INT. J. ELECTRONICS, VOL. 91, NO. 7, JULY 2004, 431–444

Taylor & Francis
Taylor & Francis Group

Representation of Boolean quantum circuits as Reed–Muller expansions

AHMED YOUNES* and JULIAN F. MILLER†

In this paper we show that there is a direct correspondence between Boolean quantum operations and certain forms of classical (non-quantum) logic known as Reed–Muller expansions. This allows us to readily convert Boolean circuits into their quantum equivalents. A direct result of this is that the problem of synthesis and optimization of Boolean quantum circuits can be tackled within the field of Reed–Muller logic.

1. Introduction

Implementing Boolean functions on quantum computers is an essential aim for those exploring the benefits that may be gained from systems operating by quantum rules. It is important to find the corresponding quantum circuits for carrying out the operations implemented on conventional computers (Yao 1993). On classical computers, a circuit can be built for any Boolean function using *AND*, *OR* and *NOT* gates. This set of gates cannot, in general, be used to build quantum circuits because the operations are not reversible (Toffoli 1980). A corresponding set of reversible gates must be used to build a quantum circuit for any Boolean operation. In classical computer science, many clever methods have been used to obtain more efficient digital circuits (Devadas *et al.* 1994) for a given Boolean function.

Recently, there have been efforts to find an automatic way to create efficient quantum circuits implementing Boolean functions. Lee *et al.* (1999) used a modified version of *Karnaugh maps* (Devadas *et al.* 1994) which depended on a clever choice of minterms to be used in the minimization process; however, this method may have poor scalability. Iwama *et al.* (2002) proposed a very useful set of transformations for Boolean quantum logic and also a method of building quantum circuits for Boolean functions using extra auxiliary qubits; however, this would increase the number of qubits to be used in the final circuits. Younes and Miller (2003) presented a method by which we can convert a truth table of any given Boolean function to its Boolean quantum circuit by applying a set of transformations, after which we get the final circuit. In this paper we will show that there is a close connection between Boolean quantum operations and certain classical Boolean operations known as Reed–Muller logic expansions (Almaini 1989). This means that the study of synthesis and optimization of Boolean quantum logic can be carried out in the classical Reed–Muller logic domain.

Received 8 November 2003. Accepted 5 June 2004.

* Corresponding author. Department of Mathematics and Computer Science, Faculty of Science (El-Shatby), Alexandria University, Egypt. e-mail: ayounes2@yahoo.com

† Department of Electronics, University of York, Heslington, York YO10 5DD, UK.

International Journal of Electronics ISSN 0020–7217 print/ISSN 1362–3060 online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207210412331272643

The plan of the paper is as follows: In § 2, we review the principles of classical Reed–Muller logic. In § 3, we review the principles of quantum computation. In § 4, we discuss the principles of Boolean quantum logic. In § 5, we show how we may implement Boolean quantum logic circuits directly from the corresponding classical Reed–Muller expansions. The paper ends with some conclusions and suggestions for further investigations.

2. Reed-Muller (RM) expansions

In digital logic design, two paradigms have been studied. The first uses the operations of *AND*, *OR* and *NOT* and is called *canonical boolean logic*. The second uses the operations *AND*, *XOR* and *NOT* and called *Reed–Muller logic* (RM). RM is equivalent to modulo-2 algebra. In this section we review the properties of RM logic.

2.1. *Modulo-2 algebra*

For any Boolean variable x, we can write the following *XOR* expressions:

$$x \oplus 1 = \overline{x}, \qquad x \oplus 0 = x$$

$$\overline{x} \oplus 1 = x, \qquad \overline{x} \oplus 0 = \overline{x}$$

Let $\overset{\bullet}{x}$ be a variable representing a Boolean variable in its true (x) or complemented form ($\overline{x}$); we can then write the following expressions:

$$\overset{\bullet}{x} \oplus 1 = \overline{\overset{\bullet}{x}}, \qquad \overset{\bullet}{x} \oplus 0 = \overset{\bullet}{x}$$

$$\overset{\bullet}{x} \oplus \overset{\bullet}{x} = 0, \qquad \overset{\bullet}{x} \oplus \overline{\overset{\bullet}{x}} = 1$$

$$1 \oplus 1 = 0, \qquad \overset{\bullet}{x_0}(1 \oplus \overset{\bullet}{x_1}) = \overset{\bullet}{x_0} \oplus \overset{\bullet}{x_0}\,\overset{\bullet}{x_1}$$

$$f \oplus f\overset{\bullet}{x} = f\overline{\overset{\bullet}{x}}, \text{ where } f \text{ is any Boolean function.}$$

For any *XOR* expression, the following properties hold:

(1) $x_0 \oplus (x_1 \oplus x_2) = (x_0 \oplus x_1) \oplus x_2 = x_0 \oplus x_1 \oplus x_2$ (associative)
(2) $x_0(x_1 \oplus x_2) = x_0x_1 \oplus x_0x_2$ (distributive)
(3) $x_0 \oplus x_1 = x_1 \oplus x_0$ (commutative)

2.2. *Representation of Reed–Muller expansions*

Any Boolean function f with n variables, $f : \{0, 1\}^n \rightarrow \{0, 1\}$, can be represented as a sum of products (Almaini 1989)

$$f(x_0, \ldots, x_{n-1}) = + \sum_{i=0}^{2^n-1} a_i m_i \tag{1}$$

where m_i are the minterms and $a_i = 0$ or 1 indicates the presence or absence of minterms respectively and the plus in front of the sigma means that the arguments are subject to Boolean operation inclusive-*OR*. This expansion can also be expressed

in RM as follows (Akers 1959)

$$f(\overset{\bullet}{x}_0, \ldots, \overset{\bullet}{x}_{n-1}) = \oplus \sum_{i=0}^{2^n-1} b_i \varphi_i \tag{2}$$

where

$$\varphi_i = \prod_{k=0}^{n-1} \overset{\bullet}{x}_k^{i_k} \tag{3}$$

where $\overset{\bullet}{x}_k = x_k$ or $\overline{x}_k$ and $x_k, b_i \in \{0, 1\}$ and i_k represent the binary digits of k.

φ_i are known as product terms and b_i determine whether a product term is presented or not. $\oplus$ indicates the *XOR* operation and multiplication is assumed to be the *AND* operation.

An RM function $f(\overset{\bullet}{x}_0, \ldots, \overset{\bullet}{x}_{n-1})$ is said to have *fixed polarity* if, throughout the expansion, each variable $\overset{\bullet}{x}_k$ is either x_k or $\overline{x}_k$ exclusively. If for some variables x_k and $\overline{x}_k$ both occur, then the function is said to have *mixed polarity*.

There is a relation between the a_i and b_i coefficients shown in equations (1) and (2), which can be found in detail in Almaini (1989).

2.3. π *notations*

Consider the fixed polarity RM functions with $\overset{\bullet}{x}_k$ in its x_k form (positive polarity RM). The RM expansion can be expressed as a ring sum of products. For n variables expansion, there are 2^n possible combinations of variables known as the π terms. 1 and 0 will be used respectively to indicate the presence or absence of a variable in the product term. For example, a four-variable term $x_3x_2x_1x_0$ contains the four variables and is represented by $1111 = 15$, $x_3x_2x_1x_0 = \pi_{15}$ and $x_3x_1x_0$ (x_2 is missing) $= \pi_{11}$.

Using this notation (Almaini 1989), the positive polarity RM expansion shown in equation (2) can be written as follows

$$f(x_0, \ldots, x_{n-1}) = \oplus \sum_{i=0}^{2^n-1} b_i \pi_i \tag{4}$$

Conversion between φ_i and π_i used in equations (2) and (4) can be done in both directions. For example, consider the three variables x_0, x_1 and x_2:

$$\varphi_7 = x_0x_1x_2 = \pi_7$$

$$\varphi_6 = x_0x_1\overline{x}_2 = x_0x_1(x_2 \oplus 1)$$

$$= x_0x_1x_2 \oplus x_0x_1$$

$$= \pi_7 \oplus \pi_6$$

$$\varphi_5 = x_0\overline{x}_1x_2 = x_0(x_1 \oplus 1)x_2$$

$$= x_0x_1x_2 \oplus x_0x_2$$

$$= \pi_7 \oplus \pi_5$$

Similarly, we can construct the rest of the conversion as follows:

$$\varphi_4 = \pi_7 \oplus \pi_6 \oplus \pi_5 \oplus \pi_4$$

$$\varphi_3 = \pi_7 \oplus \pi_3$$

$$\varphi_2 = \pi_7 \oplus \pi_6 \oplus \pi_3 \oplus \pi_2$$

$$\varphi_1 = \pi_7 \oplus \pi_5 \oplus \pi_3 \oplus \pi_1$$

$$\varphi_0 = \pi_7 \oplus \pi_6 \oplus \pi_5 \oplus \pi_4 \oplus \pi_3 \oplus \pi_2 \oplus \pi_1 \oplus \pi_0$$

For the above conversion, the inverse is also true (Almaini 1989):

$$\pi_7 = \varphi_7$$
$$\pi_6 = \varphi_7 \oplus \varphi_6$$
$$\pi_5 = \varphi_7 \oplus \varphi_5$$

and so on.

3. Quantum computers

3.1. *Quantum bits*

The quantum bit (*qubit* (Schumacher 1995)) is the quantum analogue of the classical bit. The basic difference between the qubit and the classical bit is that the qubit can exist in a linear superposition of the two states $|0\rangle$ and $|1\rangle$ at the same time (*quantum parallelism*)

$$a|0\rangle + b|1\rangle \tag{5}$$

where a and b are complex numbers called the amplitudes of the system and satisfy the condition $|a|^2+|b|^2=1$. The states $|0\rangle$ and $|1\rangle$ can be taken as the classical bit values 0 and 1 respectively. $|\ \rangle$ is called the *Dirac notation* (Dirac 1947) and is considered the standard notation for describing quantum states. In quantum circuits shown in this paper, a qubit will be represented as a horizontal line and the time flow of the circuit will be from left to right. If we consider a quantum register with n qubits that are all in superposition, then any operation applied on this register will be applied on the 2^n states representing the superposition *simultaneously*.

3.2. *Quantum gates*

In general, the quantum computation process can be understood as applying a series of quantum gates followed by applying a measurement to obtain the result (Nielsen and Chuang 2000). Quantum gates used during the computation must follow the fundamental laws of quantum physics (Dirac 1947). To satisfy this condition, using any matrix U as a quantum gate, it must be unitary; i.e. the inverse of that matrix must be equal to its complex conjugate transpose: $U^{-1}=U^{\dagger}$ and $UU^{\dagger}=I$, where U^{-1} denotes the inverse of U, $U^{\dagger}$ denotes the complex conjugate transpose of U and I is the identity matrix. A quantum register of size n can be represented as a vector in the 2^n-dimensional complex vector space (Nielsen and Chuang 2000). So, any gate applied on that register can be understood by its action on the basis vectors and can be represented as a unitary matrix of size $2^n \times 2^n$.

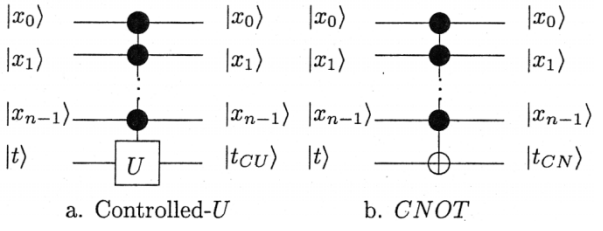

Figure 1. Controlled gates. The black circle • indicates the control qubits, and the symbol ⊕ in (b) indicates the target qubit.

For example, the *NOT* gate is a single input/output gate that inverts the state $|0\rangle$ to $|1\rangle$ and vice versa. Its 2×2 unitary matrix is:

$$NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Another important example is the Hadamard gate (*H* gate), which produces a completely random output with equal probabilities of being $|0\rangle$ or $|1\rangle$ at any measurement. Its 2×2 unitary matrix is:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The Hadamard gate has a special importance in setting up a superposition of a quantum register. Consider a three qubits quantum register $|000\rangle$: applying a Hadamard gate on each of them in parallel will set up a superposition of the 2^3 possible states. Applying any operation on that register afterward will be applied on the 2^3 states simultaneously.

Controlled operations play an important role in building up quantum circuits for any given operation (Barenco *et al.* 1995). The controlled-*U* gate is a general controlled gate with one or more control qubit(s) as shown in figure 1(a). It works as follows: *U* is applied on the target qubit $|t\rangle$ if and only if all $|x_k\rangle$ are set to $|1\rangle$; i.e., qubits will be transformed as follows

$$\left.\begin{array}{c} |x_k\rangle \to |x_k\rangle, \quad k : 0 \to n-1 \\ |t\rangle \to |t_{CU}\rangle = U^{x_0 x_1 \ldots x_{n-1}} |t\rangle \end{array}\right\} \tag{6}$$

where $x_0 x_1 \ldots x_{n-1}$ in the exponent of *U* denotes the *AND*-ing operation of the qubit-values $x_0, x_1, \ldots, x_{n-1}$.

If *U* in the general case is replaced with the *NOT* gate mentioned above, the resulting gate is called a *CNOT* gate (shown in figure 1(b)). It inverts the target qubit if and only if all the control qubits are set to $|1\rangle$ as follows

$$\left.\begin{array}{c} |x_k\rangle \to |x_k\rangle; \quad k : 0 \to n-1 \\ |t\rangle \to |t_{CN}\rangle = |t \oplus x_0 x_2 \ldots x_{n-1}\rangle \end{array}\right\} \tag{7}$$

where ⊕ is the classical *XOR* operation.

4. Boolean quantum operations (*CNOT* gates)

In building quantum circuits for Boolean functions, we will initialize one auxiliary qubit to zero, in order to hold the result of the Boolean function at the end of the computation. For our purposes, we will represent the *CNOT* gates as

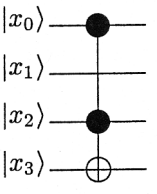

Figure 2. $CNOT(\{x_0, x_2\}|x_3)$ gate.

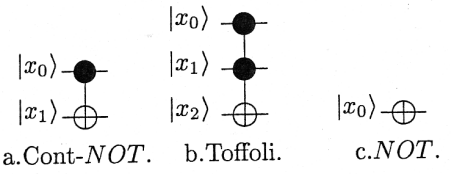

Figure 3. Special cases of the general $CNOT$ gate.

follows (Iwama *et al.* 2002): $CNOT(C|t)$ is a gate where the target qubit $|t\rangle$ is controlled by a set of qubits C such that $t \notin C$; the state of the qubit $|t\rangle$ will be flipped from $|0\rangle$ to $|1\rangle$ or from $|1\rangle$ to $|0\rangle$ if and only if all the qubits in C are set to true (state $|1\rangle$); i.e., the new state of the target qubit $|t\rangle$ will be the result of XOR-ing the old state of $|t\rangle$ with the AND-ing of the states of the control qubits. For example, consider the $CNOT$ gate shown in figure 2. It can be represented as $CNOT(\{x_0, x_2\}|x_3)$, where $\bullet$ on a qubit means that the condition on that qubit will evaluate to true if and only if the state of that qubit is $|1\rangle$, whereas $\oplus$ denotes the target qubit which will be flipped if and only if all the control qubits are set to true, which means that the state of the qubit $|x_3\rangle$ will be flipped if and only if $|x_0\rangle = |x_2\rangle = |1\rangle$ whatever the value of $|x_1\rangle$; i.e., $|x_3\rangle$ will be changed according to the operation $x_3 \rightarrow x_3 \oplus x_0 x_2$.

Some special cases of the general $CNOT$ gate have their own names. A $CNOT$ gate with one control qubit is called a *Cont-NOT* gate (figure 3(a)); a $CNOT$ gate with two control qubits is called a *Toffoli* gate (figure 3(b)); and a $CNOT$ gate with no control qubits is called a NOT gate (figure 3(c)), where C will be an empty set ($C = \phi$). We will refer to this case as $CNOT(x_0)$, where $|x_0\rangle$ is the qubit which will be flipped unconditionally.

4.1. *Boolean quantum circuits (BQCs)*

A general Boolean quantum circuit U of size m (the size of the circuit refers to the total number of $CNOT$ gates in that circuit) over an n-qubit quantum system with qubits $|x_0\rangle, |x_1\rangle, \ldots, |x_{n-1}\rangle$ can be represented as a sequence of $CNOT$ gates (Iwama *et al.* 2002) as follows

$$U_g = CNOT(C_1|t_1) \ldots CNOT\big(C_j|t_j\big) \ldots CNOT(C_m|t_m) \tag{8}$$

where $t_j \in \{x_0, \ldots, x_{n-1}\}$; $C_j \subset \{x_0, \ldots, x_{n-1}\}$; $t_j \notin C_j$ and $j : 1 \rightarrow m$. The BQC we will use in this paper can be represented as follows

$$U = CNOT(C_1|t) \ldots CNOT(C_j|t) \ldots CNOT(C_m|t) \tag{9}$$

where $t \equiv x_{n-1}$; $C_j \subseteq \{x_0, \ldots, x_{n-2}\}$.

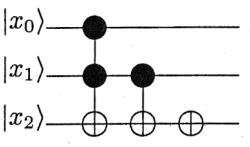

Figure 4. Boolean quantum circuit.

For example, consider the quantum circuit shown in figure 4, which can be represented as follows

$$U = CNOT(\{x_0, x_1\}|x_2) \cdot CNOT(\{x_1\}|x_2) \cdot CNOT(x_2) \tag{10}$$

Now, to trace the operations that have been applied on the target qubit $|x_2\rangle$, we will trace the operation of each of the *CNOT* gates that has been applied:

- $CNOT(\{x_0, x_1\}|x_2) \Rightarrow x_2 \rightarrow x_2 \oplus x_0x_1$
- $CNOT(\{x_1\}|x_2) \Rightarrow x_2 \rightarrow x_2 \oplus x_1$
- $CNOT(x_2) \Rightarrow x_2 \rightarrow \overline{x}_2 = x_2 \oplus 1$

Combining the three operations, we see that the complete operation applied on $|x_2\rangle$ is represented as follows

$$x_2 \rightarrow x_2 \oplus x_0x_1 \oplus x_1 \oplus 1 \tag{11}$$

If $|x_2\rangle$ is initialized to $|0\rangle$, applying the circuit will make $|x_2\rangle$ carry the result of the operation $(x_0x_1 \oplus x_1 \oplus 1)$, which is equivalent to the operation $(x_0 + \overline{x}_1)$.

5. Representation of BQC as RM

5.1. *BQC for positive polarity RM*

Considering the previous sections, we might notice that there is a close connection between RM and quantum circuits representing arbitrary Boolean function. In this section we will show the steps that allow us to implement any arbitrary Boolean function f using positive polarity RM expansions as quantum circuits.

Example:

Consider the function $f(x_0, x_1, x_2) = \overline{x}_0 + x_1x_2$. To find the quantum circuit implementation for this function, we may follow the following procedure:

(1) The above function can be represented as a sum of products as follows

$$f(x_0, x_1, x_2) = \overline{x}_0\overline{x}_1\overline{x}_2 + \overline{x}_0\overline{x}_1x_2 + \overline{x}_0x_1\overline{x}_2 + \overline{x}_0x_1x_2 + x_0x_1x_2 \tag{12}$$

(2) Converting to φ_i notation according to equation (2)

$$f(x_0, x_1, x_2) = \varphi_0 \oplus \varphi_1 \oplus \varphi_2 \oplus \varphi_3 \oplus \varphi_7 \tag{13}$$

(3) Substituting π product terms as shown in § 2.3, we get

$$\begin{aligned} f = {} & \pi_7 \oplus \pi_6 \oplus \pi_5 \oplus \pi_4 \oplus \pi_3 \oplus \pi_2 \oplus \pi_1 \oplus \pi_0 \oplus \pi_7 \oplus \pi_5 \\ & \oplus \pi_3 \oplus \pi_1 \oplus \pi_7 \oplus \pi_6 \oplus \pi_3 \oplus \pi_2 \oplus \pi_7 \oplus \pi_3 \oplus \pi_7 \end{aligned} \tag{14}$$

(4) Using modulo-2 operations to simplify this expansion, we get

$$f = \pi_7 \oplus \pi_4 \oplus \pi_0 = x_0x_1x_2 \oplus x_0 \oplus 1 \tag{15}$$

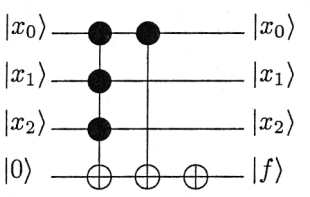

Figure 5. Quantum circuit implementation for $f(x_0, x_1, x_2) = \overline{x}_0 + x_1x_2$.

Using the last expansion in equation (15), we can create the quantum circuit, which implements this function as follows:

(1) Initialize the target qubit $|t\rangle$ to the state $|0\rangle$, which will hold the result of the Boolean function.
(2) Add *CNOT* gate for each product term in this expansion, taking the Boolean variables in this product term as control qubits and the result qubit as the target qubit $|t\rangle$.
(3) For each product term, which contains 1, we will add *CNOT*(t), so the final circuit will be as shown in figure 5.

5.2. *BQC for different RM polarities*

Consider the RM expansion shown in equation (2), where $\overset{\bullet}{x}_k$ can be x_k or $\overline{x}_k$ exclusively. For an n-variables expansion where each variable may be in its true or complemented form, but not both, then there will be 2^n possible expansions. These are known as *fixed polarity generalized Reed–Muller* (GRM) *expansions*.

We can identify different GRM expansions by a *polarity number*, which is a number that represents the binary number calculated in the following way: if a variable appears in its true form, it will be represented by 1, and by 0 for a variable appearing in its complemented form. For example, consider the Boolean function $f(\overset{\bullet}{x}_0, \overset{\bullet}{x}_1, \overset{\bullet}{x}_2)$: $f(x_0, x_1, x_2)$ has polarity 0 (000), $f(x_0, \overline{x}_1, x_2)$ has polarity 2 (010), $f(\overline{x}_0, x_1, \overline{x}_2)$ has polarity 5 (101), and $f(\overline{x}_0, \overline{x}_1, \overline{x}_2)$ has polarity 7 (111), and so on.

The RM expansion with a certain polarity can be converted to another polarity by replacing any variable x_k by $(\overline{x}_k \oplus 1)$ or any variable $\overline{x}_k$ by $(x_k \oplus 1)$. For example, consider the Boolean function $f(x_0, x_1, x_2) = \overline{x}_0 + x_1x_2$ it can be represented with different polarity RM expansions as follows

$$f = x_0x_1x_2 \oplus x_0 \oplus 1\text{: 0 polarity} \tag{16}$$

$$f = x_0x_1\overline{x}_2 \oplus x_0x_1 \oplus x_0 \oplus 1\text{: 1 polarity} \tag{17}$$

$$f = \overline{x}_0x_1\overline{x}_2 \oplus x_1\overline{x}_2 \oplus \overline{x}_0x_1 \oplus x_1 \oplus \overline{x}_0\text{: 5 polarity} \tag{18}$$

$$f = \overline{x}_0\overline{x}_1\overline{x}_2 \oplus \overline{x}_0\overline{x}_2 \oplus \overline{x}_1\overline{x}_2 \oplus \overline{x}_0\overline{x}_1 \oplus \overline{x}_1 \oplus \overline{x}_2 \oplus 1\text{: 7 polarity} \tag{19}$$

Different polarity RM expansions will give different quantum circuits for the same Boolean function. For example, consider the different polarity representations for the function $f(x_0, x_1, x_2) = \overline{x}_0 + x_1x_2$ shown above. Each representation has

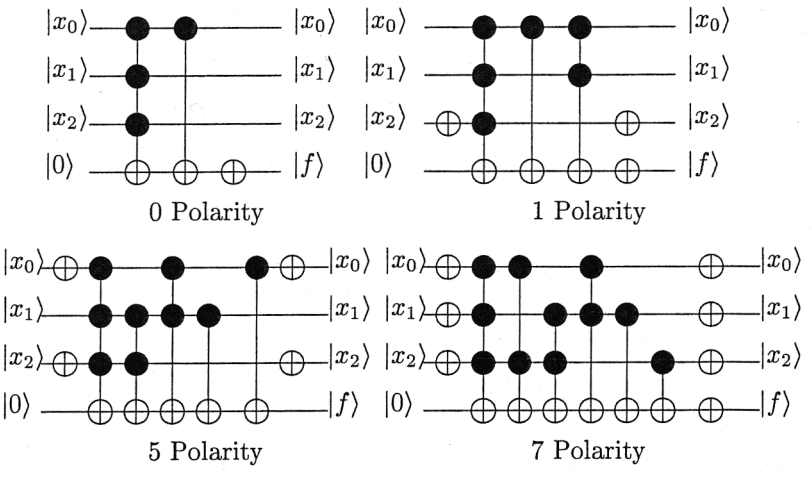

Figure 6. Quantum circuits for the Boolean function $f(x_0, x_1, x_2) = \overline{x}_0 + x_1x_2$ with different polarities.

a different quantum circuit (as shown in figure 6), using the following procedure:

(1) Initialize the target qubit $|t\rangle$ to the state $|0\rangle$, which will hold the result of the Boolean function.
(2) Add a *CNOT* gate for each product term in the RM expansion, taking the Boolean variables in this term as control qubits and the resulting qubit as the target qubit $|t\rangle$.
(3) For the product term, which contains 1, add $CNOT(t)$.
(4) For the control qubit $|x_k\rangle$, which appears in complemented form, add $CNOT(x_k)$ at the beginning of the circuit to negate its value during the run of the circuit and add another $CNOT(x_k)$ at the end of the circuit to restore its original value.

It is clear from figure 6 that changing polarity will change the number of *CNOT* gates in the circuit; i.e. its efficiency. This means that there is a need to develop search algorithms for optimizing canonical Reed–Muller expansions for quantum Boolean functions similar to those found for classical digital circuit design (Miller and Thomson 1994, Robertson *et al.* 1996), taking into account that efficient expansions for classical computers may be not so efficient for quantum computers. For example, consider $f(x_0, x_1, x_2)$, defined as follows

$$f = \overline{x}_0\overline{x}_1\overline{x}_2 + \overline{x}_0x_1\overline{x}_2 + x_0\overline{x}_1x_2 + x_0x_1x_2 \tag{20}$$

its 0 polarity expansion is given by $(x_0 \oplus x_2 \oplus 1)$ and its 3 polarity expansion is given by $(\overline{x}_0 \oplus x_2)$. From a classical point of view, 3 polarity expansion is better than 0 polarity expansion because it contains two product terms rather than the three product terms in 0 polarity expansion. On the contrary, implementing both expansions as BQC, we can see that 0 polarity expansion is better than 3 polarity expansion because of the number of *CNOT* gates used, as shown in figure 7.

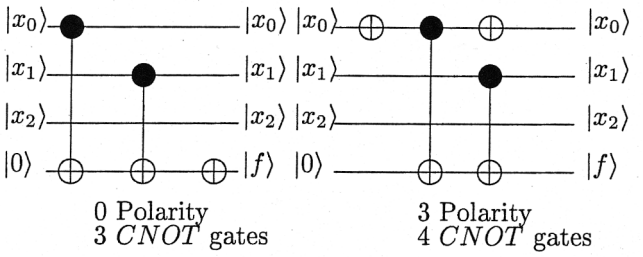

Figure 7. Changing polarity may affect the number of *CNOT* gates used.

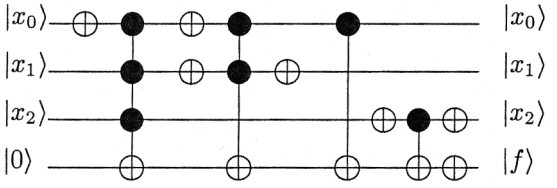

Figure 8. Mixed polarity BQC for $f = \overline{x}_0 x_1 x_2 \oplus x_0 \overline{x}_1 \oplus x_0 \oplus \overline{x}_2 \oplus 1$.

5.3. *BQC for mixed polarity RM*

Mixed polarity RMs are expansions where it is allowed that some variables $\dot{x}_k$ may appear in their true form (x_k) and their complemented form ($\overline{x}_k$), both in the same expansion. To understand how this kind of expansion can be implemented as a quantum circuit, consider the three-variable mixed polarity RM

$$f = \overline{x}_0 x_1 x_2 \oplus x_0 \overline{x}_1 \oplus x_0 \oplus \overline{x}_2 \oplus 1 \tag{21}$$

using the following procedure, we will get the quantum circuit shown in figure 8:

(1) Initialize the target qubit $|t\rangle$ to the state $|0\rangle$, which will hold the result of the Boolean function.
(2) Add a *CNOT* gate for each product term in the RM expansion, taking the Boolean variables in this term as control qubits and the resulting qubit as the target qubit $|t\rangle$.
(3) For the product term, which contains 1, add $CNOT(t)$.
(4) For control qubit $|x_k\rangle$, which appears in complemented form, add $CNOT(x_k)$ directly before and after (negate/restore) the *CNOT* gate where this variable appears in its complemented form.

5.4. *Calculating the number of CNOT gates*

For a *fixed polarity* RM expansion, the number of *CNOT* gates in the final quantum circuit can be calculated as follows

$$S_1 = m + 2K, \qquad 0 \le m \le 2^n; \qquad 0 \le K \le n \tag{22}$$

where S_1 is the total number of *CNOT* gates, m is the number of product terms in the expansion, K is the number of variables in complemented form and n is the number of inputs to the Boolean function; the term $2K$ represents the number of *CNOT* gates that will be added at the beginning and the end of the circuit (complemented

form) to negate the value of the control qubit during the run of the circuit and to restore its original value, respectively.

For a *mixed polarity* RM expansion, the number of *CNOT* gates in the final quantum circuit can be calculated as follows

$$S_2 = m + 2L, \quad 0 \leq m \leq 2^n; \quad 1 \leq L \leq n2^{n-1} \tag{23}$$

where S_2 is the total number of *CNOT* gates, m is the number of product terms in the expansion, L is the total number of occurrences of all variables in complemented form and n is the number of inputs to the Boolean function; the term $2L$ represents the number of *CNOT* gates that may be added before and after the control qubit, which appears in complemented form during the run of the circuit, to negate/restore its value, respectively.

6. Practical construction of BQC

In general, the meaning of optimality for quantum circuits is connected with practical constraints. For instance, there is the interaction between certain control qubits; circuits depend on the physical implementation, so it is sometimes difficult to take certain qubits as control qubits on the same *CNOT* gates (involved in the same operation) because the interaction between these qubits may be difficult to control. Another constraint is the number of control qubits per *CNOT* gate; at present it is not clear whether the cost of implementation of multiple-input *CNOT* gates is higher than that of fewer-input *CNOT* gates, so it may be better to use fewer control qubits per *CNOT* gate. Another constraint is the total number of *CNOT* gates in the circuit, which should be kept to a minimum so that we are able to maintain coherence during the operation of the circuit. In this section, we will review how these constraints are being handled (one at a time) at present and how RM expansions can help in handling this problem.

Consider an abstract four-qubits system, shown in figure 9, where Y and N in the associated table mean that the qubits can and cannot interact, respectively. For simplicity, the auxiliary qubit $|0\rangle$ will be able to interact with all the control qubits $|x_0\rangle$, $|x_1\rangle$ and $|x_2\rangle$. The main problem will be in the interaction between $|x_0\rangle$ and $|x_2\rangle$. This problem is known to be handled using the *SWAP* gate shown in figure 10. To illustrate this, consider the gate shown in figure 11, where $|x_0\rangle$ and $|x_2\rangle$ are involved in the same *CNOT* gate. By temporarily swapping the values of $|x_0\rangle$ and $|x_1\rangle$, this gate can be implemented on the above system with an increase in the total number of *CNOT* gates because of the added *SWAP* gates.

	x_0	x_1	x_2	0
x_0	Y	Y	N	Y
x_1	Y	Y	Y	Y
x_2	N	Y	Y	Y
0	Y	Y	Y	Y

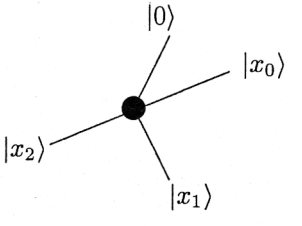

Figure 9. An abstract four-qubits system with their allowed interaction.

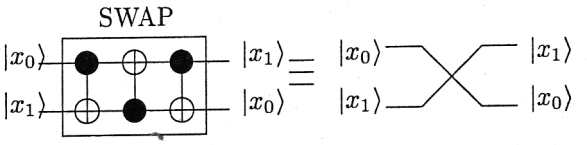

Figure 10. *SWAP* gate.

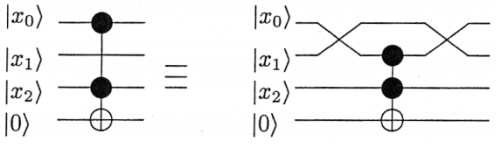

Figure 11. Solving an interaction problem, using *SWAP* gate.

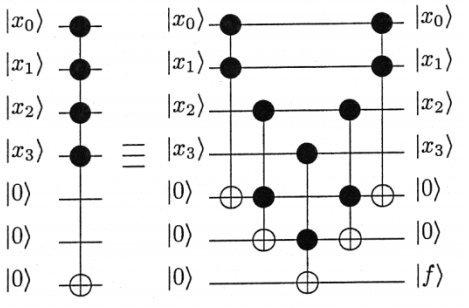

Figure 12. Decreasing the number of control qubits per *CNOT* gate.

Another constraint is the number of control qubits per *CNOT* gate. Barenco *et al.* (1995) studied this problem and showed that the number of control qubits can be reduced to two control qubits per *CNOT* gate (three-qubit gates) by using extra auxiliary qubits and an increase in the number of *CNOT* gates in the final circuit, each of which can then be decomposed to two qubit gates (Lemma 6.1 in Barenco *et al.* 1995), which was proved to be universal for quantum computation (Divincenzo 1995). In figure 12, we show an equivalent decomposition to that shown in Lemma 7.2 in Barenco *et al.* (1995) but more efficient, using the transformation rules shown by Iwama *et al.* (2002). If we only care about the total number of *CNOT* gates, we can directly use the 0 polarity RM expansion.

The above techniques deal with different constraints in a *gate level approach*. Using RM allows us to deal with the problem at a *circuit level approach*. To illustrate this, consider the equivalent quantum circuits shown in figure 13. Consider implementing these circuits on the system shown in figure 9: the 0 polarity form contains the interaction problem between $|x_0\rangle$ and $|x_2\rangle$ twice, whereas the 1 polarity and 3 polarity circuits contain this interaction only once. 0 polarity and 1 polarity contain five *CNOT* gates and 3 polarity contains six *CNOT* gates, with an increase in the simple *NOT* gates in favour of more complicated controlled operations. In terms of the total number of control qubits, 0 polarity contains five control qubits, 1 polarity contains four control qubits and 3 polarity contains three control qubits.

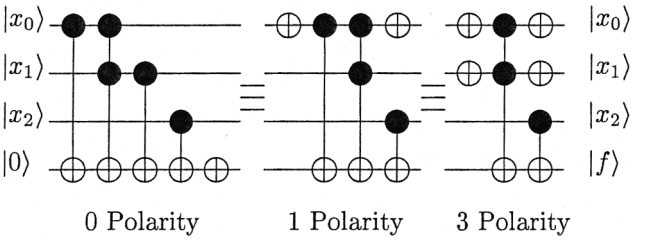

Figure 13. Optimization, using different polarities.

In that sense, using RM can be considered as a platform for synthesis and optimization of BQC to minimize the cost of using any of the above *constraint handling techniques*.

7. Conclusion

In this paper we showed that there is a close connection between quantum Boolean operations and Reed–Muller expansions, which implies that a complete study on synthesis and optimization of Boolean quantum logic can be done within the domain of classical Reed–Muller logic. If we consider a positive polarity RM expansion and its corresponding BQC, then, using our proposed method, we will get the same circuit efficiency we showed in Younes and Miller (2003) *without the use of the truth table of the Boolean function or applying any transformations*.

We showed that the sense of optimality in quantum circuit construction follows practical constraints, which can also be handled using the RM expansions. We showed that an efficient RM expression on classical computers may not be so efficient on quantum computers, and vice versa. In that sense, we showed that the construction of Boolean quantum logic could be tackled within the domain of RM and algorithms for optimizing BQC that are required should be able to be found within that domain.

References

Akers, S. B., 1959, On a theory of Boolean functions. *Journal of the SIAM*, **7**, 487–498.

Almaini, A. E. A., 1989, *Electronic Logic Systems*, second edition (Englewood Cliffs, NJ: Prentice-Hall), Chap. 12.

Barenco, A., 1995, A universal two-bit gate for quantum computation. *Proceedings of the Royal Society of London A*, **449**(1937), 679–683.

Barenco, A., Bennett, C., Cleve, R., Divincenzo, D. P., Margolus, N., Shor, P., Sleator, T., Smolin, J., and Weinfurter, H., 1995, Elementary gates for quantum computation. *Physical Review A*, **52**(5), 3457–3467.

Devadas, S., Ghosh, A., and Keutzer, K., 1994, *Logic Synthesis* (New York, USA: McGraw-Hill).

Dirac, P., 1947, *The Principles of Quantum Mechanics* (Oxford, UK: Clarendon Press).

Divincenzo, D., 1995, Two-bit gates are universal for quantum computation. *Physical Review A*, **51**(2), 1015–1022.

Iwama, K., Kambayashi, Y., and Yamashita, S., 2002, Transformation rules for designing CNOT-based quantum circuits. *Proceedings of the 39th Conference on Design Automation*, ACM Press, pp. 419–424.

Lee, J., Cheong, Y., Kim, J., and Lee, S., 1999, A practical method of constructing quantum combinational logic circuits. Los Alamos physics preprint archive, quant-ph/9911053.

Miller J., and Thomson, P., 1994, Highly efficient exhaustive search algorithm for optimising canonical Reed–Muller expansions of Boolean functions. *International Journal of Electronics*, **76**, 37–56.

Nielsen, M., and Chuang, I., 2000, *Quantum Computation and Quantum Information* (Cambridge, UK: Cambridge University Press).

Robertson, G., Miller, J., and Thomson, P., 1996, Non-exhaustive search methods and their use in the minimisation of Reed–Muller canonical expansions. *International Journal of Electronics*, **76**, 1–12.

Schumacher, B., 1995, Quantum coding. *Physical Review A*, **51**, 2738–2747.

Toffoli, T., 1980, Reversible computing. *Automata, Languages, and Programming* (Springer-Verlag), pp. 632–644.

Yao, A., 1993, Quantum circuit complexity. *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pp. 352–361.

Younes, A., and Miller, J., 2003, Automated method for building CNOT based quantum circuits for Boolean functions. Technical report CSR-03-3, University of Birmingham. Los Alamos physics preprint archive, quant-ph/0304099.