A Method of Majority Logic Reduction for Quantum Cellular Automata

Rumi Zhang, Konrad Walus, *Student Member, IEEE*, Wei Wang, *Member, IEEE*, and Graham A. Jullien, *Fellow, IEEE*

***Abstract*—The basic Boolean primitive in quantum cellular automata (QCA) is the majority gate. In this paper, a method for reducing the number of majority gates required for computing three-variable Boolean functions is developed to facilitate the conversion of sum-of-products expression into QCA majority logic. Thirteen standard functions are introduced to represent all three-variable Boolean functions and the simplified majority expressions corresponding to these standard functions are presented. We describe a novel method for using these standard functions to convert the sum-of-products expression to majority logic. By applying this method, the hardware requirements for a QCA design can be reduced. As an example, a 1-bit QCA adder is constructed with only three majority gates and two inverters. The adder is designed and simulated using QCADesigner, a design and simulation tool for QCA. We will show that the proposed method is very efficient and fast in deriving the simplified majority expressions in QCA design.**

***Index Terms*—Boolean function, majority expression, majority reduction, quantum cellular automata (QCA) adders, three cube.**

I. Introduction

QUANTUM cellular automata (QCA) is a nanotechnology that has recently been recognized as one of the top six emerging technologies with potential applications in future computers [1]–[6]. Several studies have reported that QCA can be used to design general-purpose computational and memory circuits [7]–[10]. First proposed in 1993 by Lent *et al.*, and experimentally verified in 1997, QCA is expected to achieve high device density, extremely low power consumption, and very high switching speed.

The fundamental QCA logic primitives are the three-input majority gate, wire, and inverter. Each of these can be considered as a separate QCA locally interconnected structure, where QCA digital architectures are combinations of these cellular automata structures. Traditional logic reduction methods [11]–[14], such as Karnaugh maps (K-maps), always produce simplified expressions in the two standard forms: sum of products (SOP) or product of sums (POS). However, we will encounter difficulties in converting these two forms into majority expressions due to the complexity of multilevel majority gates. In CMOS/silicon design, the logic circuits are usually implemented using AND and OR gates based on SOP or POS formats. However, since QCA logic is based on a majority gate primitive, it is critical that an efficient technique be established for designing with this primitive.

In this paper, we develop a Boolean algebra based on a geometrical interpretation of three-variable Boolean functions to facilitate the conversion of sum-of-products expressions into reduced majority logic. Thirteen standard functions are introduced, which represent all possible three-variable Boolean functions. For each of these standard functions, we present the reduced majority expression. As an example of this technique, we present a QCA adder design, and show that the proposed method is able to reduce the total hardware, as compared to previously published designs.

This paper is organized as follows. In Section II, we provide a brief background to QCA technology. In Section III, we develop a Boolean algebra based on a geometrical interpretation of three-variable Boolean functions to facilitate the conversion of sum-of-products expressions into reduced majority logic. Thirteen standard functions are introduced to represent all three-variable Boolean functions and the simplified majority expressions corresponding to these standard functions are given. A procedure to convert a generic three-variable Boolean function into majority gates is also developed. In Section IV, as an example, we introduce a 1-bit QCA adder constructed with only three majority gates and two inverters. We conclude this paper in Section V.

II. Background Material

A. QCA Basics

QCA technology is based on the interaction of bi-stable QCA cells constructed from four quantum dots. The cell is charged with two free electrons, which are able to tunnel between adjacent dots. These electrons tend to occupy antipodal sites as a result of their mutual electrostatic repulsion. Thus, there exists two equivalent energetically minimal arrangements of the two electrons in the QCA cell, as shown in Fig. 1. These two arrangements are denoted as cell polarization $P = +1$ and $P = -1$. By using cell polarization $P = +1$ to represent logic "1" and $P = -1$ to represent logic "0," binary information is encoded in the charge configuration of the QCA cell.

B. QCA Logic Devices

The fundamental QCA logic primitives include a QCA wire, QCA inverter, and QCA majority gate [15]–[18], as described below.

Manuscript received February 18, 2004; revised April 9, 2004. This work was supported in part by research grants from the Natural Sciences and Engineering Research Council of Canada, Micronet (a Network of Centres of Excellence), and iCORE (Alberta).

R. Zhang and W. Wang are with the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON N6G 1H1, Canada (e-mail: zhangrumi@hotmail.com; wwang@eng.uwo.ca).

K. Walus and G. A. Jullien are with the Advanced Technology Information Processing Systems Laboratory, Department of Electrical and Computer Engineering, University of Calgary, 2 Calgary, AB T2N 1N4, Canada (e-mail: walus@atips.ca; jullien@atips.ca).

Digital Object Identifier 10.1109/TNANO.2004.834177

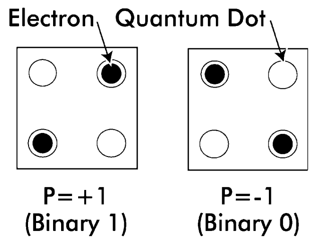


Fig. 1. QCA cells showing how binary information is encoded in the two fully polarized diagonals of the cell.

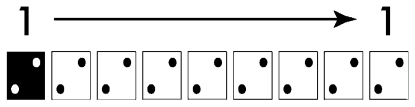


Fig. 2. QCA wire (90°).

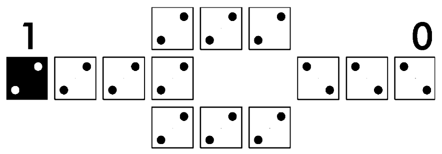


Fig. 3. QCA inverter.

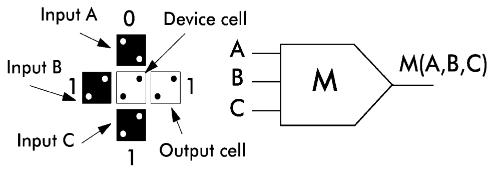


Fig. 4. QCA majority gate.

QCA Wire: In a QCA wire, the binary signal propagates from input to output because of the electrostatic interactions between cells. The propagation in a 90° QCA wire is shown in Fig. 2. Other than the 90° QCA wire, a 45° QCA wire can also be used. In this case, the propagation of the binary signal alternates between the two polarizations.

QCA Inverter: A QCA layout of an inverter circuit is shown in Fig. 3. Cells oriented at 45° to each other take on opposing polarization. This orientation is exploited here to create the inverter shown in this figure.

QCA Majority Gate: The QCA majority gate performs a three-input logic function. Assuming the inputs are A, B, and C, the logic function of the majority gate is

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

A layout of a QCA majority gate is shown in Fig. 4. The tendency of the majority device cell to move to a ground state ensures that it takes on the polarization of the majority of its neighbors. The device cell will tend to follow the majority polarization because it represents the lowest energy state.

By fixing the polarization of one input to the QCA majority gate as logic "1" or logic "0," an AND gate or OR gate will be obtained, respectively, as follows:

$$M(A,B,0) = AB \tag{2a}$$
$$M(A,B,1) = A + B. \tag{2b}$$

Thus, we can base all QCA logic circuits on three-input majority gates. In order to achieve efficient QCA design, majority gate-based design techniques are required. In the literature, the study of majority gates mainly focuses on the implementations of the n-input majority function [19], [20] using threshold logic. An n-input majority gate produces a logic "1" output if the majority of its inputs are logic "1"; two out of three in the case of a three-input majority function. If n is even, $n/2+1$ inputs must be at logic "1" to produce an output of logic "1." The three-input QCA majority gate is a special case of the n-input majority function. Currently, there is evidence from the latest fabrication technology that feed-forward networks, constructed with threshold gates, are becoming a promising solution for computer arithmetic [19], [20]. However, there does not appear to be any published study on the implementation of logic functions using minimal numbers of three-input majority gates. Although the K-maps method provides a simple and straightforward procedure for minimizing Boolean functions, the simplified expressions produced by K-maps do not guarantee a simplified majority expression. For example, a full-adder carry function requires three AND and two OR gates, but can be created with just one three-input majority gate. It is expected that Boolean algebra based on majority gates will be significantly important for future QCA design techniques.

III. Boolean Algebra Based on Majority Gates

Here, we develop a Boolean algebra of three variables to facilitate the conversion of a sum-of-products expression to minimized majority logic. We first apply a geometric interpretation of the Boolean function, namely the three-cube method [21], [22] to find 13 standard functions to represent all three-variable Boolean functions. We then propose simplified majority expressions for each of these 13 functions along with a procedure to obtain the efficient majority expression for any given three-variable Boolean function amenable to QCA implementation.

A. Three-Cube Representation of Three-Variable Boolean Function

According to [21], [22], a Boolean function of n variables can be represented by a binary n cube; i.e., an n-dimensional binary hyper-cube. Each literal is represented by one of the dimensions of the hyper-cube with coordinates of "0" or "1." Here, we apply the method [21], [22] for three-variable Boolean functions in the three-cube domain. For Boolean functions of three binary variables, i.e., $a, b,$ and c, one can obtain 2^3 distinct minterms: $\bar{a}\bar{b}\bar{c}$, $\bar{a}\bar{b}c$, $\bar{a}b\bar{c}$, $\bar{a}bc$, $a\bar{b}\bar{c}$, $a\bar{b}c$, $ab\bar{c}$, and abc, each corresponding to a vertex (point) of the three cube. For example, $a\bar{b}c$ has the coordinates $(1,0,1)$. A three-variable function is represented in a three-dimensional space. This function space is defined by a set of points, called the *on-set* points, for which the function evaluates to "1." The set of points corresponding to the function evaluating to "0" is called the *off-set*.

There are several ways to write a given Boolean function as a SOP. Each representation is called a *cover* because it covers all the points in the function's *on-set*. Each member of the cover is a subspace of the function space, which is written algebraically as a product (or cube). A function has many covers, some of

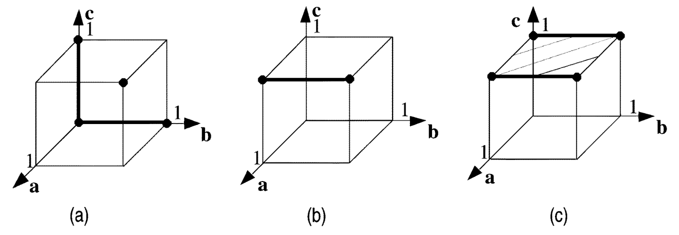


Fig. 5. Three-cube structure of three-variable Boolean functions. (a) Three-cube structure. (b) Edge. (c) Face.

which have more literals than others. The minimum cover of a function is a cover with both fewest cubes and fewest literals.

Example 1: The three-cube structure for the function $F = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc$ is given in Fig. 5(a). Between the two covers of Fig. 5(a), $\bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc$ and $\bar{a}\bar{b} + \bar{a}\bar{c} + abc$, the cover $\bar{a}\bar{b} + \bar{a}\bar{c} + abc$ is the minimum cover.

We use the following example to show the procedure to obtain the minimum cover by using the geometrical characteristics of the three cube.

Example 2: In Fig. 5(b), the vertices of $abc + a\bar{b}c$ form a one-dimensional subspace of the function, described by an edge $abc + a\bar{b}c = ac$. Furthermore, in Fig. 5(c), the two edges ac and $\bar{a}c$ can be further simplified in a two-dimensional subspace, described by a face $ac + \bar{a}c = c$. Using such strategies, we can obtain the minimum cover, i.e., the simplest sum-of-products expression.

K-maps can also achieve the same simplification. The advantage of using the graphical representation of a Boolean function in the three-cube structure is that it clearly shows the possible distribution of the sum of any minterms. Thus, we can easily use the three-cube method to find standard functions for all three-variable Boolean functions.

B. Proposed 13 Standard Three-Variable Boolean Functions

Three binary variables A, B, and C can only produce eight unique minterms. Any three-variable Boolean function can be represented by the combinations of up to eight of these minterms. Since each minterm corresponds to a vertex (point) of the three cube, the three-variable Boolean function can be represented as points on the three cube. If we are only concerned with the relationships associated with these points, we can obtain 13 possible structures and corresponding functions for the cases of one to four points on a three cube, as shown in Table I.

We use the case of two points as an example to illustrate how to obtain the structures and functions in Table I. The possible relationships between the two points are: 1) two adjacent points (one edge); 2) two nonadjacent points, but in one face; and 3) two nonadjacent points, and not in one face. The three structures correspond to $F = AB, F = ABC + A\bar{B}\bar{C}$, and $F = ABC + \bar{A}\bar{B}\bar{C}$, respectively. It is noted that the Boolean variables A, B, and C can be mapped to any one of $a, b, c, \bar{a}, \bar{b}$, and $\bar{c}$. Thus, these three functions can represent all the two points cases; in other words, all three-variable Boolean functions of two minterms. Following a similar approach, we find 13 standard structures and functions for all the cases from one to four points on a three cube.

The three-variable Boolean function of 5–7 minterms (points) can be represented using the complement form of 3–1 minterms. Based on DeMorgan's theorem, a Boolean function, expressed as the sum of several minterms, can also be expressed as the complement of the sum of the remaining minterms. For example, the function F

$$F = \sum(2,4,5,6,7) = \overline{\sum(0,1,3)}. \tag{3}$$

Furthermore, a three-variable Boolean function of eight minterms (points on a three cube) is always "1." Thus, we have the following mappings for 5–7 minterms, based on the results from 3–1 minterms, along with the special case for eight minterms:

$$\begin{aligned}
\text{Five minterms:}\quad & F = \overline{\text{Three minterms}}\\
\text{Six minterms:}\quad & F = \overline{\text{Two minterms}}\\
\text{Seven minterms:}\quad & F = \overline{\text{One minterm}}\\
\text{Eight minterms:}\quad & F = 1.
\end{aligned}$$

Clearly, the cases representing 5–8 minterms can be mapped back to our 13 standard functions.

Since the 13 standard functions are obtained by using the above exhaustive search method, we can conclude that any three-variable Boolean function can be converted into one of these 13 standard functions. Consider, as an example, the following Boolean function.

Example 3:

$$\begin{aligned}
F &= \bar{a}c + \bar{a}b + a\bar{b}c + bc\\
&= \bar{a}\bar{b}c + \bar{a}bc + \bar{a}b\bar{c} + a\bar{b}c + abc; \quad \text{5 minterms}\\
&= c + \bar{a}b; \quad \text{simplified}\\
&= \overline{a\bar{c} + \bar{b}\bar{c}}; \quad \overline{\text{3 minterms}}\\
&= \overline{AB + BC}; \quad \text{Let } (A,B,C) = (a,\bar{c},\bar{b})\\
&= \overline{\text{Function 5 in Table I.}}
\end{aligned}$$

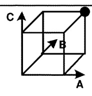
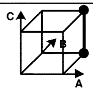
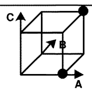
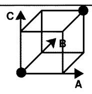
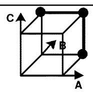
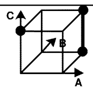
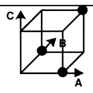
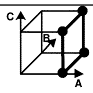
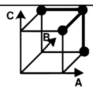
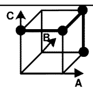
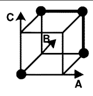
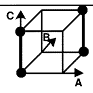
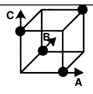
It is shown in [23] and [24] that there are 256 Boolean functions for three variables, but that they can be mapped to only 80 equivalence classes by permuting their inputs to produce the same functionality. The 13 standard functions, as discussed above, can be considered as a further simplification of the methods of [23]and [24] with the added functionality of efficiently converting three-variable Boolean functions to their majority gate equivalences.

C. Proposed Majority Gate Representation of 13 Standard Functions

The simplified majority expressions for 13 standard functions are given in Table II.

1) Functions 1, 2, 4, 6, 8, 10, and 12 are obtained directly by converting the sum-of-products expressions using (2a) and (2b).

TABLE I
13 Standard Functions

Number of points	3-Cube Structure	Description	Function No.	Standard Function
One point		one point	1	$F = ABC$
Two points		two points: one edge	2	$F = AB$
		two points: nonajacent, but in one face	3	$F = ABC + A\bar{B}\bar{C}$
		two points: nonadjacent, and not in one face	4	$F = ABC + \bar{A}\bar{B}\bar{C}$
Three points		three points: two edges sharing one common point	5	$F = AB + BC$
		three points: one edge and one nonadjacent point	6	$F = AB + \bar{A}\bar{B}C$
		three points: all nonadjacent	7	$F = ABC + \bar{A}B\bar{C} + A\bar{B}\bar{C}$
Four points		four points: one face	8	$F = A$
		four points: three edges share one common point	9	$F = AB + BC + AC$
		four points: two edges perpendicular to each other	10	$F = AB + \bar{B}C$
		four points: two edges sharing one common point and one nonadjacent point	11	$F = AB + BC + \bar{A}\bar{B}\bar{C}$
		four points: two parallel edges but not in a face	12	$F = AB + \bar{A}\bar{B}$
		four points: all nonadjacent	13	$F = ABC + \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}B\bar{C}$

2) For functions 5 and 11, we use the following distributive law to reduce the number of majority gates involved:

$$AB + BC = B(A + C). \tag{4}$$

Function 5: the direct conversion will require three majority gates and is now reduced to two majority gates

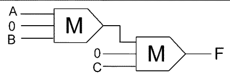
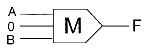
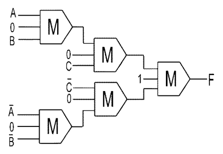
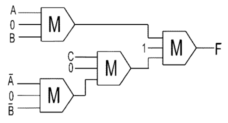
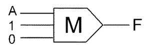
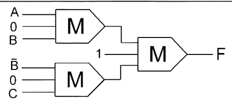
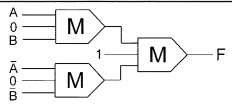
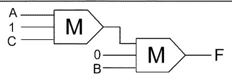
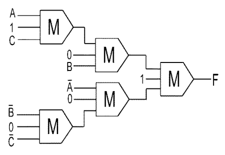
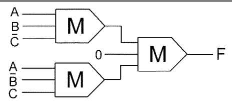
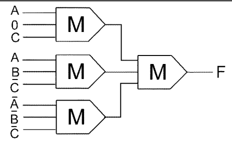
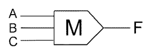
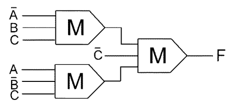
$$F = AB+BC = B(A+C) = M(B, M(A, C, 1), 0) \tag{5a}$$

Function 11: the direct conversion will require six majority gates and is now reduced to five majority gates

$$\begin{aligned} F &= AB + BC + \bar{A}\bar{B}\bar{C} \\ &= B(A + C) + \bar{A}\bar{B}\bar{C} \\ &= M(M(B, M(A, C, 1), 0) \\ &\quad M(\bar{A}, M(\bar{B}, \bar{C}, 0), 0), 1). \end{aligned} \tag{5b}$$

3) For Functions 3, 7, 9, and 13, majority gate reduction techniques are used to reduce the number of majority gates required.

TABLE II
MAJORITY EXPRESSION OF 13 STANDARD FUNCTIONS

Function No.	Standard Function	Majority Expression	Diagram
1	$F = ABC$	$M(M(A,B,0),C,0)$	
2	$F = AB$	$M(A,B,0)$	
4	$F = ABC + \bar{A}\bar{B}\bar{C}$	$M(M(M(A,B,0),C,0), M(M(\bar{A},\bar{B},0),\bar{C},0),1)$	
6	$F = AB + \bar{A}\bar{B}C$	$M(M(A,B,0), M(M(\bar{A},\bar{B},0),C,0),1)$	
8	$F = A$	$M(A,0,1)$	
10	$F = AB + \bar{B}C$	$M(M(A,B,0), M(\bar{B},C,0),1)$	
12	$F = AB + \bar{A}\bar{B}$	$M(M(A,B,0), M(\bar{A},\bar{B},0),1)$	
5	$F = AB + BC$	$M(B,M(A,C,1),0)$	
11	$F = AB + BC + \bar{A}\bar{B}\bar{C}$	$M(M(B,M(A,C,1),0), M(\bar{A},M(\bar{B},\bar{C},0),0),1)$	
3	$F = ABC + A\bar{B}\bar{C}$	$M(M(A,B,\bar{C}), M(A,\bar{B},C),0)$	
7	$F = ABC + \bar{A}B\bar{C} + A\bar{B}\bar{C}$	$M(M(A,C,0), M(A,B,\bar{C}),M(\bar{A},\bar{B},\bar{C}))$	
9	$F = AB + BC + AC$	$M(A,B,C)$	
13	$F = ABC + \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}B\bar{C}$	$M(M(\bar{A},B,C), M(A,\bar{B},C),\bar{C})$	

Function 3: the direct conversion will require five majority gates and is now reduced to three majority gates

$$\begin{aligned} F &= ABC + A\bar{B}\bar{C} \\ &= M(A, M(M(B,C,0), M(\bar{B},\bar{C},0),1),0) \\ &= M(M(A,B,\bar{C}), M(A,\bar{B},C),0). \end{aligned} \tag{5c}$$

Function 7: the direct conversion will require eight majority gates and is now reduced to four majority gates

$$\begin{aligned} F &= ABC + \bar{A}B\bar{C} + A\bar{B}\bar{C} \\ &= M(M(B, M(M(A,C,0), M(\bar{A},\bar{C},0),1),0), \\ &\quad M(M(A,\bar{B},0),\bar{C},0),1) \\ &= M(M(A,C,0), M(A,B,\bar{C}), M(\bar{A},\bar{B},\bar{C})). \end{aligned} \tag{5d}$$

Function 9: the direct conversion will require five majority gates and is now reduced to one majority gate

$$\begin{aligned} F &= AB + BC + AC \\ &= M(M(B, M(A,C,1),0), M(A,C,0),1) \\ &= M(A,B,C). \end{aligned} \tag{5e}$$

Function 13: the direct conversion will require 11 majority gates and is now reduced to three majority gates

$$\begin{aligned} F &= ABC + \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}B\bar{C} \\ &= M(M(A, M(M(B,C,0), M(\bar{B},\bar{C},0),1),0), \\ &\quad M(\bar{A}, M(M(\bar{B},C,0), M(B,\bar{C},0),1),0),1) \\ &= M(M(\bar{A},B,C), M(A,\bar{B},C),\bar{C}). \end{aligned} \tag{5f}$$

Tables I and II will enable us to convert any Boolean function of three variables into a simplified majority gate expression.

D. Proposed Procedure to Build Majority Expressions

Here, we propose a procedure to build simplified majority expressions for a given Boolean function using Tables I and II. The procedure uses the following algorithm to generate a majority gate expression of the three-variable Boolean function F.

```
Algorithm 1
1) Map $F$ onto on-set points in the three-cube structure.
   For five or more minterms, use the complement $\bar{F}$,
   which is to employ the complement of the function
   $\bar{F}$, which is plotted as off-set points.
2) Locate the corresponding format in the 13 stan-
   dard functions for $F$ or $\bar{F}$.
3) Using Tables I and II, look up the simplified majority
   expression for the function $F$.
4) Reduce, if possible, the number of inverter gates by
   using the following relationships:
       $M(\bar{A},\bar{B},\bar{C}) = \overline{M(A,B,C)}$
       $M(\bar{A},\bar{B},0) = \overline{M(A,B,1)}$
       $M(\bar{A},\bar{B},1) = \overline{M(A,B,0)}$
```

We illustrate the application of the procedure using the following example.

Example 4: Derive the simplified majority expression for the function $F = \bar{a}c + \bar{a}b + a\bar{b}c + bc$.

On-set points (black points) and *off-set* points (grey points) of function F, mapped to a three cube, are shown in Fig. 6 as five minterms. In Fig. 6, $\bar{F}$ represents three minterms (grey points) consisting of two edges, which share one common point and corresponds to function 5 in Tables I and II as follows:

$$\begin{aligned} \bar{F} &= M(M(A,C,1),B,0); \quad \text{Using function 5} \\ &= M(M(a,\bar{b},1),\bar{c},0); \quad \text{Let } (A,B,C) = (a,\bar{c},\bar{b}) \\ F &= M(\bar{a},b,0),c,1). \end{aligned}$$

IV. QCA Adders

Here, we will apply the proposed majority reduction method to design QCA adders. We will show that the proposed method is very efficient and fast in deriving simplified majority expressions for QCA design.

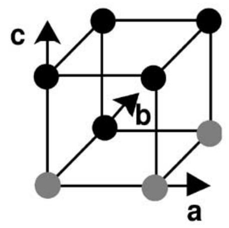


Fig. 6. Three-cube structure of the function in Example 4.

A. QCA Addition Algorithm

A 1-bit full adder is defined as follows.

Inputs: Operand bits a and b and carry-in c_{in}.
Outputs: Sum bit s and carry-out c_{out}.

$$s = abc_{\text{in}} + \bar{a}\bar{b}c_{\text{in}} + \bar{a}b\overline{c_{\text{in}}} + a\bar{b}\overline{c_{\text{in}}} \tag{6a}$$

$$c_{\text{out}} = ab + ac_{\text{in}} + bc_{\text{in}}. \tag{6b}$$

Proposition 1: By using (1) and function 13 in Tables I and II, we obtain the simplified QCA addition majority gate expressions as follows:

$$c_{\text{out}} = M(a,b,c_{\text{in}}) \tag{7a}$$

$$\overline{c_{\text{out}}} = M(\bar{a},\bar{b},\overline{c_{\text{in}}}) \tag{7b}$$

$$s = M(\overline{c_{\text{out}}}, c_{\text{in}}, M(a,b,\overline{c_{\text{in}}})). \tag{7c}$$

Proof: By using (1) and (6b), we obtain

$$\begin{aligned} c_{\text{out}} &= ab + ac_{\text{in}} + bc_{\text{in}} = M(a,b,c_{\text{in}}) \\ \overline{c_{\text{out}}} &= \bar{a}\bar{b} + \bar{a}\overline{c_{\text{in}}} + \bar{b}\overline{c_{\text{in}}} = M(\bar{a},\bar{b},\overline{c_{\text{in}}}). \end{aligned}$$

We find that (6a) will be four nonadjacent points (four minterms) when plotted on a three cube, which matches function 13 in Tables I and II. Thus, (6a) can be rewritten as

$$\begin{aligned} s &= abc_{\text{in}} + \bar{a}\bar{b}c_{\text{in}} + \bar{a}b\overline{c_{\text{in}}} + a\bar{b}\overline{c_{\text{in}}} \\ &= M(M(\bar{a},\bar{b},\overline{c_{\text{in}}}), \\ &\quad M(a,b,\overline{c_{\text{in}}}), c_{\text{in}}); \quad \text{Let } (A,B,C) = (a,\bar{b},\overline{c_{\text{in}}}) \\ &= M(\overline{c_{\text{out}}}, c_{\text{in}}, M(a,b,\overline{c_{\text{in}}})); \quad \text{Using (7b).} \end{aligned}$$

Q.E.D.

It is noteworthy that, using a similar procedure, we can obtain two other equations to compute s as follows:

$$s = M(\overline{c_{\text{out}}}, a, M(\bar{a},b,c_{\text{in}})); \quad \text{Let } (A,B,C) = (c_{\text{in}},\bar{b},\bar{a}) \tag{8a}$$

$$s = M(\overline{c_{\text{out}}}, b, M(a,\bar{b},c_{\text{in}})); \quad \text{Let } (A,B,C) = (\bar{a},c_{\text{in}},\bar{b}). \tag{8b}$$

Equation (8a) is just a variant of (7c) by switching the signals a and c_{in}, while (8b) switches b with c_{in}. These two equations are the same as (7c) in terms of designing the QCA adder.

Based on Proposition 1, the 1-bit QCA adder is shown in Fig. 7. The calculation of c_{out} involves one majority gate and the calculation of s involves two majority gates and two inversion operations $\overline{c_{\text{out}}}$ and $\overline{c_{\text{in}}}$. Thus, the adder only requires three majority gates and two inverters. In contrast, in the original QCA addition algorithm [25], the calculation of c_{out} also

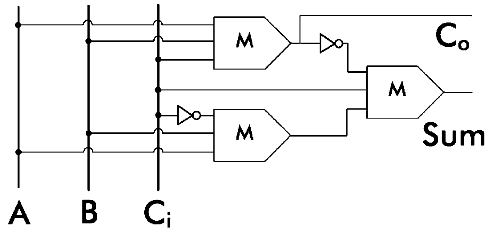


Fig. 7. 1-bit QCA full adder (three majority gates and two inverters).

TABLE III
COMPARISON OF QCA ADDERS

QCA Adders	Complexity of one-bit adder	Clocking	Speed
Our Reduction Method	*Three majority gates and two inverters*	*Simple*	*Fast*
Previous Design [25]	*Five majority gates and three inverters*	*Simple*	*Fast*

requires only one majority gate, but the computation of s is considerably more complex. The calculation of s is based on

$$s = M(M(\bar{a}, b, c_{\text{in}}), \quad M(a, \bar{b}, c_{\text{in}}), \quad M(a, b, \overline{c_{\text{in}}})). \tag{9}$$

This requires four majority gates and three inversion operations $\bar{a}, \bar{b}$, and $\overline{c_{\text{in}}}$. In contrast, the original 1-bit adder design requires five majority gates and three inverters. We note that a bit-serial adder [26] proposed by Fijany *et al.* in 2003 also only uses three majority gates, but our three-gate design does not have the bit-serial restriction.

B. QCA Adder Design

This 1-bit QCA adder consists of three majority gates and two inverters, as shown in Fig. 7. As noted above, this represents a considerable reduction in hardware compared to the original design [25] and retains the simple clocking scheme.

The performance data of the QCA adder and of those considered in [25] are given in Table III.

C. Simulation Results

The QCA adders were designed and simulated using QCADesigner, a layout and simulation tool for QCA logic, developed at the University of Calgary, Calgary, AB, Canada [27], [28]. The design and simulation procedure is as follows. We first generate the layout of the 1-bit QCA adder, and then we setup the circuit clocking. Finally, we set up the vector table simulation to simulate the adder.

1-bit Adder Design and Simulation: The layout of the 1-bit QCA adder is shown in Fig. 8.

Performance Comparison: To maintain consistency with size measurements in [25], we assume that the QCA cells are made of 2-nm quantum dots. The cells are separated by 10 nm.

This design requires only 36% of the area used by the original adder design [25], while maintaining the same clocking performance, as shown in Table IV. The decrease in area requirement is consistent with our theoretical predictions, which we have proven with a complete QCA layout of the adder, as shown in Fig. 8. A comparison between this design and the bit-serial adder [26] has not been carried out since implementation data is not available for the bit-serial design.

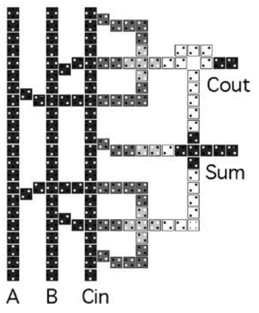


Fig. 8. Layout of the 1-bit QCA adder.

TABLE IV
SIMULATION RESULTS OF QCA ADDERS

QCA Adders	Area	Speed	Clocking
Our Reduction Method	$0.053um^2$	*same*	*same*
Previous Design [25]	$0.149um^2$	*same*	*same*

V. CONCLUSION

In this paper, we have introduced a new strategy for simplifying Boolean functions based on majority gate logic. The study has had a direct application to the design of logic functions in QCA where the logic primitive is a three-input majority gate. Thirteen standard functions have been developed to represent all three-variable Boolean functions and the simplified majority expressions corresponding to these standard functions have been proposed. Furthermore, a procedure has been introduced to map a given three-variable Boolean function to efficient QCA majority logic design. As a case study, the proposed technique is used to develop a 1-bit QCA adder that is constructed with only three majority gates and two inverters. The QCA adder has been designed and simulated using the standard QCA computer-aided (CAD) tool QCADesigner. It is expected that the reduction method presented in this paper will produce significant hardware savings for many future QCA architectures.

ACKNOWLEDGMENT

Workstations and infrastructure support was provided by the Canadian Microelectronics Corporation (CMC). The authors would also like to thank the anonymous reviewers of this paper for their incisive comments.

REFERENCES

[1] M. Wilson *et al.*, *Nanotechnology: Basic Science and Emerging Technologies*. London, U.K.: Chapman & Hall, 2002.
[2] C. S. Lent *et al.*, "Quantum cellular automata," *Nanotechnology*, vol. 4, pp. 49–57, 1993.
[3] C. S. Lent and P. D. Tougaw, "A device architecture for computing with quantum dots," *Proc. IEEE*, vol. 85, pp. 541–557, Apr. 1997.
[4] W. Porod, "Quantum-dot devices and quantum-dot cellular automata," *Int. J. Bifurcation and Chaos*, vol. 7, no. 10, pp. 2199–2218, 1997.

[5] G. Toth and C. S. Lent, “Quasiadiabatic switching for metal-island quantum-dot cellular automata,” *J. Appl. Phys.*, vol. 85, no. 5, pp. 2977–2984, 1999.
[6] I. Amlani *et al.*, “Experimental demonstration of a leadless quantum-dot cellular automata cell,” *Appl. Phys. Lett.*, vol. 77, no. 5, pp. 738–740, 2000.
[7] A. Vetteth *et al.*, “RAM design using quantum-dot cellular automata,” in *Nanotechnology Conf.*, vol. 2, 2003, pp. 160–163.
[8] A. Vetteth *et al.*, “Quantum dot cellular automata carry-look-ahead adder and barrel shifter,” presented at the IEEE Emerging Telecommunications Technologies Conf., 2002.
[9] S. Frost, A. F. Rodrigues, A. W. Janiszewski, R. T. Raush, and P. M. Kogge, “Memory in motion: A study of storage structures in QCA,” presented at the 1st Non-Silicon Computing Workshop, 2002.
[10] D. Berzon and T. J. Fountain, “A memory design in QCA using the SQUARES formalism,” Univ. Collage, London, U.K., Tech. Rep., 1998.
[11] Logic minimization using Karnaugh maps. Dartmouth College, Hanover, NH. [Online]. Available: http://thayer.dartmouth.edu/bpogue/handouts/0420Logic20minimization.pdf
[12] Tabular method of minimization. Univ. Surrey, Guildford, U.K. [Online]. Available: http://www.ee.surrey.ac.uk/Projects/Labview/minimization/tabular.html
[13] ESPRESSO Two level minimization. Univ. California at Berkeley, Berkeley, CA. [Online]. Available: www-cad.eecs.berkeley.edu/brayton/courses/219b/ppslides/2-espresso.ppt
[14] Two level logic minimization. Indian Inst. Technol., Delhi, India. [Online]. Available: http://www.cse.iitd.ernet.in/mbala/cs719/lectures/lect19
[15] A. Orlov *et al.*, “Experimental demonstration of a binary wire for quantum-dot cellular automata,” *Appl. Phys. Lett.*, vol. 74, no. 19, pp. 2875–2877, 1999.
[16] A. Orlov *et al.*, “Experimental demonstration of clocked single-electron switching in quantum-dot cellular automata,” *Appl. Phys. Lett.*, vol. 77, no. 2, pp. 295–297, 2000.
[17] G. Toth, “Correlation and coherence in quantum-dot cellular automata,” Ph.D. dissertation, Dept. Elect. Eng., Univ. Notre Dame, Notre Dame, IN, 2000.
[18] I. Amlani *et al.*, “Digital logic gate using quantum-dot cellular automata,” *Science*, vol. 284, pp. 289–291, 1999.
[19] Threshold logic. Delft Univ. Technol., Delft, The Netherlands. [Online]. Available: http://einstein.et.tudelft.nl/sorin/open/98MSprops8.html
[20] Capacitive threshold-logic circuits. Worcester Polytech. Inst., Worcester, MA. [Online]. Available: http://turquoise.wpi.edu/CTL/
[21] W. Wolf, *Modern VLSI Design: System-on-Chip Design*, ser. Modern Semiconduct. Des. Upper Saddle River, NJ: Prentice-Hall, 2002.
[22] R. K. Brayton, C. McMullen, G. D. Hachtel, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Norwell, MA: Kluwer, 1984.
[23] J. R. Koza, *Genetic Programming: On the Programming of Computers by Natural Selection*. Cambridge, MA: MIT Press, 1992.
[24] Input Boolean program spaces. Univ. College London, London, U.K. [Online]. Available: http://www.cs.ucl.ac.uk/staff/W.Langdon/csrp-98-16/node4.html
[25] P. D. Tougaw and C. S. Lent, “Logical devices implemented using quantum cellular automata,” *J. Appl. Phys.*, vol. 75, no. 3, pp. 1818–1825, 1994.
[26] Bit-serial adder based on quantum dots. NASA, Washington, DC. [Online]. Available: http://www.nasatech.com/Briefs/Jan03/NPO20869.html
[27] QCADesigner user manual and guide. ATIPS Lab., Univ. Calgary, Calgary, AB, Canada. [Online]. Available: http://www.qcadesigner.ca/manual/index.html
[28] K. Walus. (2002) *ATIPS Laboratory QCADesigner homepage* [Online]. Available: http://www.qcadesigner.ca/

Rumi Zhang received the Bachelor degree in engineering from Tsinghua University, Beijing, China, in 1995, and is currently working toward the Master’s degree in electrical and computer engineering at the University of Western Ontario, London, ON, Canada.

Konrad Walus (S’02) received the Electrical Engineering degree from the University of Windsor, Windsor, ON, Canada, in 2001, the Master’s degree in electrical engineering from the University of Calgary, Calgary, AB, Canada, in 2002, and is currently working toward the Ph.D. degree at the University of Calgary.

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

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

Wei Wang (S’99–M’02) received the B.Sc. degree from the Beijing University of Aeronautics, Beijing, China, in 1992, and the Ph.D. degree from Concordia University, Montreal, QC, Canada, in 2002.

From 2000 to 2002, he was a Design Engineer with EMS Technologies, Montreal, QC, Canada. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON, Canada. His main research interests include very large scale integration (VLSI), nanotechnology, signal processing, digital design, application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs), computer arithmetic, and cryptography.

Graham A. Jullien (M’71–SM’83–F’03) received the B.Tech. degree in electrical engineering from the University of Loughborough, Loughborough, U.K., in 1965, the M.Sc. degree from the University of Birmingham, Birmingham, U.K., in 1967, and the Ph.D. degree from the University of Aston, Aston, U.K., in 1969.

From 1961 to 1966, he was a Student Engineer and Data Processing Engineer with English Electric Computers, Kidsgrove, U.K. From 1975 to 1976, he was a Visiting Senior Research Engineer with the Central Research Laboratories, EMI Ltd., Hayes, U.K. From 1969 to 2000, he was with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada, where he was a University Professor and Director of the Very Large Scale Integration (VLSI) Research Group. Since January, 2001, he has been with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada, where he holds the iCORE Research Chair in Advanced Technology Information Processing Systems. He is a member of the Board of Directors of the Canadian Microelectronics Corporation (CMC) and is a member of the Steering Committee and Board of Directors of the Micronet Network of Centres of Excellence. He has authored or coauthored widely in the fields of digital signal processing, computer arithmetic, neural networks, and VLSI systems, and teaches courses in related areas. He currently serves on the Editorial Board of the *Journal of VLSI Signal Processing*.

Dr. Jullien has served on the Technical Committees of numerous international conferences. He is a past associate editor of the IEEE TRANSACTIONS ON COMPUTERS. He hosted and was program co-chair of the 11th IEEE Symposium on Computer Arithmetic, was program chair for the 8th Great Lakes Symposium on VLSI, and Technical Program chair for the 1999 Asilomar Conference on Signals, Systems, and Computers. He was general chair for the 2003 Asilomar Conference and was general co-chair of the International Workshop on System-on-Chip for Real-Time Systems, Calgary, AB, Canada, 2003.