

Crosstalk in QCA Arithmetic Circuits

Faizal Karim, Konrad Walus and André Ivanov

UBC Department of Electrical and Computer Engineering, 2356 Main Mall, Vancouver, BC,
Canada V6T 1Z4

ABSTRACT

Quantum-Dot Cellular Automata (QCA) is one of several proposed computational nanotechnology paradigms that are being investigated as alternatives to CMOS at the nano-scale. QCA has been reported to offer relatively low power consumption, and very high device density. In recent years, several researchers have started investigating relatively complex circuit architectures using QCA. Such design efforts have highlighted the crosstalk problem in QCA and the lack of research in this area. This paper explores the nature of crosstalk in QCA. We show how crosstalk can be amplified due to several parameters including wire length and the distance between adjacent cells. We develop a model and method that allows us to test for crosstalk using a set of test vectors. We also propose a set of cell placement guidelines and design geometries that help to minimize QCA crosstalk in large circuits.

Keywords: Crosstalk, QCA, Arithmetic Circuits

1. INTRODUCTION

As the demand for smaller, more power efficient technology places a “brick wall” on current deep-submicron technology, molecular quantum-dot cellular automata (QCA) - an emerging nano-scale computing paradigm - sees its performance improve as its cell sizes decrease.¹⁻⁴ However, as device geometries scale down, the crosstalk between adjacent cells can create virtual bridging faults. Such faults are logically irreversible and create major problems in high performance circuits. QCA is a computing paradigm that represents binary information in the electron charge configuration in chemical molecules. For this reason, crosstalk in QCA is layout dependent and it is most appropriate to discuss crosstalk risk estimation and reduction with respect to cell placement and routing. To this effect, we present a set guidelines for cell placements from which one can create QCA circuits while reducing the risk of crosstalk between neighbouring cells. Moreover, in this paper, we also present a systematic approach to testing for crosstalk errors that provides 100% fault coverage using a minimal set of test vectors.

While crosstalk has, to date, been a largely under addressed issue in QCA, there has been a wide range of work done with respect to testing of QCA circuits in recent years. Most notably, Lombardi et. al. have investigated test vector generation to detect single stick-at faults (SSF) and bridging faults (BF) in QCA circuits and have suggested test-friendly design architectures that help minimize the number of test vectors required for majority gate based designs.^{5,6} Their work, however, is limited to functional faults in QCA circuits due to cell displacements and deposition defects, and does not discuss in any detail, the effect of crosstalk on the primary outputs.⁷ Gupta et. al. at Princeton University have developed a comprehensive test generation methodology to cover functional faults in QCA circuits, but have also limited their scope to exclusively manufacturing defects.⁸

The paper is organized as follows. In section 2 we provide our justification for examining crosstalk issues by simulating and analysing common scenarios in QCA circuits. In section 3, we develop a bridging fault model that accurately simulates the behaviour of bridging faults in majority gates, and exploit several key features of majority circuits that allow for easy testability. We finish the paper by applying our model on a full adder and provide examples on how to develop a minimal test vector set to achieve 100% fault tolerance.

Further author information:

Faizal Karim: E-mail: faizalk@ece.ubc.ca, Telephone: 1 604 827 3412

Konrad Walus: E-mail: konradw@ece.ubc.ca, Telephone: 1 604 827 5987

André Ivanov: E-mail: ivanov@ece.ubc.ca, Telephone: 1 604 822 6936

2. CROSSTALK ANALYSIS

Crosstalk in QCA occurs when an adjacent cell (or wire) begins to bear more influence on the cells of interest than its own neighbours - forcing the cells to take on an undesired polarization. Once latched, the cells will force other neighbouring cells to its new polarization, thus propagating the error to the primary output. For this reason, it can be shown that all crosstalk in QCA can be modelled as a dominant bridging fault. To support our contention, we set up the following simulations as shown in Figures 1 and 2.

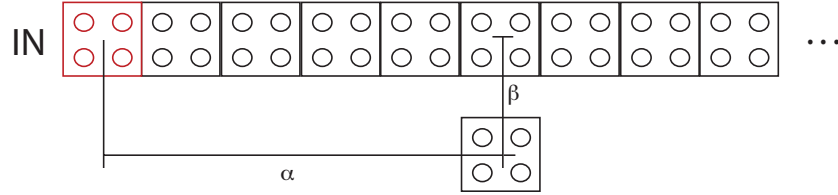


Figure 1. Simulation of Crosstalk due to a Single Cell

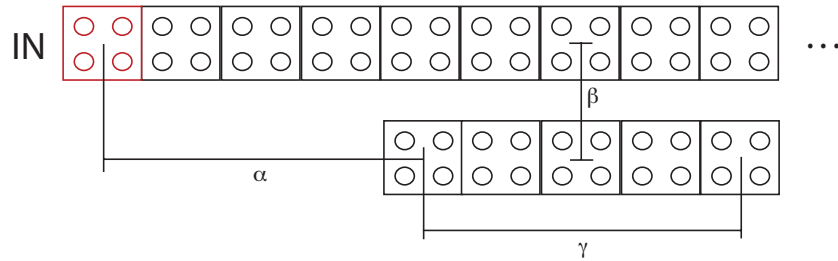


Figure 2. Simulation of Crosstalk due to a Wire

Here, α is the number of cells between the input cell of the local wire to the first cell of the interfering wire, β is the vertical distance between the two wires, and γ is the length of the interfering wire. In both cases, the local wire was 100 cells long and the simulations were performed using the bistable simulation engine in QCADesigner with the parameters shown in Table 1.

Table 1. Parameters used for the Bi-Stable Simulation engine in QCADesigner.

Number of Samples	128000
Convergence Tolerance	10^{-6}
Radius of Effect	1000nm
Max Iterations per Sample	1000
Cell Size	20nm

The simulation for the single cell was conducted by increasing α in increments 5 cells, and then adjusting β until the circuit functioned correctly. The results are shown in Figure 3. For the case of an adjacent wire, we incremented γ from 1 to 10 cells, and then once again, adjusted β until the circuit produced the correct output. The results in Figure 4 show the variance of β with respect to γ for three different α .

It can be seen from Figures 3 and 4 that the further away from the input the interfering cells are, the more susceptible to crosstalk the circuit becomes. This occurs because by the time the input signal propagates to the region of crosstalk, the interfering signal has already perturbed those cells sufficiently so that the desired

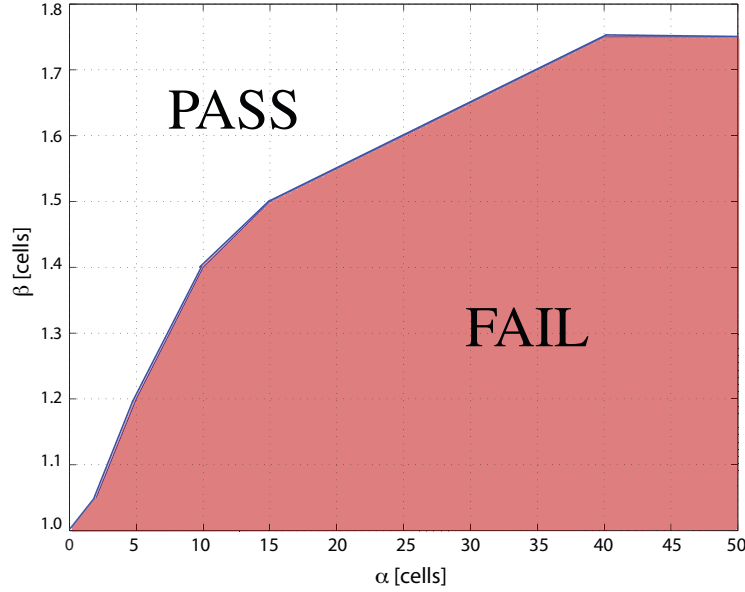


Figure 3. Simulation results for crosstalk due to a single cell.

input cannot correctly polarize them, essentially placing the wire in a meta-stable state. Additionally, the results from Figure 4 suggest that the longer an interfering adjacent wire is, the more likely it is to provide unwanted crosstalk between cells. Here, this occurs because the total kink energy and coupling between wires increases as the number of cells in the interfering wire is increased. In both cases, the crosstalk between the local and interfering cells leads to a dominant bridging fault which propagates to the primary output.

The results yielded from our simulations can also be used to provide a set of basic design rules to help prevent crosstalk in QCA circuits. For a given α , a suitable β can be found from either Figure 3 or 4 to not only ensure proper functionality of the circuit but, at the same time, to maintain a minimally sized design. Similarly, values for γ can be found for any given α or β and vice-versa. We will examine these design dimension issues in more detail in the later sections of this paper.

3. CROSSTALK MODELLING AND TESTING

An example of where crosstalk can become an issue is shown in Figure 5(a). Here, a majority gate is depicted with its output being pulled up alongside input A, creating unwanted crosstalk between them as shown in Figure 5(c). In this scenario, crosstalk cannot be avoided by using the continuous clocking scheme suggested by Lent et. al.^{1, 9, 10}

For the purposes of this paper, we will model all dominant bridging faults as shown in Figure 5(d). We don't distinguish between the bridging faults shown in Figures 5(c) and 5(d), since in both cases they will result in a stuck-at A ($s - a - A$) fault as shown in Figure 6.

Figure 6(a) shows the truth tables of both a correctly functioning majority gate as well as two majority gates exhibiting bridging faults due to input A. Here, it can be seen that the $s - a - A$ fault is easily detected by applying either of the test vectors $\{011, 100\}$ to the input and observing the output. Obtaining a set of input vectors to detect all possible bridging faults proves to be quite easy due to the simplicity of the majority gate.

Consider a majority gate with input lines A, B, and C and an output Y such that $Y = AB + AC + BC$.

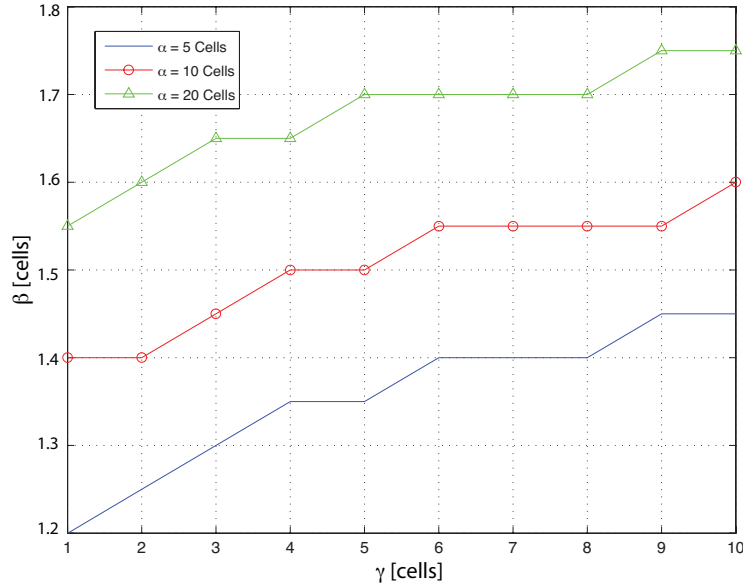


Figure 4. Simulation results for crosstalk due to a wire.

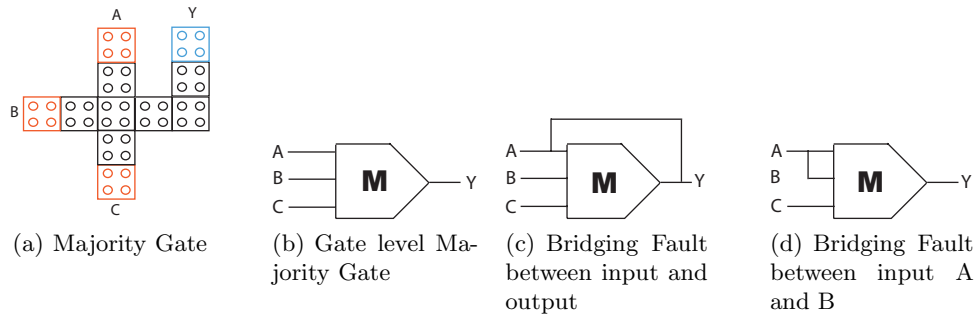


Figure 5. Majority Gates with Crosstalk

Property 1. Let a majority gate, G , have inputs a, b , and c (for input lines A, B and C, respectively) and an output, y . If stuck-at- x ($s - a - x$) is detectable by abc , then $s - a - x$ is also detectable by $a'b'c'$.

This property is particularly useful because for any circuit that is composed exclusively of majority gates, Property 1 suggests that we only require a maximum of 2^{n-1} test vectors, where n represents the number of primary inputs. Indeed we can assume all our circuits are entirely composed of majority gates. If a circuit contains AND/OR gates, these are themselves generated with majority gates for which one of the inputs is fixed. As suggested by Lombardi et. al.,⁵ the fixed cells can be implemented using control lines which can be seen as extra input lines during testing time.

Property 2. Let a majority gate, G , have inputs a, b , and c , and an output y . If inputs a and b of a majority gate are set to opposite values, i.e. $\{10, 01\}$, then y will be equal to the input c .

This property suggests that we can map any input to the output of any majority gate providing that we can manipulate the other two inputs to have opposite binary values, i.e. put the gate at its threshold. For purely majority gate based circuits, the following two properties hold:

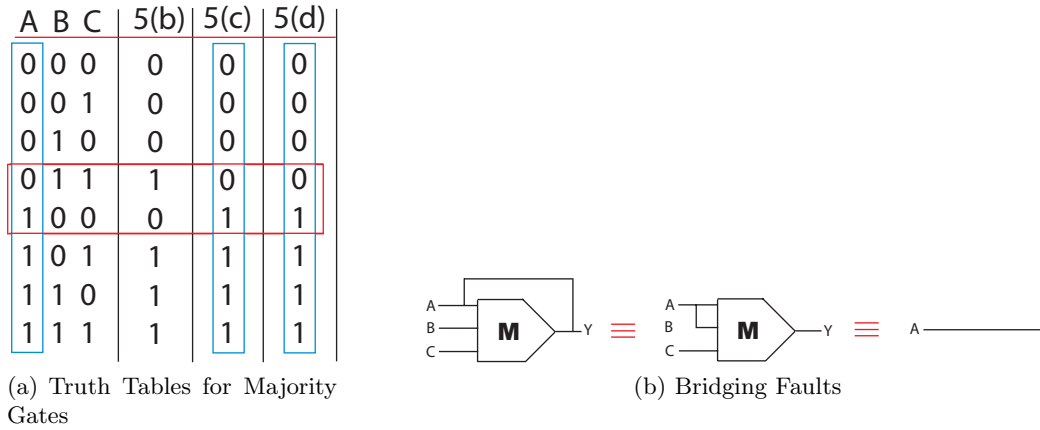


Figure 6. Truth Tables for Functioning and Non-functioning Majority Gates and Diagrams

Property 3. Let a circuit, C , be completely composed of majority gates. If there exists a path in C such that all majority gates in that path cannot be placed on the threshold, then the circuit C is said to be redundant.

Proof. Let C be a circuit composed exclusively of majority gates. Consider a majority gate, G , with the inputs a, b and c and an output, y . Suppose that the inputs a and b are such that they cannot be made opposite, i.e. G cannot be placed on the threshold. Then y will simply be equal to a and b for all abc making G redundant. Therefore, C is a redundant circuit.

Property 4. The converse of Property 3 is true.

Bridging faults at every majority gate can be detected by setting the input of interest to 1 and the other two to 0 such that it is outvoted. For example, to detect a s-a-A fault in the majority gate shown in Figure 5(b), we can input the test vector $\{100\}$ and then check the validity of the output. If the output yields a 1, then the majority gate has a s-a-A fault. If not, then there does not exist a bridging fault due to input A. Since there are only 3 inputs to a majority gate, we require only 3 test vectors - $\{001, 010, 100\}$ - to test for bridging faults within a majority gate. If any of the inputs to the majority gate are inverted, then the corresponding bit in every test vector must also be inverted, i.e. if input B is inverted, then the test vectors become $\{011, 000, 110\}$. These three test vectors however do not consider the potential of bridging between neighbouring majority gates. Consider the layout in Figure 7(a). Here the middle majority gate, M2, is not only at risk of crosstalk amongst its inputs B, C and D , but may also be affected by inputs A and E of majority gates M1 and M3, respectively. In this example, we omit the other inputs in M1 and M3 as they do not play a role in the potential crosstalk between their majority gate and M2. In order to test for bridging faults on B due to A , and on D due to input E , we must introduce a fourth test vector with the input pattern $\{101\}$. We select this vector for its ability to detect the $s-a-A(0)$ and $s-a-E(0)$ bridging faults. As was the case for the other three test vectors, if any of the inputs to M2 are inverted, then so must the bits in this test vector.

Figure 7(b) shows the four test vectors required to test for all potential bridging faults for a given majority gate. It can be seen from the figure that all potential bridging faults - including simultaneous ones - can be detected by the four vectors. Therefore it is suffice to say that for a purely majority circuit, if there are N first level majority gates, then the entire circuit can be tested by using no more than $4N + m$ test vectors where m represents the number of primary inputs not connected to a first level majority gate. Such a result suggests that the size of the test set increases linearly with the number of primary inputs. However, in smaller circuits, or in those that do not have entirely unique inputs, Property 1 may yield fewer test vectors. Therefore, the maximum number of input vectors required can be represented as $\min[4N + m, 2^{n-1}]$.

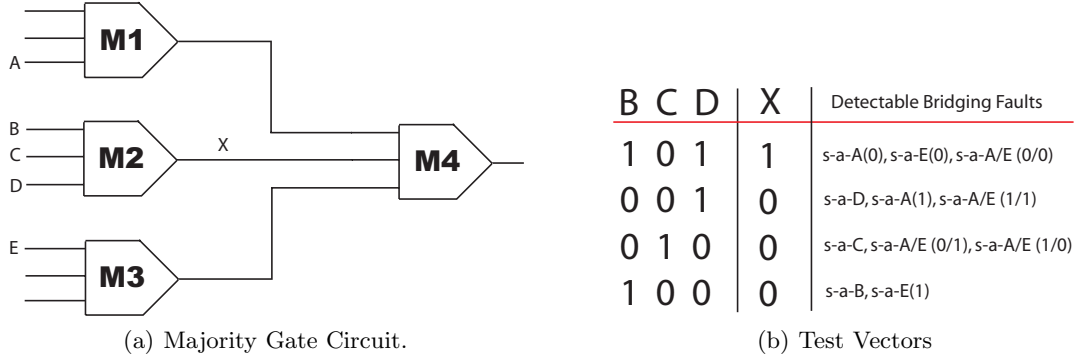


Figure 7. Majority Gate Circuit with Test Vectors

4. FULL ADDER

Here we consider the effects of crosstalk on a full adder and discuss a procedure on how to test for crosstalk errors in the circuit. First, consider the full adder shown in Figure 8(a), created using 20nm cells. In this implementation, we consider the effect of crosstalk between inputs C_{in} and A where C_{in} represents the local wire, and A provides the interference. Here, α , β and γ have all been marked on the figure with α and γ being 5 and 9 cells, respectively. In this example, in order to prevent crosstalk, we refer to Figure 4 and look up β for the given α and γ . According to Figure 4, in order to design a correctly functioning circuit, β must be no smaller than 1.45 cells (29nm). If such a large β is not practical, then one suggestion would be to reduce α until a suitable β can be achieved.

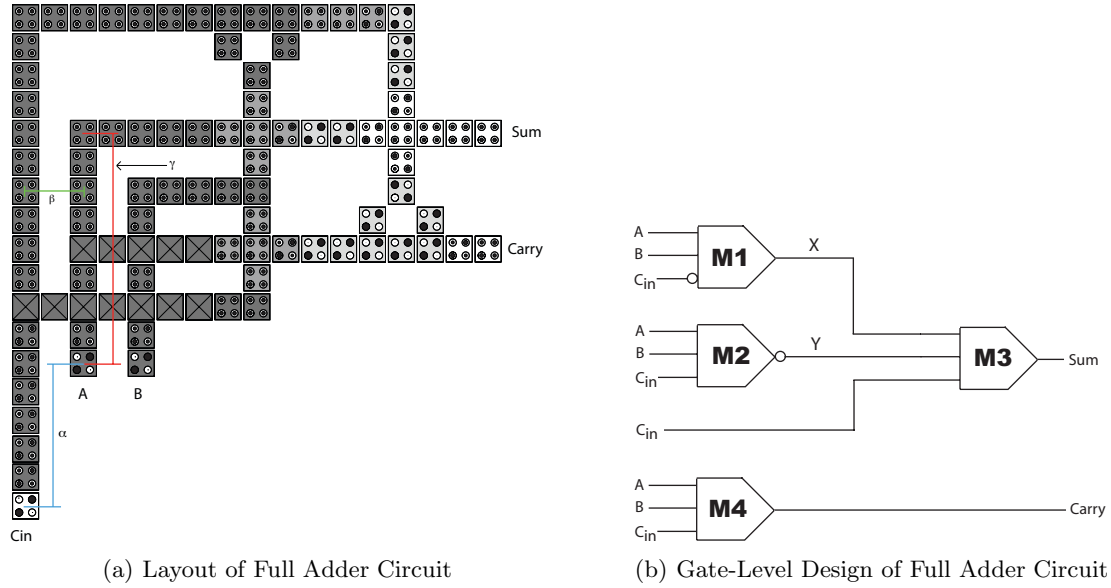
In spite of one's best efforts, there is still the possibility for crosstalk to become an issue due to fabrication imperfections. Therefore, it is important that we can test for all crosstalk issues by using a minimal set of test vectors. Consider the gate-level design of the full adder in Figure 8(b). Here, we test for crosstalk errors in each majority gate individually as shown in Figure 9. In this case, because of the relatively small number of inputs, the maximum number of test vectors required is 4 (2^{3-1}) and so we will only require the top half of the truth table as indicated by Property 1. By quick inspection of Figure 9, we can see that the input vector $\{000\}$ will test for the $s-a-\bar{C}$ fault in majority gate 1, $\{001\}$ will test for $s-a-C$ in majority gate 2, $\{010\}$ will test for $s-a-A$, $s-a-B$ and $s-a-C$ faults for majority gates 1, 2 and 3, respectively, and the input vector $\{011\}$ will test for $s-a-B$, $s-a-A$, and $s-a-C$ in majority gates 1, 2, and 3, respectively. BF detection for the Carry can be performed by using the same test vectors as for majority gate 2.

Now consider the full adder shown in Figure 10(a). This implementation of the full adder introduces a redundant majority gate, M3, in place of the input, C_{in} . Due to the redundancy in the circuit, not all bridging faults can be detected using the four input vectors $\{000, 001, 010, 011\}$ as suggested in Property 1.

As can be seen from Figure 11(a), the $s-a-A$ fault is undetectable by any of the four input vectors. Similarly, the $s-a-B$ and $s-a-C$ bridging faults are also undetectable as seen in Figures 11(b) and 11(c). This occurs because the redundancy in the circuit does not allow us to place all majority gates in their paths at the threshold (Properties 3 and 4). Therefore, removing all redundancies in the circuit becomes critical in our ability to detect the bridging faults in a circuit. Removing the redundancy in a circuit is fairly trivial however, as it can be shown that any redundant gate can simply be replaced by the the input whose bridging fault cannot be detected, i.e. M1 can be replaced with a wire A , M2 can be replaced by $\bar{B}$, or M3 by C_{in} as in Figure 8(c).

5. CONCLUSION

In this paper, we developed a model and method to test for crosstalk in QCA circuits and provided a set of design rules to help prevent the presence of crosstalk in arithmetic circuits. First, we showed that all crosstalk errors can be modelled as dominant bridging faults. Based on this model, we developed a minimal set of test vectors to



A	B	C	X	Y	S
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	0	0
1	1	1	1	0	1

(c) Truth Table for Full Adder

Figure 8. Layout and Gate-Level Design of Full Adder Circuit with Truth Table

A	B	C	$s-a-A$	$s-a-B$	$s-a-\bar{C}$	S
0	0	0	0	0	1	0
0	0	1	1	1	1	1
0	1	0	0	1	1	1
0	1	1	0	1	0	0

(a) Truth Table for Majority Gate 1

A	B	C	$s-a-A$	$s-a-B$	$s-a-C$	S
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	1	0	0	0

(b) Truth Table for Majority Gate 2

A	B	C	$s-a-C$	S
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	1	0

(c) Truth Table for Majority Gate 3

Figure 9. Truth Tables for Full Adder

exhaustively test the functionality of any 3-input majority gate. Next, we demonstrated that, for non-redundant majority circuits, we could sensitize any path from a given primary input to the primary output. Using this property, we arrived at the conclusion that the maximum number of test vectors required to exhaustively test a purely majority circuit was equal to $\min[4N + m, 2^{n-1}]$, where N is the number of first level majority gates, m is the number of primary inputs not connected to a first level majority gate, and n is the total number of primary

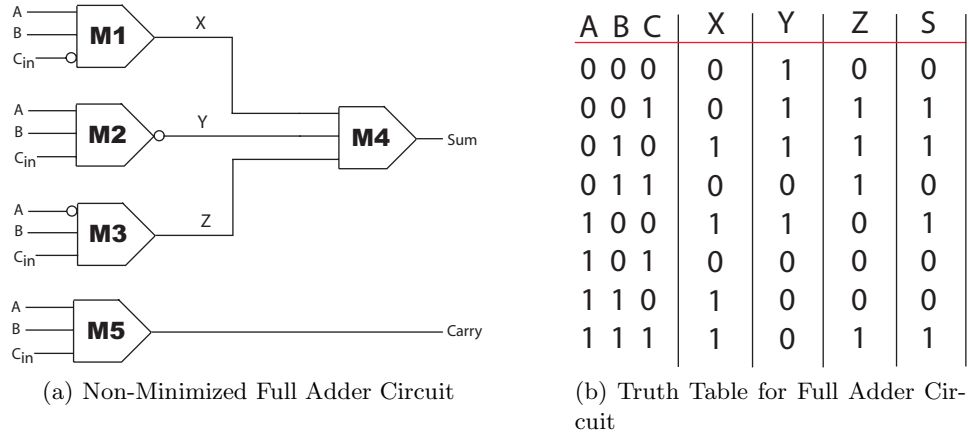


Figure 10. Full Adder Circuit and Truth Table

A	B	C	$s-a-A$	$s-a-B$	$s-a-\bar{C}$	S
0	0	0	0	0	1	0
0	0	1	1	1	1	1
0	1	0	1	1	1	1
0	1	1	0	1	0	0

(a) Truth Table for Majority Gate 1

A	B	C	$s-a-A$	$s-a-B$	$s-a-C$	S
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	1	0	0	0

(b) Truth Table for Majority Gate 2

A	B	C	$s-a-\bar{A}$	$s-a-B$	$s-a-C$	S
0	0	0	1	0	0	0
0	0	1	1	0	1	1
0	1	0	1	1	1	1
0	1	1	0	0	0	0

(c) Truth Table for Majority Gate 3

Figure 11. Truth Tables for Full Adder

inputs. With an example, we showed what minimum spatial requirements existed when designing a full adder, as well as provided a set of test vectors to exhaustively test the circuit for crosstalk errors.

ACKNOWLEDGMENTS

The authors acknowledge financial support from the University of British Columbia in Vancouver, Canada.

REFERENCES

1. K. Hennessy and C. S. Lent, "Clocking of molecular quantum-dot cellular automata," *J. of Vacuum Science and Technology B: Microelectronics and Nanometer Structures* **19**, pp. 1752–1755, 2001.
2. C. S. Lent, B. Isaksen, and M. Lieberman, "Molecular quantum-dot cellular automata," *J. Am. Chem. Soc.* **125**, pp. 1056–1063, 2003.
3. M. L. et. al., "Quantum-dot cellular automata at a molecular scale," *Ann. N.Y. Acad. Sci.* **960**, pp. 225–239, 2002.
4. G. Toth, *Correlation and Coherence in Quantum-Dot Cellular Automata*, 2000. Ph.D Thesis.

5. M. B. Tahoori, J. Huang, M. Momenzadeh, , and F. Lombardi, "Testing of quantum cellular automata," *IEEE Transactions on Nanotechnology* **3**, pp. 432–442, 2004.
6. M. B. Tahoori, M. Momenzadeh, J. Huang, and F. Lombardi, "Defects and faults in quantum cellular automata at nano scale," *Proc. of the 22nd IEEE VTS*, pp. 291–296, 2004.
7. M. Momenzadeh, M. Ottavi, , and F. Lombardi, "Modeling qca defects at molecular-level in combinational circuits," *Proc. of the 20th IEEE DFT*, pp. 208–216, 2005.
8. P. Gupta, N. K. Jha, , and L. Lingappan, "Test generation for combination quantum cellular automata (qca) circuits," *Proc. Design, Automation and Test in Europe*, pp. 311–316, 2006.
9. K. Walus, G. Schulhof, , and G. A. Julien, "Implementation of a simulation engine for clocked molecular qca," *Proc. in Calgary*, 2006.
10. C. S. Lent, "Quantum cellular automata," *Nanotechnology* **4**, pp. 49–57, 1993.