [10]. It is defined by the output functions:

$$\begin{cases} y1 = MV(x1, x2, x3) \\ y2 = MV(x1, x2, x3') \\ y3 = MV(x1', x2, x3) \end{cases}$$

Figure 5 gives the layout of the QCA1 gate.
For *QCA1*, there are 7 different fault patterns (shown in Table 3) that can be caused by a single missing/additional cell defect. The test set that covers all these fault patterns consists of three vectors:$< a_1, a_3, a_5 >$ ($< 001, 011, 101 >$)

– *The QCA2 gate* is another reversible gate proposed in [10]. It is defined by the output functions

$$\begin{cases} y1 = MV(x1, x2, x3) \\ y2 = MV(x1, x2, x3') \\ y3 = MV(x1', x2, x3') \end{cases}$$

The gate can be implemented using the layout given in Fig. 6.
QCA2 also has 7 possible fault patterns as given in Table 4. The test detecting all fault patterns has three vectors: $< a_0, a_2, a_4 >$ ($< 000, 010, 100 >$).

2.2 Testability of a 1D Array

The 1D array studied in this paper is an Iterative Logic Array (ILA), i.e., the array is made of N identical modules and identical interconnections between modules. The module in the array is one of the aforementioned reversible gates. A *single fault* array means there is at most one faulty module in the array, while a *multiple fault array* refers to the scenario in which any number of modules in the array can be faulty. C-testability refers

Table 1 Fault patterns of the Fredkin gate

Input	Fault	FP													
vector	free	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a_0	a_0	a_0	a_1	a_1	a_0	a_1	a_0	a_1	a_2	a_2	a_0	a_0	a_2	a_2	a_0
a_1	a_2	a_3	a_3	a_3	a_2	a_3	a_3	a_3	a_2	a_2	a_3	a_2	a_2	a_0	a_2
a_2	a_1	a_1	a_1	a_1	a_1	a_1	a_1	a_0	a_3	a_3	a_1	a_1	a_3	a_3	a_3
a_3	a_3	a_3	a_3	a_3	a_3	a_3	a_3	a_2	a_3	a_3	a_3	a_3	a_3	a_1	a_3
a_4	a_4	a_4	a_5	a_4	a_4	a_5	a_4	a_4	a_4	a_6	a_4	a_4	a_6	a_4	a_4
a_5	a_5	a_5	a_5	a_5	a_4	a_4	a_4	a_5	a_5	a_7	a_5	a_5	a_7	a_5	a_5
a_6	a_6	a_6	a_7	a_6	a_6	a_7	a_6	a_6	a_6	a_6	a_6	a_4	a_4	a_6	a_4
a_7	a_7	a_7	a_7	a_7	a_6	a_6	a_6	a_7	a_7	a_7	a_7	a_5	a_5	a_7	a_5

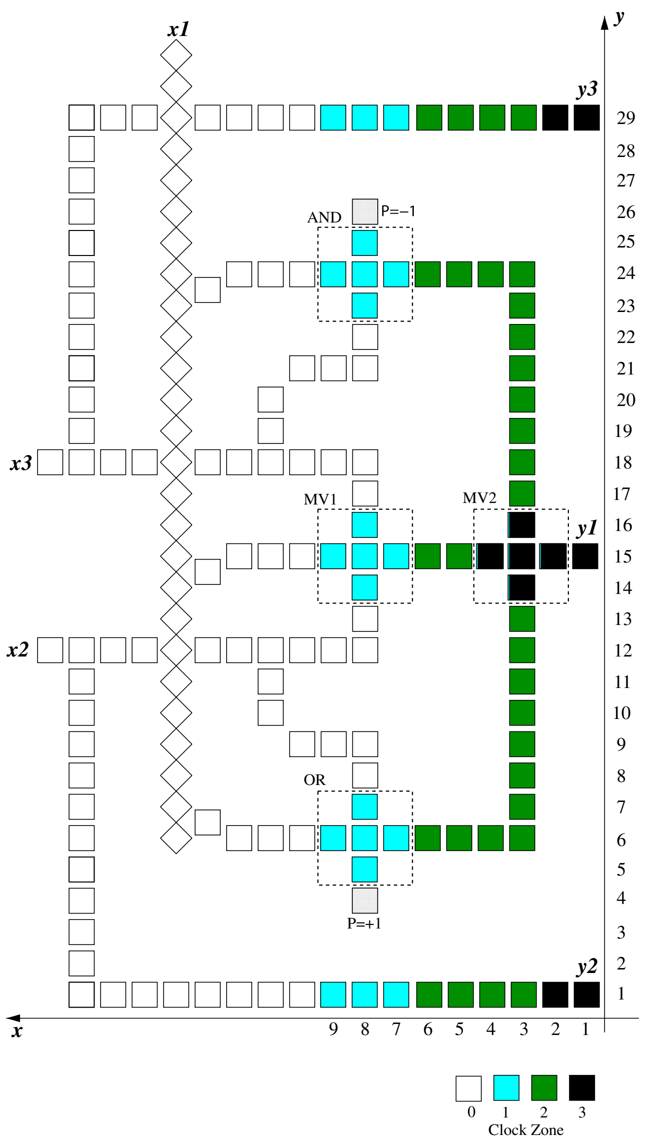


Fig. 4 QCA layout of the Toffoli gate

to the property that the number of vectors for testing the 1D array is *independent* of the array size, N. In [10], the 1D array configuration with *no direct* controllability and observability of the intermediate modules has been considered (shown in Fig. 7).

Due to the reversibility of each module, controllability in the array is not difficult. In the absence of a fault, the leftmost module is provided with all possible input combinations during the testing process, then all modules in the array will also receive all possible input combinations. So if the array is fault-free, the exhaustive patterns should be observed at the primary outputs. It has been proved that for a single fault, detection is guaranteed when an exhaustive test set is applied to the primary inputs [10].

3 Array Testability with Multiple Faults

3.1 Original Array

Testing becomes more difficult if multiple faulty modules are present. If any module in the array has an *irreversible fault* (i.e. a fault that change the module's reversible function into an irreversible one), then the number of its distinct output patterns is less than the number of distinct input patterns. Therefore the erroneous output due to this fault can always be propagated to the primary outputs of the array. So, *any number* of irreversible faults can be observed and multiple fault detection is accomplished. The C-testability of a 1D array made of Toffoli gates is guaranteed, because all fault patterns (as shown in Table 2) are irreversible.

However, fault masking may occur if all faults in the array are *reversible* (resulting in a reversible gate function different from the correct one). In such a case, even a fully exhaustive test set can not detect the faults. To solve masking of reversible faults and achieve C-testability, lines for controllability and/or observability have to be added to the internal modules of the array. Using the additional controllability and observability, C-testability can be accomplished. It is assumed that an

Table 2 Fault patterns of the Toffoli gate

Input	Fault free	FP												
vector	output	1	2	3	4	5	6	7	8	9	10	11	12	13
a_0	a_0	a_0	a_4	a_0	a_4	a_0	a_0	a_0	a_4	a_0	a_4	a_0	a_4	a_0
a_1	a_1	a_1	a_5	a_1	a_1	a_1	a_1	a_1	a_5	a_1	a_5	a_5	a_1	a_1
a_2	a_2	a_2	a_6	a_2	a_2	a_2	a_2	a_2	a_6	a_2	a_6	a_2	a_6	a_2
a_3	a_7	a_7	a_7	a_3	a_3	a_7	a_7	a_3	a_7	a_3	a_7	a_3	a_3	a_7
a_4	a_4	a_4	a_4	a_4	a_4	a_0	a_0	a_0	a_0	a_0	a_0	a_4	a_4	a_0
a_5	a_5	a_5	a_5	a_5	a_5	a_5	a_1	a_1	a_5	a_1	a_1	a_5	a_1	a_5
a_6	a_6	a_6	a_6	a_6	a_6	a_6	a_6	a_6	a_2	a_6	a_6	a_2	a_6	a_2
a_7	a_3	a_7	a_7	a_3	a_3	a_7	a_3	a_3	a_7	a_7	a_7	a_3	a_3	a_7

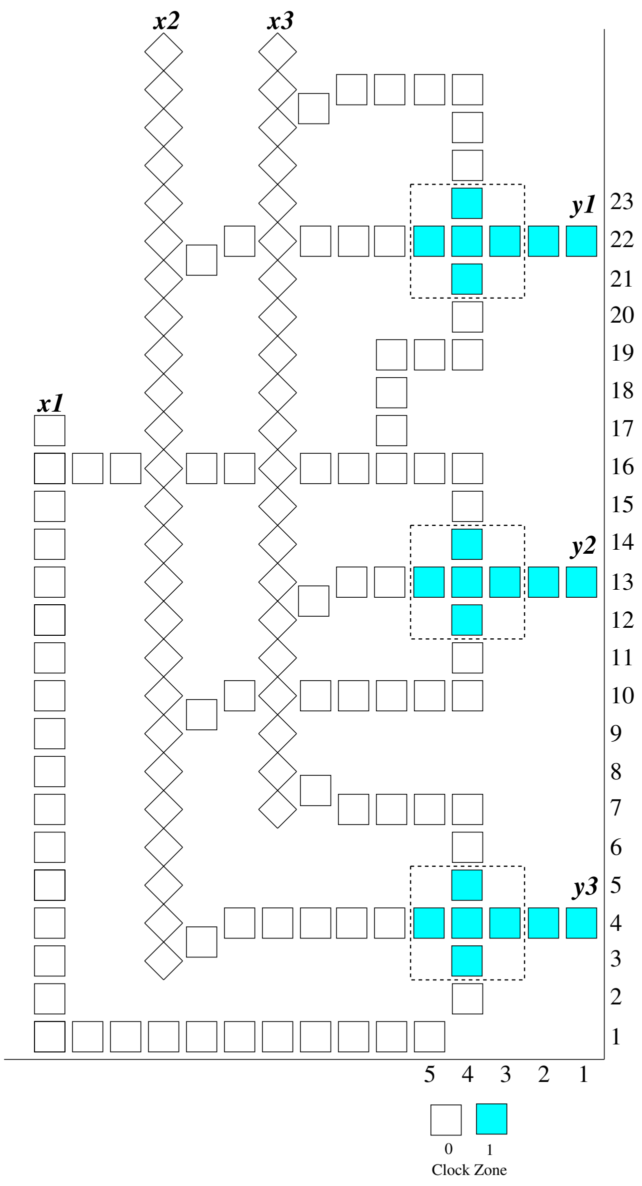


Fig. 5 QCA layout of QCA1

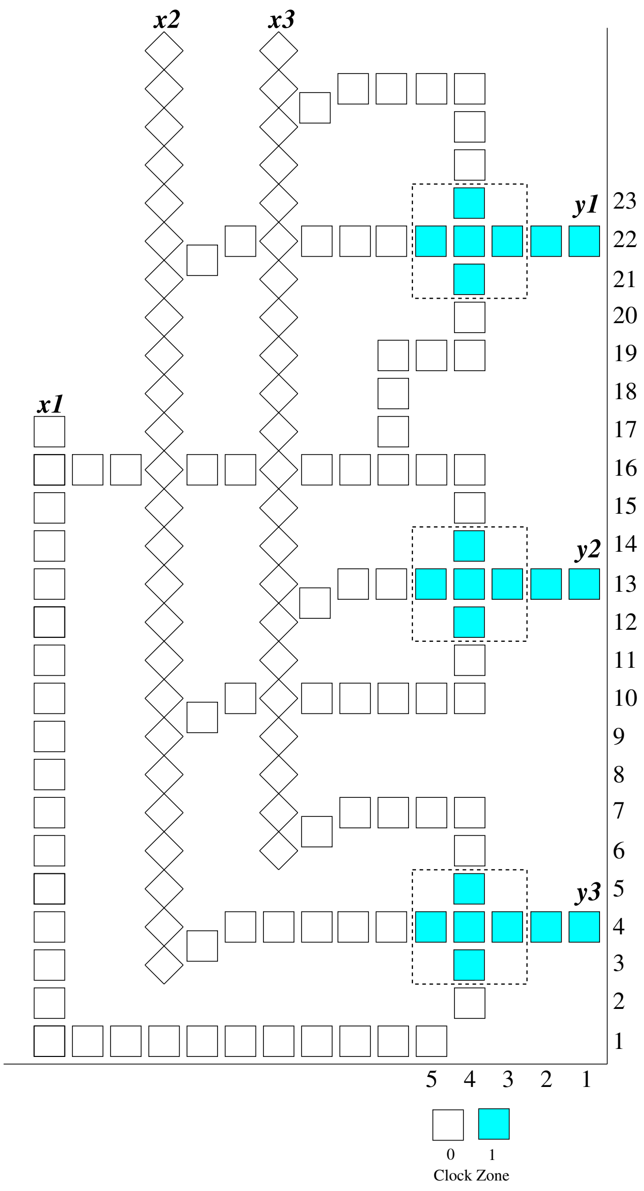


Fig. 6 QCA layout of QCA2

Table 3 Fault patterns of the QCA1 gate

Input	Fault free	FP						
vector	output	1	2	3	4	5	6	7
a_0	a_0	a_0	a_0	a_0	a_0	a_0	a_0	a_4
a_1	a_1	a_0	a_0	a_1	a_1	a_3	a_1	a_1
a_2	a_3	a_3	a_3	a_3	a_1	a_3	a_7	a_7
a_3	a_5	a_5	a_4	a_7	a_5	a_7	a_5	a_5
a_4	a_2	a_2	a_3	a_0	a_2	a_0	a_2	a_2
a_5	a_4	a_4	a_4	a_4	a_6	a_4	a_0	a_0
a_6	a_6	a_7	a_7	a_6	a_6	a_4	a_6	a_6
a_7	a_7	a_7	a_7	a_7	a_7	a_7	a_7	a_3

Table 4 Outputs of fault free and faulty QCA2

Input	Fault free	FP						
vector	output	1	2	3	4	5	6	7
a_0	a_1	a_0	a_0	a_1	a_1	a_1	a_1	a_5
a_1	a_0	a_0	a_0	a_0	a_0	a_2	a_0	a_0
a_2	a_3	a_3	a_2	a_3	a_1	a_3	a_7	a_7
a_3	a_5	a_5	a_5	a_7	a_5	a_7	a_5	a_5
a_4	a_2	a_2	a_2	a_0	a_2	a_0	a_2	a_2
a_5	a_4	a_4	a_5	a_4	a_6	a_4	a_0	a_0
a_6	a_7	a_7	a_7	a_7	a_7	a_5	a_7	a_7
a_7	a_6	a_7	a_7	a_6	a_6	a_6	a_6	a_2

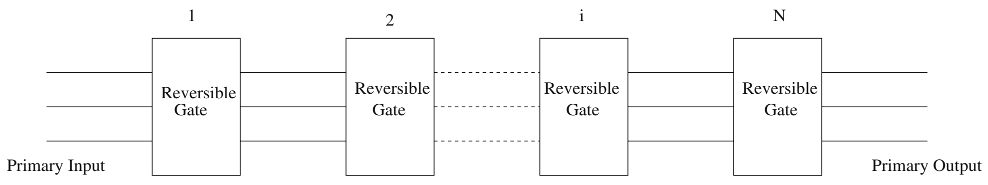


Fig. 7 1-D array of modules made of reversible logic gates

exhaustive test set will be applied to the primary inputs unless stated otherwise.

3.2 Additional Observability

The 1D array with added lines for observability is shown in Fig. 8. Different fault models can be considered for the modules:

1. The traditional stuck-at fault (*SAF*), and the line bridging fault (*BF*) are *irreversible faults*, i.e. faults that will result in an irreversible function. Therefore, they can be detected with a constant number of tests. The array is C-testable and no additional control, or observable line is required to the modules. However, these fault models are not sufficient for modeling the faults in QCA [12].
2. Consider an arbitrary functional fault (*AFF*) model in which it is assumed that a fault may cause an arbitrary change in the truth table of the circuit. If there is any number of modules with irreversible faults, then the number of possible output combinations of the array will be less than an exhaustive combination (eight for three inputs). This scenario (irrespective of the number of faulty modules) can be detected at the primary outputs. So, only those *reversible faults* must be considered.
3. This fault model is referred to as the arbitrary reversible fault model (*ARF*). In ARF, a fault is assumed to change the truth table of the gate as long as the resulting function is still reversible. It is desirable to achieve C-testability provided observability is added, i.e., the outputs of the intermediate modules are made directly observable. Unfortunately, it will be shown next through an example that even if just a subset of the ARF model is considered, the array is not C-testable.
4. Consider the so-called single pin inversion (*SPI*) model as a subset of the ARF model. In the SPI model, every module has at most one fault in one of its input/output pins, such that an inversion of the signal occurs at that pin. The inversion fault is very common in QCA circuits, because it can easily result from imperfect deposition of QCA cells [12]. As shown in Fig. 9, if a fault appears between the nth output pin of the kth module and the nth input pin of the $(k+1)$th module, then detection can only be guaranteed by observing directly the internal pins (in this case pin n) between these two modules. The faulty pin can be any one of the module pins, so we must observe every internal connection to detect all SPI faults, i.e. this is a pathological case of an array as a fully observable system (i.e., every output of every module is observable).

The general fault model above gives pessimistic result for C-testability of reversible arrays under multiple faults. However, a case-by-case study of different logic gates (as module of the array), together with their specific input/output functions and fault patterns, shows that C-testability is often achievable. All possible reversible faults in a module must be considered when selecting lines in a module for observability. Three conditions are proposed for selecting lines for observability in each module of an array:

1. *Rule 1:* If all reversible faults change some entries of the truth table of one output signal, then this output signal should be a primary observable line. For example, consider a reversible module with fault free and fault patterns given in Table 5. It has two reversible fault patterns: FP$_1$ affects the output y_3, FP$_2$ affects the outputs y_2 and y_3. By Rule 1, the truth table of y_3 is changed by all reversible faults,

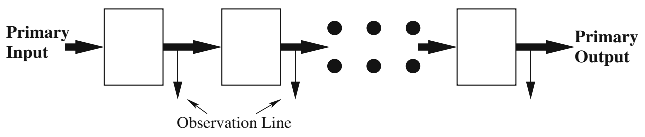


Fig. 8 1D array of reversible module with increased observability

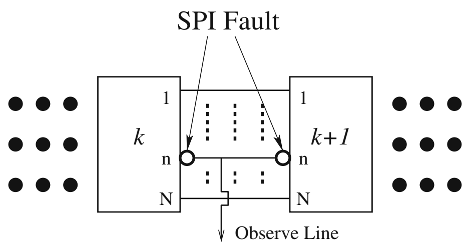


Fig. 9 SPI (single pin inversion) fault

Table 5 Example for rule 1

Input	Fault-free	FP_1	FP_2
$x_1x_2x_3$	$y_1y_2y_3$	$y_1y_2y_3$	$y_1y_2y_3$
0 0 0	0 0 0	0 0 **1**	0 0 0
0 0 1	0 0 1	0 0 **0**	0 0 1
0 1 0	0 1 0	0 1 0	0 1 0
0 1 1	0 1 1	0 1 1	0 1 1
1 0 0	1 0 0	1 0 0	1 **1 1**
1 0 1	1 0 1	1 0 1	1 0 **0**
1 1 0	1 1 1	1 1 1	1 **0** 1
1 1 1	1 1 0	1 1 0	1 1 0

Table 6 Example for rule 2

Input	Fault-free	FP_1	FP_2	FP_3
$x_1x_2x_3$	$y_1y_2y_3$	$y_1y_2y_3$	$y_1y_2y_3$	$y_1y_2y_3$
0 0 0	0 0 0	0 0 **1**	0 0 0	0 0 0
0 0 1	0 0 1	0 0 **0**	0 0 1	0 0 1
0 1 0	0 1 0	0 1 0	0 1 0	0 1 0
0 1 1	0 1 1	0 1 1	0 1 1	0 1 1
1 0 0	1 0 0	1 0 0	1 **1 1**	1 **1** 0
1 0 1	1 0 1	1 0 1	1 0 **0**	1 0 1
1 1 0	1 1 1	1 1 1	1 **0** 1	1 1 1
1 1 1	1 1 0	1 1 0	1 1 0	1 **0** 0

so if y_3 is made a primary observable line in the modules, then the array is C-testable.

The application of Rule 1 to QCA1 and QCA2 arrays makes them C-testable with one observable line as primary output in each module. As shown in Fig. 10, the array can be fully tested by applying the exhaustive test set (of cardinality 8) at the primary inputs. For the QCA1 array, the only reversible fault pattern is FP_7 and it modifies the output $y3$ of a faulty module. So by observing this output, detection will occur at the faulty module. Similarly, the QCA2 array is C-testable by making the output $y2$ as a primary observable line in each module.

2. *Rule 2:* If all reversible faults can be propagated to a module output prior to masking, then such output should be a primary observable line.

   In Table 6, a reversible module and a fault pattern are shown as an example. The module has 3 reversible fault patterns: FP_1 changes the output y_3, FP_2 changes the outputs y_2 and y_3, FP_3 changes y_2. FP_1 and FP_2 can be propagated to y_3 of the faulty module. FP_3 can be propagated to y_3 of the module after the faulty one, independently of the status (faulty or fault free) of this module. By Rule 2, the selection of y_3 as an observable line for each module makes the array C-testable.

3. *Rule 3:* If the above conditions cannot be satisfied, two or more observable lines are needed in each module. Each of these lines can be used to observe part of the possible fault patterns and the union of the covered fault patterns must be the entire fault pattern set.

   By applying Rules 1 and 2, it is possible to observe different module outputs to detect different fault patterns. In the general case, the problem of selecting multiple output lines for observability is a set covering problem. Let the outputs lines of a reversible module be $y_1, y_2 ... y_n$. The possible fault patterns of each module are denoted by $\mathrm{FP}_1, FP_2 ... FP_m$. By observing an output line y_i, a group of fault patterns can be detected, i.e. y_i "covers" a group of fault patterns. Thus, the problem is equivalent to selecting the minimum cardinality set of output lines such that all fault patterns are covered. The set covering problem is NP hard. However, heuristic algorithms based on greedy criteria can be used to find approximate solution.

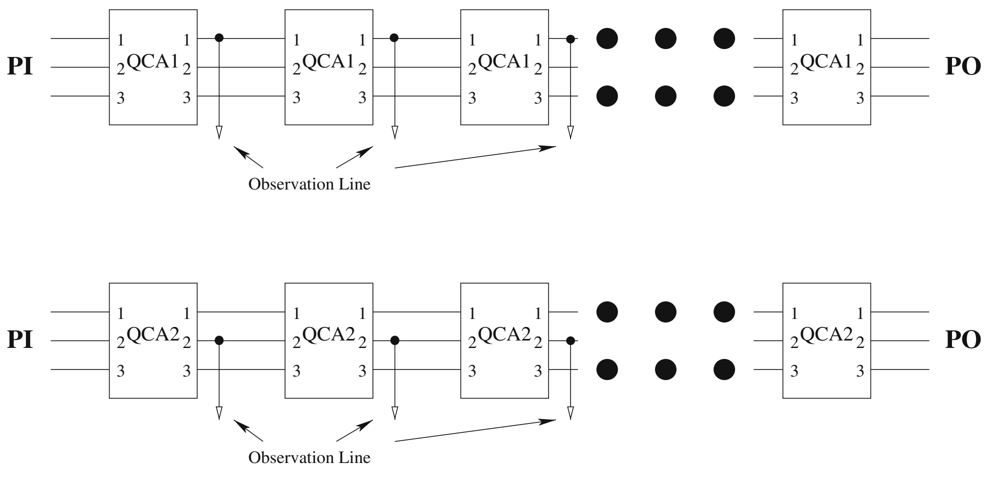


Fig. 10 C-Testability of 1D array

Consider the Fredkin gate for example; there are 2 reversible fault patterns, FP_7 and FP_{13}. FP_7 changes the output $y3$ and FP_{13} changes the output $y2$; so, Rule 1 cannot establish a primary observable line for both fault patterns. The output of two adjacent modules with FP_7 and FP_{13} will result in masking; hence, Rule 2 cannot be used. By applying Rule 3, $y1$ and $y2$ are selected as multiple observable lines, because all possible reversible fault patterns are detected.

Each of the above three rules gives a sufficient condition for constructing a C-testable array. Rule 1 requires only one observable line per module and is relatively easy to apply, so it is preferred. Rule 2 requires also one observable line, but propagation under different multiple fault patterns must be established. Therefore, it can be applied if Rule 1 fails. Rule 3 requires multiple observable lines and may account for a large overhead. Furthermore, the analysis of Rules 1 and 2 can facilitate the application of Rule 3. Hence, realistically Rule 3 should be applied after Rules 1 and 2 have been unsuccessful. A combination of Rules 1 and 2 provides the necessary condition for constructing a C-testable array with only one primary observable line per module. Rule 3 can ensure C-testability in an array as long as a large overhead is acceptable.

3.3 Additional Observability and Controllability

By adding lines for controllability as well as observability, the testability of the array can be improved. A smaller number of test patterns will be needed to fully test the array. Unfortunately, there is no technique for systematically building and testing this type of array. 1D arrays made of Fredkin gates, QCA1 gates and QCA2 gates are presented as examples.

3.3.1 Array of Fredkin Gate

For the Fredkin gate, it has been shown [10] that fault masking exists in an 1D array such as in Fig. 7. However, the 1D array of Fig. 11a is C-testable. In the array of N modules, additional controllability is provided by setting the u input of each module as a primary (vertical) input; additional observability is obtained by setting the $y2$ output of each module as a primary (vertical) output. Let the primary (horizontal) inputs $x1, x2$ of the first module in the array be denoted as PI_1, PI_2. Let the ith module in the array be G_i, the input u of G_i be denoted as U_i. The mapping between inputs and outputs of a fault free module as well as a faulty module is shown in Table 1. The fault patterns are categorized into two types: (1) FP_1,FP_8, FP_9, FP_{10}, FP_{11}, FP_{12}, FP_{13} and FP_14 are type I; (2) FP_2,FP_3,FP_4,FP_5,FP_6 and FP_7 are type II. The test vector set that detects multiple faults is as follows:

1. *First Test Vector:* $PI_1 = 0$, $PI_2 = 1$, $U_i = 0$ for all i.
   If the array is fault free, then all modules in the array will receive input vector $ux1x2 = 001 = a_1$. The expected (fault free) output at $y2$ of any module in the array is "0".
2. *Second Test Vector:* $PI_1 = 1$, $PI_2 = 1$, $U_i = 1$ for all i.
   If the array is fault free, then all modules in the array will receive as input $ux1x2 = 111 = a_7$. The expected (fault free) output at $y2$ of any module is "1".
3. *Third Test Vector:* $PI_1 = 1$, $PI_2 = 0$, $U_i = 0$ for $i = 1, 3, 5, 7...$ and $U_i = 1$ for $i = 2, 4, 6...$, as shown in Fig. 11b.
   If the array is fault free, the input vector $ux1x2 = 010 = a_2$ is applied to all G_i (for i as an odd integer). The expected output at $y2$ of G_i is "1" for odd i and "0" for even i.
4. *Fourth Test Vector:* $PI_1 = 0$, $PI_2 = 0$, $U_i = 1$ for $i = 1, 3, 5, 7...$ and $U_i = 0$ for $i = 2, 4, 6...$, as shown in Fig. 11b.
   If the array is fault free, $ux1x2 = 010 = a_2$ is applied as input vector to all G_i (for i as an even integer). The expected output at $y2$ of G_i is "0" for odd i and "1" for even i.

Initially, it will be shown that if the array contains any (single or multiple) type II fault pattern(s), detection is accomplished by observing $y2$ as primary output of the modules using the first and second test vectors. When applying vector 1, all $U_i = 0$ and $v = u$ for all modules, every module (either faulty or fault free) will have as input $x1 = 0$. Therefore, any module in the array has $ux1 = 00$; so, if a module with fault pattern FP_2,FP_3,FP_5 or FP_7 is present, then the $y2$ output of that module will be "1" instead of the expected "0". Thus, these fault patterns will be detected by vector 1. Similarly, when applying vector 2, all $U_i = 1$ and $PI_1 = 1$, $PI_2 = 1$; therefore, each module in the array will have input $ux1 = 11$. If a module with fault pattern FP_4 or FP_6 is present, the $y2$ output of that module is "0" (instead of the expected "1"). Thus by applying vector 1 and vector 2, any number of type II fault patterns can be detected.

Consider next the case in which the array contains faulty modules with only type I fault patterns. Let the faulty module that is closest to the primary inputs be G_i, i.e. only fault free modules exist from the primary inputs to G_i. G_i must have one of the type I fault patterns, i.e. FP_1,FP_8 ,FP_9 ,FP_{10} ,FP_{11} ,FP_{12} ,FP_{13} or

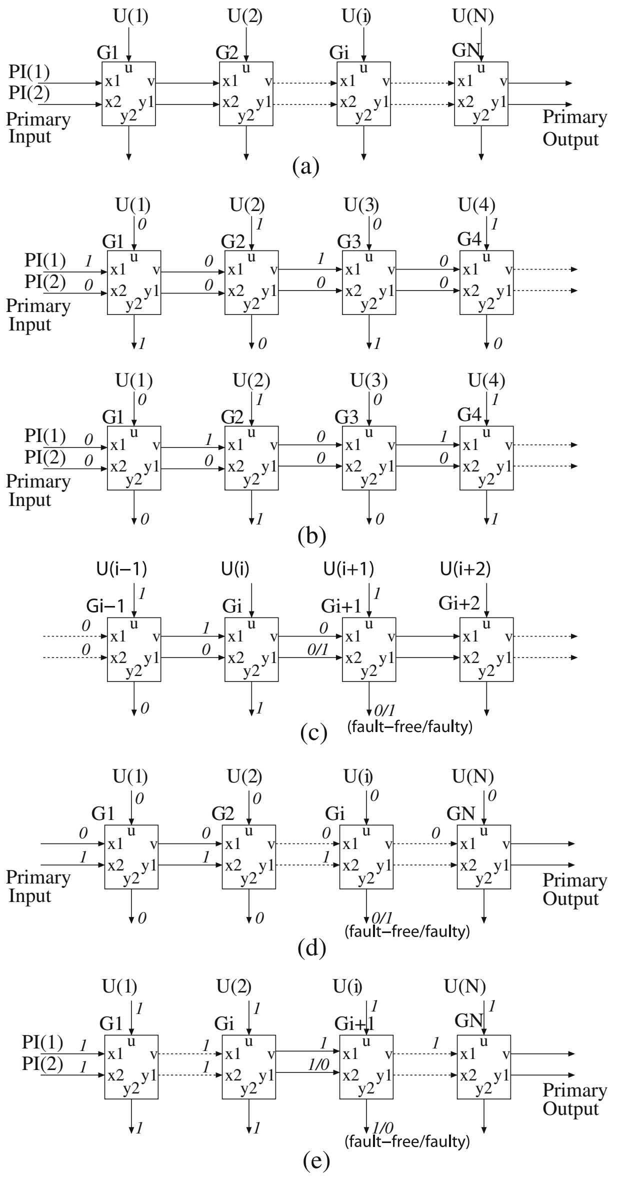


Fig. 11 C-Testability of 1D *Fredkin gate* array (added observability and controllability lines)

FP_{14}. If i is odd, the third vector $ux1x2 = 010$ to G_i is applied; if i is even, then the fourth vector $ux1x2 = 010$ to G_i is applied. In both cases, G_i will have as inputs $ux1x2 = 010$ and the expected output is $vy1y2 = 001$; therefore, G_{i+1} is expected to have as inputs $ux1x2 = 100$ and generate a “0” at the output $y2$, as shown in Fig. 11c. If G_i has fault patterns FP_8 ,FP_9 ,FP_{10},FP_{12} ,FP_{13} or FP_{14}, G_i will produce a “1” at the output $y1$, thus G_{i+1} will have as inputs $ux1x2 = 101$. Since G_{i+1} is either fault free, or it generates one of the type I fault patterns, then G_{i+1} will produce a “1” (instead of the expected “0”) at the output $y2$ (as shown in Table 1), thus, the fault can be detected. Assume that G_i generates FP_1. When the first vector is applied, G_i will have as inputs $ux1x3 = 001$, at the $y2$ output a “1” will be produced (instead of the expected “0”). So, this fault can also be detected. The only other case is that G_i has as fault pattern FP_{11}, it will be shown next that the second vector can detect it. When the second vector is applied, G_i will have as inputs $ux1x2 = 111$, the expected value of the $y2$ output of all modules is “1”. Since G_1 has as fault pattern FP_{11}, then it will a “0” (instead of a “1”) at the $y1$ output. So G_{i+1} has as inputs $ux1x2 = 110$. G_{i+1} is either fault free or contains one of the type I fault patterns; so G_{i+1} will produce a “0” (instead of the expected “1”) at the primary output $y2$ (as shown in Table 1), thus detection is accomplished.

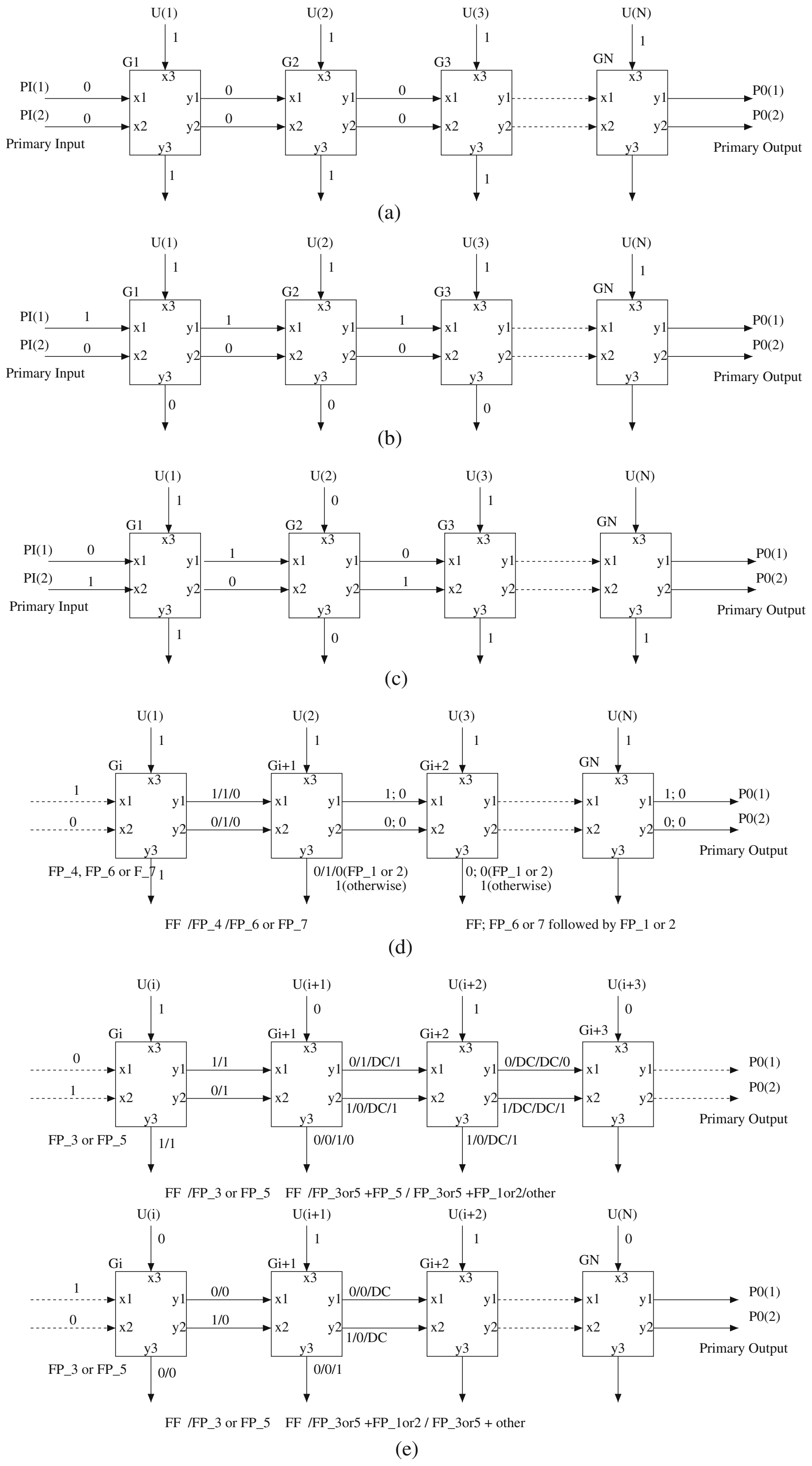
3.3.2 Array of QCA1 Gates

With additional controllability and observability, a 1D array of QCA1 (shown in Fig. 12a) is C-testable under a multiple faulty module assumption. In an array of N modules, additional controllability is provided by setting $x3$ of each module as a primary vertical input; additional observability is obtained by setting $y3$ of each module as a primary vertical output. Let the primary inputs for $x1$ and $x2$ of the first module in the array be denoted as PI_1 and PI_2; the primary outputs for the $y1$ and $y2$ outputs of the last module be denoted as PO_1 and PO_2 and the ith module in the array be denoted by G_i. If $x3$ of G_i is denoted by U_i, then the fault patterns of the mapping between inputs and outputs for the fault free and faulty modules with FP_i are shown in Table 3.

The test vector set that detect any multiple faulty modules, is given as follows:

1. *First Test Vector:* $PI_1 = 0, PI_2 = 0,\ U_i = 1$ for all i, as shown in Fig. 12a. If the array is fault free, $x1x2x3 = 001 = a_1$ is applied to all G_i. The expected output at $y3$ is “1” for every G_i. The expected value at the primary outputs is $PO_1PO_2 = 00$.
2. *Second Test Vector:* $PI_1 = 1, PI_2 = 0,\ U_i = 1$ for all i, as shown in Fig. 12b. If the array is fault free, this applies $x1x2x3 = 101 = a_5$ to all G_i. The expected value at $y3$ is “0” for each G_i. The expected value at the primary outputs is $PO_1PO_2 = 10$.
3. *Third Test Vector:* $PI_1 = 0, PI_2 = 1,\ U_i = 1$ for $i = 1, 3, 5, 7...$ and $U_i = 0$ for $i = 2, 4, 6...$, as shown in Fig. 12c. If the array is fault free, this vector applies the input vector $x1x2x3 = 011 = a_3$ to all G_i (for odd i) and $x1x2x3 = 100 = a_4$ to all G_i (for even i). The expected value at $y3$ of G_i is “1” (for odd i) and “0” (for even i). The expected value at the primary outputs is $PO_1PO_2 = 01$ (for even N) and $PO_1PO_2 = 10$ (for odd N).

Fig. 12 C-Testability of 1D *QCA1 gate* array (added observability and controllability lines)



A test set with 100% coverage for an array with a module made of a QCA1 gate is $x1x2x3{=}a_1{=}001, a_3{=}011$ (or $a_4{=}100$) and $a_5{=}101$. The first test vector applies a_1 to all modules, while the second test vector applies a_5 to all modules. The third test vector applies a_3 or a_4 to all modules. Next, it will be shown that these vectors can detect multiple faulty modules for arbitrary N.

Assume that the faulty module closest to the primary inputs is given by G_i; then G_i must have one of the fault patterns given in Table 3. Consider initially the case when G_i has FP_1 or FP_2. When applying $x1x2x3{=}a_1{=}001$ to G_i, G_i will produce a "0" at the $y3$ output (instead of the expected "1"), thus it will be detected. Next consider the case when G_i has FP_4, FP_6 or FP_7. When applying $x1x2x3{=}a_5{=}101$ to G_i (as shown in Fig. 12d) and the fault pattern is FP_4, then $x1x2x3{=}a_7{=}111$ is applied to G_{i+1}. So, $y3$ of G_{i+1} will be "1" (instead of the expected "0") and detection will occur. If the fault pattern is FP_6 or FP_7, then $x1x2x3{=}a_1{=}001$ is applied to G_{i+1}. If G_{i+1} does not have FP_1 or FP_2, then $y3$ of G_{i+1} will become "1" (instead of the expected "0") and detection is accomplished. If there is FP_1 or FP_2 at G_{i+1}, G_{i+2} will receive as input $x1x2x3{=}a_1{=}001$, and so on, till the last module of the array. So, there will be either an unexpected "1" at $y3$ of some module, or an unexpected "00" at PO_1, PO_2. Detection is always accomplished for a faulty array. A different case occurs when G_i has FP_3 or FP_5, as shown in Fig. 12e. For the third vector, if i is odd, $x1x2x3{=}a_3{=}011$ is applied to G_i and G_{i+1} has $x1x2x3{=}a_6{=}110$ as input. If G_{i+1} has FP_1 or FP_2, an unexpected "1" will be detected at $y3$ of G_{i+1}. If G_{i+1} has FP_5, then $G_i + 2$ will receive $x1x2x3{=}a_4{=}100$ as input and an unexpected "0" is observed at $y3$ of G_{i+2}. Else, $x1x2x3{=}a_7{=}111$ will be applied to G_{i+2}. If G_{i+2} has FP_7, it will be detected by the second test vector; if it does not have FP_7, then $x1x2x3{=}a_6{=}110$ will be the input of G_{i+3}, and the previously described case will apply (for $x1x2x3{=}a_6{=}110$ on G_{i+1}). So, the fault will be detected either at $y3$ of some module, or at the primary output.

If i is even, the fault will result in $x1x2x3{=}a_1{=}001$ at G_{i+1}. If G_{i+1} does not have FP_1 or FP_2, this will result in an unexpected "1" at $y3$ of G_{i+1}. If G_{i+1} has FP_1 or FP_2, FP_1 or FP_2 is detected by the first test vector. Therefore, the array shown in Fig. 11a is C-testable and four test vectors are required to detect any number of faulty modules.

3.3.3 Array of QCA2 Gates

With additional controllability and observability, a QCA2 array as shown in Fig. 13a is C-testable under the assumption that multiple modules may be faulty.

In an array of N modules, additional controllability is provided by setting the $x2$ input of each module as a primary input; additional observability is obtained by setting the $y2$ output of each module as a primary output. Let the primary inputs at $x1$, $x3$ of the first module in the array be denoted as PI_1, PI_2; the primary outputs at the $y3$, $y1$ outputs of the last module be denoted as PO_1, PO_2. Let the ith module in the array be G_i, the input $x2$ of G_i be U_i. The fault patterns corresponding to the mappings between inputs and outputs of fault free and faulty modules (FP_i) are shown in Table 4.

The test vector set that can detect any number of faulty modules is as follows:

1. *First Test Vector:* $PI_1 = 0, PI_2 = 0, U_i = 0$ for all i, as shown in Fig. 13a. If the array is fault free, this vector applies the input vector $x1x2x3 = 000 = a_0$ to all G_i (for i as an odd integer) and the input vector is $x1x2x3 = 100 = a_4$ to all G_i (for i as an even integer). The expected output at $y2$ of G_i is "0" for odd i and "1" for even i. The expected output at the primary outputs is $PO_1PO_2 = 00$ if N is even and $PO_1PO_2 = 10$ if N is odd.
2. *Second Test Vector:* $PI_1 = 1, PI_2 = 0, U_i = 0$ for all i, as shown in Fig. 13a If the array is fault free, this vector applies as input $x1x2x3 = 000 = a_0$ to all G_i (for even i) and $x1x2x3 = 100 = a_4$ (for odd i) to all G_i. The expected output at $y2$ of G_i is "0" for even i and "1" for odd i. The expected output at the primary outputs is $PO_1PO_2 = 00$ if N is odd and $PO_1PO_2 = 10$ if N is even.
3. *Third Test Vector:* $PI_1 = 0, PI_2 = 0, U_i = 1$ for $i = 1, 3, 5, 7...$ and $U_i = 0$ for $i = 2, 4, 6...$, as shown in Fig. 13b If the array is fault free, then this vector applies as input $x1x2x3 = 010 = a_2$ (for odd i) and $x1x2x3 = 100 = a_4$ (for even i) to all G_i. The expected output at $y2$ of G_i is "1" for all i. The expected output at the primary outputs is $PO_1PO_2 = 00$ if N is even and $PO_1PO_2 = 10$ if N is odd.
4. *Fourth Test Vector:* $PI_1 = 0, PI_2 = 0, U_i = 0$ for $i = 1, 3, 5, 7...$ and $U_i = 1$ for $i = 2, 4, 6...$, as shown in Fig. 13b If the array is fault free, this vector applies as input $x1x2x3 = 010 = a_2$ (for even i) and $x1x2x3 = 100 = a_4$ to all G_i (for odd i). The expected output at $y2$ of G_i is "1" for i. The expected output at the primary outputs is $PO_1PO_2 = 00$ if N is odd and $PO_1PO_2 = 10$ if N is even.

A 100% coverage test set for a single QCA2 module is given by three tests, i.e. $x1x2x3{=}a_0{=}000, a_2{=}010, a_4{=}100$. The combination of vectors 1 and 2 applies as input a_0 and a_4 to all modules, while the combination of vectors 3 and 4 applies a_2 and a_4 to all modules. Next,

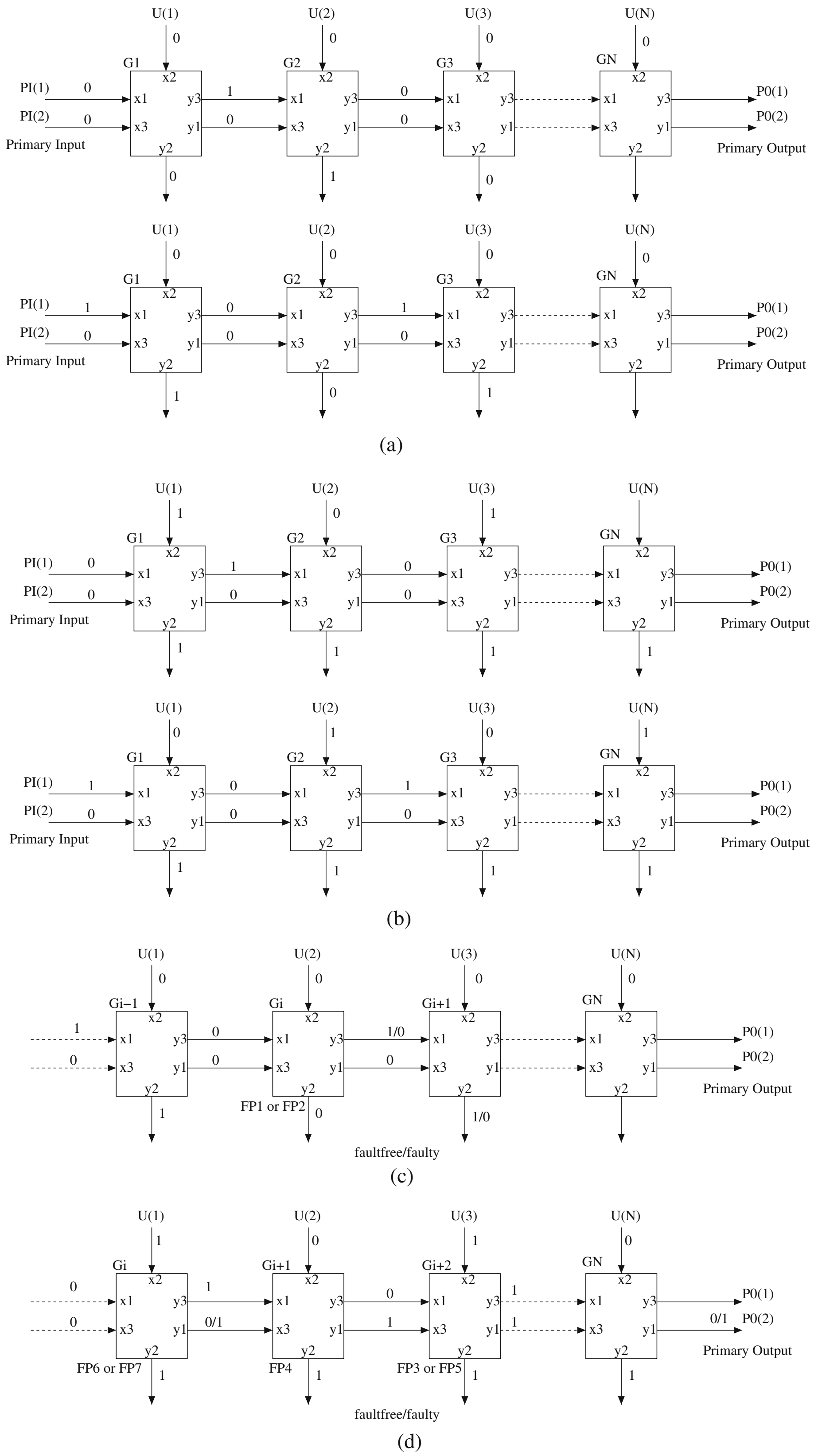


Fig. 13 C-Testability of 1D *QCA2 Gate* array (added observability and controllability lines)

it will be shown that these vectors can detect multiple faulty modules for an arbitrary N.

Assume that the faulty module closest to the primary input is G_i, then G_i must have one of the fault patterns from Table 4. First consider the case when G_i has FP_3 or FP_5. When applying vectors 1 and 2, $x1x2x3$=a_4=100 will be applied to G_i once. Therefore, G_i will produce "0" at $y2$ instead of the expected "1", thus it will be detected. Next consider the case when G_i has FP_4. When applying vectors 3 and 4, $x1x2x3$=a_2=010 will be applied to G_i once. G_i will generate "0" at $y2$ instead of the expected "1", thus it will be detected.

Another possible scenario is that G_i has FP_1 or FP_2, as shown in Fig.13c. When applying vectors 1 and 2, $x1x2x3$=a_0=000 will be applied to G_i once. G_i will generate $y1y2y3 = 000$ at the output, so G_{i+1} will have as inputs $x1x2x3$=a_0=000. Since G_{i+1} is either fault free or has one of the faulty patterns in Table 4, then G_{i+1} will generate "0" at $y2$ instead of the expected "1", thus it will be detected.

The only remaining case is when G_i has FP_6 or FP_7. When applying vectors 3 and 4, $x1x2x3$=a_2=010 will be applied to G_i once. G_i will generate $y1y2y3 = 111$ at the output so G_{i+1} will have as inputs $x1x2x3$=a_5=101 (as shown in Fig. 13d). If G_{i+1} is fault free or has any faulty pattern other than FP_4, it will generate "0" at $y2$ instead of the expected "1", thus it will be detected. If G_{i+1} has FP_4, then it will have as outputs $y1y2y3 = 110$ and thus it will provide $x1x2x3$=a_3=011 as inputs to G_{i+2}. If G_{i+2} is fault free or has any faulty pattern other than FP_3 or FP_5, it will generate "0" at $y2$ instead of the expected "1", thus it will be detected. If G_{i+2} has FP_3 or FP_5, then it will have as outputs $y1y2y3 = 111$ and thus providing $x1x2x3$=a_5=101 as inputs to G_{i+3}. At this point it can be seen that the fault is detected at y_2 of some module after G_i unless the faulty patterns of the 1D array are G_i has FP_6 or FP_7, G_{i+j} has FP_4 for $j = 1, 3, 5, 7...$ and G_{i+j} has FP_3 or FP_5 for $j = 2, 4, 6, 8....$

If the faulty array has the above faulty pattern, it will produce a "1" instead of the expected "0" at the primary output PO_2 (the output $y1$ of the last module), thus the fault can also be detected.

In conclusion, the array shown in Fig. 11a is C-testable and four test vectors are required to detect any number of faulty modules

4 Evaluation and Conclusion

In this paper, testing of 1D QCA reversible gate arrays under multiple faults has been considered. The fault module considered for the reversible gates are single missing/additional cell defect as it is suitable for the molecular implementation of QCA. Two methods have been explored and applied to the arrays of four different reversible gates. The first method adds only observe lines to the modules in the array, while the second method adds control lines as well as observe lines to the modules. The methods and gates are compared according to the array testability and the cardinality of full-coverage test set.

The results are summarized in Table 7. The first column of the Table is the type of gate used as module of the array. The second column shows the fault assumption: "S" stands for single faulty module and "M" stands for multiple faulty modules. The third column shows the cardinality of the test set for detection, i.e. the number of vectors in the test set. The last column is the type of array. The 1D array is shown in Fig. 7, where only the inputs of the first module can be controlled and only the outputs of the last module can be observed. A 1D array with 1OB, 1CO is an array in which each internal module has one controllable input and one observable output, e.g. the one shown in Fig. 11a. A 1D array with 2OB is an array in which each internal module has two observable outputs, but no controllable input.

The following conclusions can be drawn:

- One-dimensional array of Toffoli gates is C-testable under multiple faults.
- By adding observability lines to each module in the array, all the arrays considered can be fully tested. This method requires a larger test set. But its overhead in additional lines is modest.
- By adding both observability and controllability lines, all arrays considered can be tested with at most four vectors in the presence of multiple faulty

Table 7 C-testability of 1D array

Module	Faults S/M	Test set cardinality	Type of array
Toffoli	S	3	1D array
	M	3	1D array
Fredkin	S	3	1D array
	M	4	1D array with 1OB, 1CO
	M	8	1D array with 2OB
QCA1	S	3	1D array
	M	3	1D array with 1OB, 1CO
	M	8	1D array with 1OB
QCA2	S	4	1D array
	M	4	1D array with 1OB, 1CO
	M	8	1D array with 1OB

modules. This method can decrease the size of the test set but the overhead in additional lines can be significant.

– The choice of either methods depends on the specific gate used as module in the reversible array. For example, the Fredkin gate as module needs the same number of additional lines with both methods, but it requires a smaller test set by adding lines for both controllability and observability.

References

1. Agrawal VD (1981) An information theoretic approach to digital fault testing. IEEE Trans Comput 30:582–587
2. Bennett CH (1973) Logic reversibilty of computation. IBM J Res Develop 17:525–532
3. Chakraborty A (2005) Synthesis of reversible circuits for testing with universal test set and C-testability of reversible iterative logic arrays. In: Proc. 18th intl. conf. VLSI design
4. Compano R, Molenkamp L, Paul DJ (1999) Technology roadmap for nanoelectroincs. European Commission IST programme, Future and Emerging Technologies
5. Friedman, AD (1973) Easily testable iterative systems. IEEE Trans Computers C-22:1061–1064
6. Hennessy K, Lent CS (2001) Clocking of molecular quantum-dot cellular automata. J Vac Sci Technol 19(5):1752–1755
7. Landauer R (1961) Irreversibility and heat generation in the computing process. IBM J Res Develop 5:183–191
8. Lent CS, Liu M, Lu Y (2006) Bennett clocking of quantum-dot cellular automata and the limits to binary logic scaling. Nanotechnology 17:4240–4251
9. Lent CS, Tougaw PD, Porod W (1994) Quantum cellular automata: the physics of computing with arrays of quantum dot molecules. In: Proc. workshop on physics and computing, pp 5–13
10. Ma X, Huang J, Metra C, Lombardi F (2006) Testing reversible 1D arrays of molecular QCA. In: Proc. IEEE intl. symposium on defect and fault tolerance in VLSI systems (DFT), pp 71–79, Oct
11. Patel KN, Hayes JP, Markov IL (2004) Fault testing for reversible circuits. IEEE Trans CAD 23(8):1220–1230
12. Tahoori MB, Momenzadeh M, Huang J, Lombardi F (2004) Testing of quantum cellular automata. IEEE Trans Nanotechnology 3(4):432–442
13. Toffoli T (1980) Reversible computing. MIT Laboratory for Computer Science, Technical Report MIT/LCS/TM-151, Feb
14. Walus K, Dysart T, Jullien GA, Budiman RA (2003) QCADesigner: a rapid design and simulation tool for quantum-dot cellular automata. In: 2nd intl. workshop on quantum dots for quantum computing and classical size effect circuits. Notre Dame, IN, August
15. Wu C-W, Cappello PR (1990) Easily testable iterative logic arrays. IEEE Trans Comput 39:640–652

Xiaojun Ma received his B.S. degree in Electronic Engineering (2001) and M.S. degree in Microelectronics (2004) from Fudan University, China. Since 2004, he has been studying as a Ph.D. candidate at ECE Dept., Northeastern University. His current research interests are bio/nano computing, emerging technologies, reversible computing, defect tolerance and CAT/CAD.

Jing Huang received her B.S. degree in electronics engineering from Fudan University, Shanghai, China in 2001. She worked in the Computer Aided Test Lab in EE Department, Fudan University as research assistant from 1999 to 2001. She received her M.S. degree in Electrical Engineering and Ph.D. degree in Computer Engineering from ECE Dept., Northeastern University, Boston in 2005 and 2007, respectively. She worked as research assistant at Northeastern University from 2003 to 2007, her research interests include testing, design for testability and fault tolerance of VlSI, reconfigurable systems and nanotechnologies. She is currently a design engineer at Sun Microsystems.

Cecilia Metra obtained the degree (summa cum laude) in Electronic Engineering and the Ph.D. degree in Electronic Engineering and Computer Science from the Univ. of Bologna (Italy). She is an Associate Professor in Electronics in the Department of Electronic, Computer Science and Systems (DEIS) of the Univ. of Bologna. She is also affiliated with the Advanced Research Center on Electronic Systems for Information and Communication Technologies E. De Castro (ARCES) of the Univ. of Bologna. She has also been Visiting Scholar at the Univ. of Washington, Seattle (USA) from 1998 to 2001, and Visiting Faculty Consultant for Intel Corporation, Santa Clara (CA) in 2002.

She has been General Co-Chair of/The IEEE lnt'l Symp. on Defect and Fault Tolerance in VLSI Systems/ 2005 and 1999, the /IEEE Int'l On-Line Testing Symp./ 2006 and the /IEEE lntI On-Line Testing Workshop/ 2001, and Program Co-Chair of/The IEEE Int'l Symp. on Defect and Fault Tolerance in Vi.Sl Systems/ 1998, the /IEEE Int'l On-Line Testing Symp./ 2005, 2004 and 2003, and the /IEEE Int'l On-Line Testing Workshop/ 2002. She serves/served as Topic Chair for several international conferences, including the/International Test Conference /(ITC), the/Design, Automation and Test In Europe Conference /(DATE), the/European Test Symposium/ (ETS) and as Member of the Organizing Committee and/or Technical Program Committee of several international conferences, including the /Design Automation Conference/ (DAC) and the /IEEE VlSI Test Symposium/ (VTS).

Her research interests are in the field of Design and Test of Digital Systems, Reliable and Error Resilient Systems, Fault Tolerance, On-Line Testing, Fault Modeling, Concurrent Diagnosis and Debug. She is a Member of the Editorial Board of the/Microelectronics Journal/, the/Journal of Electronic Testing/: /Theory and Applications/ and the /1EEE Transactions on Computers/. She is a member of the IEEE Computer Society.

Fabrizio Lombardi graduated in 1977 from the University of Essex (UK) with a B.Sc. (Hons.) in Electronic Engineering. In 1977 he joined the Microwave Research Unit at University College London, where he received the Master in Microwaves and Modern Optics (1978), the Diploma in Microwave Engineering (1978) and the Ph.D. from the University of London (1982).

He is currently the holder of the International Test Conference (ITC) Endowed Chair Professorship at Northeastern University, Boston. At the same Institution during the period 1998–2004 he served as Chair of the Department of Electrical and Computer Engineering. Prior to Northeastern University he was a faculty member at Texas Tech University, the University of Colorado-Boulder and Texas A&M University.

Dr. Lombardi has received many professional awards: the Visiting Fellowship at the British Columbia Advanced System

Institute, University of Victoria, Canada (1988), twice the Texas Experimental Engineering Station Research Fellowship (1991–1992, 1997–1998) the Halliburton Professorship (1995), the Outstanding Engineering Research Award at Northeastern University (2004) and an International Research Award from the Ministry of Science and Education of Japan (1993–1999). Dr. Lombardi was the recipient of the 1985/86 Research Initiation Award from the IEEE/Engineering Foundation and a Silver Quilt Award from Motorola-Austin (1996).

Since 2000, Dr. Lombardi is an Associate Editor of the IEEE Design and Test Magazine. He also serves as the Chair of the Committee on "Nanotechnology Devices and Systems" of the Test Technology Technical Council of the IEEE (2003-). In the past, Dr. Lombardi was an Associate Editor (1996–2000) and the Associate Editor-in-Chief (2000–2006) of IEEE Transactions on Computers and twice a Distinguished Visitor of the IEEE-CS (1990–1993 and 2001–2004). Since January 1, 2007 he is the Editor-In-Chief of the IEEE Transactions on Computers.

Dr. Lombardi has been involved in organizing many international symposia, conferences and workshops sponsored by professional organizations as well as guest editor of Special Issues in archival journals and magazines such as IEEE Transactions on Computers, IEEE Transactions on Instrumentation and Measurement, the IEEE Micro Magazine and the IEEE Design & Test Magazine. He is the Founding General Chair of the IEEE Symposium on Network Computing and Applications.

His research Interests are testing and design of digital systems, bio and nano computing, emerging technologies, defect tolerance and CAD VLSI. He has extensively published in these areas and coauthored/edited seven books.