# Nearest Neighbour based Synthesis of Quantum Boolean Circuits

Amlan Chakrabarti and Susmita Sur-Kolay

*Abstract*—**Quantum Boolean circuit synthesis issues are becoming a key area of research in the domain of quantum computing. For gate-level synthesis, minterm based and Reed-Muller canonical decomposition techniques are adopted as common approaches. Physical implementation of quantum circuits have inherent constraints and hence nearest neighbour template of input lines is gaining importance. In this work, we present a brief analysis of the various Fixed Polarity Reed Muller (FPRM) expressions for a given quantum Boolean circuit and also introduce the rules for the nearest neighbour template-based synthesis of these forms. The corresponding circuit costs are evaluated.**

**Index Terms**—**Quantum Boolean Circuit, Reversible Logic, Reed-Muller Expression, Nearest Neighbour Template**

## I. INTRODUCTION

The model of Quantum Computing stands on the understanding of quantum circuits and their application to solve computational problems. A quantum circuit is employed to process quantum bits (qbit). A qbit may be considered as the equivalent to a binary bit in a classical computer [1]. It can be taken as a particular spin state of an electron, or a certain polarization state of a photon. The spin state of an electron may be up (↑) or (↓) down, or the polarization state of a photon may be vertical (↕) or horizontal ( ↔ ). The two quantum mechanical states are represented in standard quantum mechanics [2] by standard ket notation |0> and |1>. The real difference between the classical and quantum states is that while in the former, the states are definite, in quantum computing the states are superposed. For example, a quantum state is represented by superposition of two states like $\psi = a\ |0\rangle + b\ |1\rangle$ where $a$ and $b$ are the complex amplitudes representing the probabilities of state |0> and |1> respectively satisfying the condition $a^2 + b^2 = 1$. Unlike a classical computer in which a bit has exactly one value from the set $\{0, 1\}$, a qbit can represent both states simultaneously. The maximum number of possible states depends on the number of qbits i.e., $n$ qbits can represent $2^n$ states.

A 2-qbit vector can simultaneously represent the states |00>, |01>, |10>, |11> and the probability of their occurrence depends on the value of the complex amplitudes $c_0$, $c_1$, $c_2$ and $c_3$. The superposed quantum state $\psi$ is represented as $\psi = c_0 |00\rangle + c_1 |01\rangle + c_2 |10\rangle + c_3 |11\rangle$. Hence comes the concept of quantum register [6] of $n$ qbits holding $2^n$ simultaneous values. This also implies that if we perform an operation on the contents of a register, all possible values are operated on simultaneously, thus leading to quantum parallelism [5]. However, in practice it is quite complex to achieve quantum parallelism, and is dependent on the property of quantum decoherence [3], [7].

While Quantum Boolean circuit synthesis for Reed-Muller expansion using $C^kNOT$ gates exists, this paper proposes a method for physically viable synthesis using nearest neighbour templates of quantum gates.

The organization of this paper is as follows. Preliminary concepts of reversible logic which is predominant in quantum gates appear in Section II. A brief introduction to quantum gate network is presented in Section III. Implementation of Quantum Boolean functions with Reed-Muller decomposition and circuit construction based on physical realization is given in Section IV and concluding remarks in Section V.

## II. PRELIMINARIES

### A. Reversible Logic

A Boolean function is reversible if each of the values in the input set can be mapped with a unique value in the output set. Landauer [8] proved that the usage of traditional irreversible circuits leads to power dissipation and Bennet [4] showed that a circuit consisting of only reversible gates does not dissipate power. Above all, some of the applications like digital signal processing, computer graphics, cryptography, reconfigurable computing require the preservation of input data.

### B. Reversible Logic Gates

A reversible logic gate implements a reversible Boolean function and necessarily has equal number of input and output wires. Next we discuss about a few reversible gates.

*$C^kNOT$ Gates:* In general, a $C^kNOT$ gate has $k+1$ input and output wires. It has $k$ control inputs and the $(k+1)^{th}$ input is inverted at the output only if all the $k$ control inputs are at logic high. For $k=0$, it is equivalent to a *NOT* gate which maps the input $x \rightarrow x \oplus 1$, i.e., classical *XOR* of input with logic 1, as shown in Fig.1(a). For $k=1$ it is termed as controlled *NOT (CNOT)* gate which maps the two inputs $(x, y) \rightarrow (x, x \oplus y)$ as shown in Fig. 1(b). The $C^2NOT$ gate, also termed *as TOFFOLI* gate, maps the three inputs top-control, bottom-control and target qbits $(x, y, z) \rightarrow (x, y, x \oplus x.y)$ as shown in Fig. 1(c), where the classical *XOR* and

*AND* operations are involved. A $C^kNOT$ gate with $k$ control qbits and a single target qbit, shown in Fig. 1(d), maps $(x_1,\ldots,x_{k-1},x_k, y) \rightarrow (x_1,\ldots,x_k, y \oplus x_1.x_2\ldots x_k)$. The control and the target qbits are indicated by • and ⊕ respectively.

*Swap Gates:* A swap gate is a *2x2* reversible gate. It interchanges the values of two input qbits at the output. Fig. 2 illustrates the internal architecture of a swap gate.

### C. Reversible Quantum Boolean Circuits

The synthesis of Quantum Boolean Circuits (QBCs) can be done if we define a set of transformation rules for reversible QBCs. A QBC is a quantum system of $n$ qbits specified by $|x_1\rangle|x_2\rangle\ldots\ldots\ldots|x_n\rangle$ and a number of reversible quantum gates. In QBC the convention for circuit representation is to have the input qbits at the extreme left, which interact with a sequence of reversible quantum gates as desired and finally the output appears at the extreme right where all the input values are restored at the output. The desired function is obtained with the help of a set of *ancillary* bits which are initialized at the input with $|0\rangle$.

### D. Previous Work

Younnes and Miller have introduced in their work [9], techniques for representation of quantum Boolean circuits using Reed-Muller expansions and have mainly focused on generalized $C^kNOT$ based circuit synthesis. Though $C^kNOT$ gates are acceptable in logic design, the technology based implementation of quantum circuits demands the usage of only one, two and three qbit quantum logic gates like *NOT, CNOT, SWAP* and $C^2NOT$. Hence there is a need for defining efficient synthesis techniques in quantum circuits involving only quantum gates with small fan-in.

## III. QUANTUM GATE NETWORK

A classical logic operation is a Boolean operation on a set of inputs and resulting to a single output. So a logic function $f$ can be generalized as $f: \{0,1\}^n \rightarrow \{0,1\}$, where $n$ is the number of inputs. A logic function is expressed by its minterms for implementation. A *minterm* corresponds to an input combination corresponding to which the value of the logic function is logic high and is represented by the equivalent decimal integer of the $n$-bit binary value.

In quantum computing, the realization of Boolean logic needs the implementation of reversible logic operations and hence the *CNOT* gates provide us a possible solution. We need to build a circuit utilizing reversible *CNOT* gates for the implementation of Boolean function in the quantum domain. The circuit for a $n$ qbit single valued Quantum Boolean function can be represented as a network with $n+1$ input and output qbits, as shown in Fig. 3. The $n$ control qbits are represented as $|a_i\rangle$ for $i= 0,1,2\ldots,n-1$ and the extra target output qbit as $|f\rangle$ to store the result qbit, as shown in Fig.3. The extra input qbit is initialized to $|0\rangle$.

It should be noted that in quantum domain, there is no feedback and fanout. Also for reversibility we have to restore all the input values after each quantum gate operation, hence classical synthesis from minterms are not directly applicable. The choice of proper gate library for synthesizing a quantum gate network affects the total circuit cost in terms of technology based implementation. For this purpose, we evaluate the gate cost for synthesizing the Quantum Boolean function using the Reed Muller canonical decomposition technique utilizing a quantum gate library with *NOT*, *CNOT*, *SWAP* and $C^2NOT$ gates only.

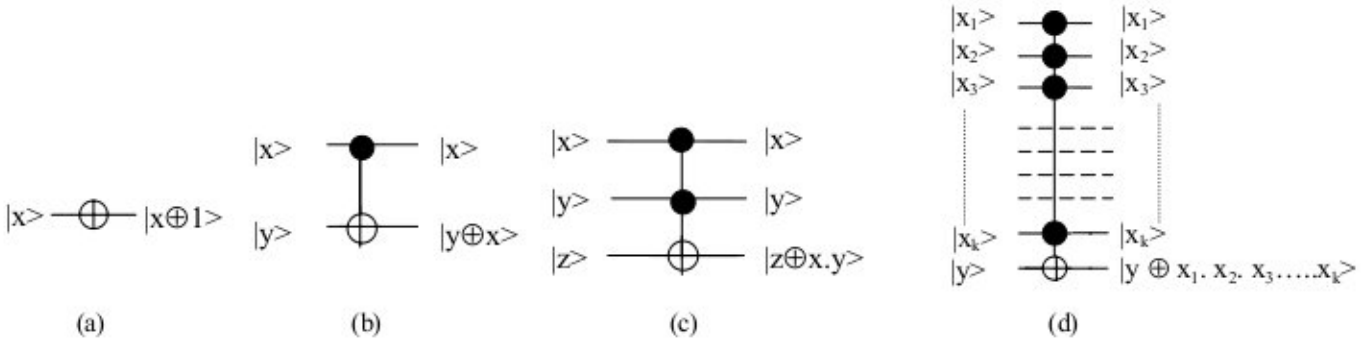


Figure 1: (a) NOT gate, (b) CNOT gate, (c) $C^2$NOT or TOFFOLI gate, (d) $C^k$NOT gate.

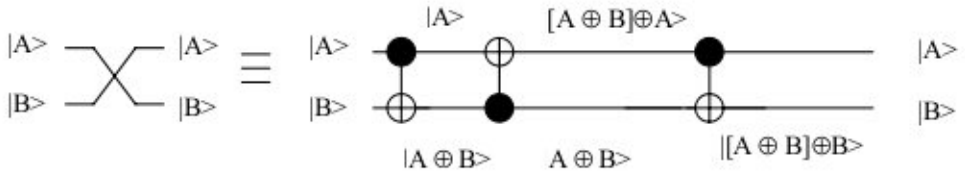


Figure 2: SWAP Gate

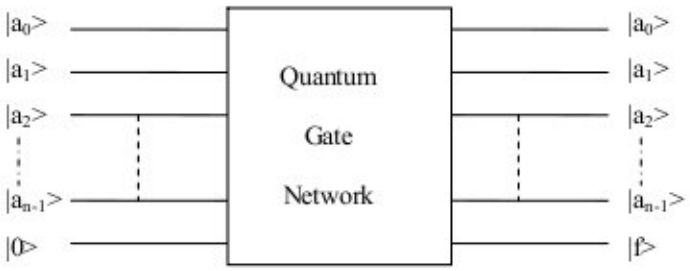


Figure 3: A Generalized Quantum Boolean Circuit

## IV. IMPLEMENTATION

A boolean function $f$ of $n$ variables can be expressed in the Reed Muller form as (Akers 1959):

$$f(\overset{\bullet}{x}_0, \ldots\ldots \overset{\bullet}{x}_{n-1}) = \oplus \sum_{i=0}^{2^n-1} b_i \varphi_i \text{ ; where } \varphi_i = \prod_{k=0}^{n-1} \overset{\bullet}{x}_k$$

and $\overset{\bullet}{x}_k = x_k$ or $\overline{x_k}$, $b_i \in (0, 1)$.

$\varphi_i$ are known as product terms and $b_i$ determines whether a a product term is present or not. The XOR operation is indicated by $\oplus$ and multiplication is assumed to be the AND operation. The canonical Reed Muller expression can be classified as **Positive Polarity Reed Muller (PPRM),** where the variables are un-complemented. For each variable $x_i$ in the given expression if we use the un-complemented literal ( $x_i$ ) or the complemented literal ( $\overline{x}_i$ ) throughout, then it is **Fixed Polarity Reed Muller (FPRM)** expression.

Consider the realization of the function $f_1 (x_0, x_1, x_2) = \Sigma (1,5,6,7)$ as given in Table 1. The PPRM expression for it and the seven $(2^3 - 1)$ FPRM forms are given next.

Table 1: Truth Table for $f_1 (x_0, x_1, x_2)$

| $x_2$ | $x_1$ | $x_0$ | $f_1(x_0,x_1,x_2)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$f_1(x_0,x_1,x_2) = m_1 + m_5 + m_6 + m_7 = x_2.x_1 \oplus x_1.x_0 \oplus x_0 \quad (1)$$

$f_1(x_0,x_1,x_2)$ can also be expressed in different FPRM forms as :

$$f_1(x_0,x_1,x_2) = x_2x_1 \oplus x_1\bar{x}_0 \oplus x_1 \oplus \bar{x}_0 \oplus 1 \quad \textit{(1 polarity)} \quad (2)$$

$$f_1(x_0,x_1,x_2) = x_2\bar{x}_1 \oplus x_2 \oplus \bar{x}_1x_0 \quad \textit{(2 polarity)} \quad (3)$$

$$f_1(x_0,x_1,x_2) = x_2\bar{x}_1 \oplus x_2 \oplus \bar{x}_1\bar{x}_0 \oplus \bar{x}_1 \quad \textit{(3 polarity)} \quad (4)$$

$$f_1(x_0,x_1,x_2) = \bar{x}_2x_1 \oplus x_1 \oplus x_1x_0 \oplus x_0 \quad \textit{(4 polarity)} \quad (5)$$

$$f_1(x_0,x_1,x_2) = \bar{x}_2x_1 \oplus x_1\bar{x}_0 \oplus \bar{x}_0 \oplus 1 \quad \textit{(5 polarity)} \quad (6)$$

$$f_1(x_0,x_1,x_2) = \bar{x}_2\bar{x}_1 \oplus \bar{x}_1x_0 \oplus \bar{x}_2 \oplus \bar{x}_1 \oplus 1 \quad \textit{(6 polarity)} \quad (7)$$

$$f_1(x_0,x_1,x_2) = \bar{x}_2\bar{x}_1 \oplus \bar{x}_1\bar{x}_0 \oplus \bar{x}_2 \oplus 1 \quad \textit{(7 polarity)} \quad (8)$$

The position index of a '1' in the 3-bit binary equivalent of the decimal integer value of a polarity corresponds to the index of the complemented variable. For example, only variable $x_2$ is complemented in *4 polarity* FPRM above.

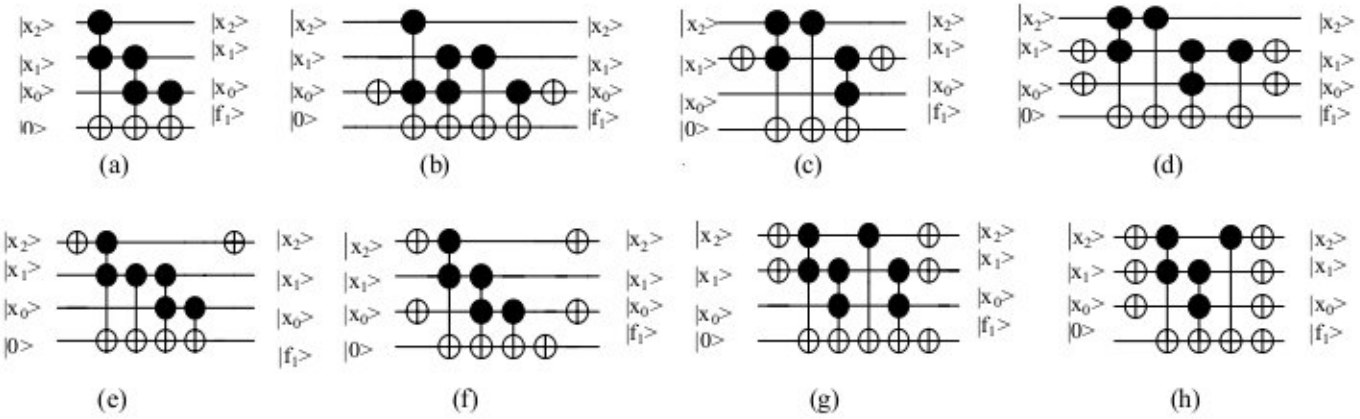


Figure 4: Quantum gate networks for all possible Reed Muller expressions of the function $f_1$ in Table 1:
(a) PPRM, (b) 1 polarity FPRM , (c) 2 polarity FPRM, (d) 3 polarity FPRM,
(e) 4 polarity FPRM, (f) 5 polarity FPRM, (g) 6 polarity FPRM, (h) 7 polarity FPRM.

As seen in Figure 3, *SWAP* gates play a key role in bringing the control and the target qbits of any quantum gate on adjacent lines in a quantum gate network which is called the **nearest neighbor configuration**. The requirement of nearest neighbour relationship between the control and the target qbits is truly justified due to the limitation of the J-coupling force [10] required to perform multi-qbit logic operations and this works effectively only between the adjacent qbits.

We present below a set of circuit templates for non-adjacent qbit controlled *CNOT* and $C^2NOT$ in our nearest neighbor based synthesis approach. We introduce the circuit templates for *CNOT*, and $C^2NOT$ gates utilizing the *SWAP* gates. In Fig. 5, for a $C^2NOT$ we use the notation ($ctrl_1$,$ctrl_2$,*target*) where the integers $ctrl_1$, $ctrl_2$ and *target* are respectively the indices of the input qbits of the circuit for the top-control, bottom-control and the target qbit of this $C^2NOT$ gate. The same convention is also followed for *CNOT* gate which is represented as *CNOT*(*ctrl*,*target*). According to the convention for index values of input qbit lines mentioned earlier, we assign index 1 to the bottommost control qbit input of the circuit, and the successive index values are assigned as we go upwards to the topmost qbit line. Thus, $C^2NOT(4,3,1)$ represents a $C^2NOT$ gate with its top-control on the 4th input qbit line, its bottom-control on the 3rd line and its target is on the lowest (1st) input qbit line. The exact number of SWAP gates required for nearest neighbor configuration is determined by the differences in the index values of the two control qbits with the target qbit, which can be calculated by the following rules:

**Rule 1:** *For a $C^2NOT$ gate, we require $s_t$ pairs of SWAP gates if the difference between the index values of the top-control and the target qbit is $s_t$ with $s_t$ greater than 2, and $s_b$ pair of swap gates if the difference between the index values of the bottom-control and the target qbit is $s_b$ with $s_b$ greater than 1.*

**Example:** In $C^2NOT(4,3,1)$ (Fig. 5(a)) the difference in index value between the top-control and target is $4-1=3$ and that between the bottom-control and the target is $3-1=2$, hence we require two pairs of *SWAP* gates, one each to make the top-control and bottom-control as the nearest neighbor of the target qbit.

In $C^2NOT(4,2,1)$ (Fig. 5(b)) the difference between the top-control and target is $4-1=3$ and that between the bottom-control and the target is $2-1=1$ , hence we require only one pair of *SWAP* gate.

**Rule2:** *For a CNOT gate we require $s_c$ pairs of SWAP gates if the difference between the index values of the control and the target qbit is $s_c$ with $s_c$ greater than 1.*

**Example:** In $CNOT(4,1)$ (Fig. 5(c)) the difference in index value between the control and the target qbit is $4-1 = 3$, hence we require 2 pairs of *SWAP* gates, and similiarly we require a single pair of *SWAP* gate for $CNOT(3,1)$.

The circuit in Fig. 6(a) corresponds to the 1 polarity FPRM of the function $f_1$. Using the nearest neighbour template $C^2NOT(4,3,1)$ in Fig. 3(b), the circuit shown in Fig. 6(b) is obtained. We can observe an increase in the gate count and circuit level due to the usage of the extra *SWAP* gates. Hence we need to focus on the minimization of gate count and number of levels in the QBC, in order to reduce the quantum circuit cost.

The Reed-Muller form of quantum Boolean circuits typically involves generalized $C^kNOT$ gates depending on the number variables and hence we have to convert each of the $C^kNOT$ gates to equivalent $C^2NOT$ based representation. Figure 7 shows a $C^2NOT$ equivalent circuit for a $C^4NOT$ gate involving two ancillary qbits. The number of $C^2NOT$gate required for a single $C^kNOT$ is $2(k-2)+1$ and the number of ancillary qbits required is k-2, where $k$ is the number of control qbits in the $C^kNOT$ gate.

**Observation:** Any Quantum Boolean Circuit (QBC) can be synthesized using $C^2NOT$, *CNOT*, *NOT* and *SWAP* gates.

The cost of the $C^2NOT$ gate and the 2*2 gates like *CNOT* and *SWAP* can be taken from the basis of technology based implementation of these gates. The related works [11,12] in NMR based quantum computing hardware development suggest that the cost of

- a *1*1* gate is 1 ,
- a *2*2* gate is 5, and
- a 3*3 gate is 5 times that of a *2*2* gate, i.e., 25.

Based on this, we calculate the gate requirement and the quantum gate cost for the different polarities of Reed-Muller circuits in Table 2.

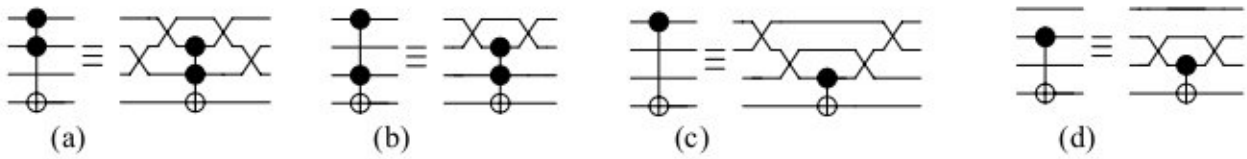


Figure 5: Nearest neighbour configuration templates for
(a) $C^2$NOT(4,3,1), (b) $C^2$NOT(4,2,1), (c) CNOT(4,1), (d) CNOT(3,1)

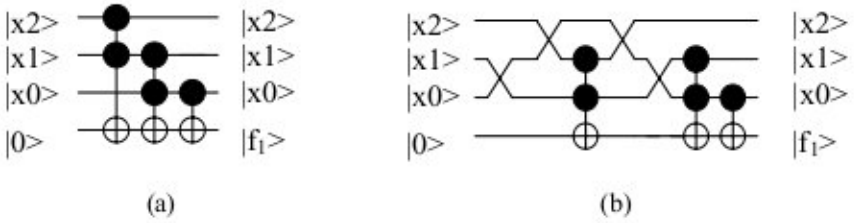


Figure 6: (a) An example QBC (b) Synthesis using nearest neighbor templates

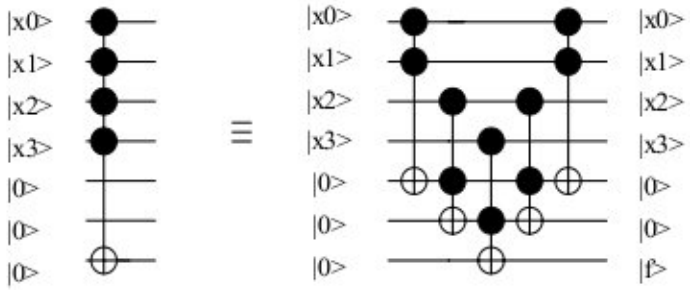


Figure 7: A $C^4NOT$ gate and its equivalent nearest neighbour circuit with $C^2NOT$ gates

Table 2: Gate Count and Quantum Gate cost for different FPRM circuits for $f_1(x_0,x_1,x_2)$

| Polarity | # $C^2$NOT Gates | # CNOT Gates | # SWAP Gates | # NOT Gates | Quantum Gate Cost |
|---|---|---|---|---|---|
| PPRM | 2 | 1 | 4 | 0 | 75 |
| 1 polarity FPRM | 2 | 2 | 6 | 3 | 93 |
| 2 polarity FPRM | 2 | 1 | 8 | 2 | 97 |
| 3 polarity FPRM | 2 | 2 | 10 | 6 | 116 |
| 4 polarity FPRM | 2 | 2 | 6 | 2 | 92 |
| 5 polarity FPRM | 2 | 1 | 4 | 5 | 80 |
| 6 polarity FPRM | 2 | 2 | 10 | 5 | 115 |
| 7 polarity FPRM | 2 | 1 | 8 | 6 | 101 |

## V. CONCLUSION

Our work focuses on defining the nearest neighbour synthesis techniques for quantum Boolean circuits utilizing the Reed-Muller logic decomposition. The proposed circuit synthesis technique utilizes a library of fundamental quantum logic gates consisting of *NOT*, *SWAP*, *CNOT*, $C^2NOT$ gates. The rules for converting general *CNOT*, $C^2NOT$ gates into their nearest neighbour equivalent form can be utilized for the development of low-level circuit synthesis automation tools in the quantum computing domain. Our future work will be defining an advanced search technique, which will evaluate the best circuit for a given Reed-Muller based quantum Boolean circuit in terms of quantum gate cost.

## REFERENCES


[1] P. W. Shor, "Quantum Computing," in *Documenta Mathematica* - Extra Volume ICM, 1998, pp. 1-1000.

[2] R. P. Feynman, "Quantum Mechanical Computers," in *Foundations of Physics*, vol.16, 1986, pp. 507-531.

[3] M. A.Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2002, ch.8.

[4] C.Bennett. "Logical reversibility of computation," in *I.B.M. J. Res. .Dev*., 1973, pp. 525-532.

[5] J. Preskil, "Quantum Computing: Pro and Con," in *Proc. Royal Society*, London, A454, 1998, pp. 469-486.

[6] H. –K. Lo, S. Popescu and T. Spiller, *Introduction to quantum computation and information.* Singapore : World Scientific Publ., 1999.

[7] H.Buhrman, R. Cleve and A. Wigderson. "Quantum vs. classical communication and computation," in *Proc. 30th Annual ACM Symposium on the Theory of Computation*, ACM Press, El Paso, Texas, 1998, pp. 63-68.

[8] R.Landauer. "Irreversibility and heat generation in the computing process," *I.B.M. J. Res. Dev*., 1961, pp.183-191.

[9] A. Younes and J. Miller, " Representation of Boolean Quantum Circuits as Reed Muller Expressions," *arxiv: quant-ph/0304134*, May 2003.

[10] I. A. Grigorenko and D.V. Khveshchenko, "Single-Step Implementation of Universal Quantum Gates," *Physical Review Letters*, 95.110501, 2005.

[11] J.Kim, J-S.Lee, and S.Lee, "Implementation of the refined Deutsch-Jozsa algorithm on a three-bit NMR quantum computer," *Physical Review A*, vol. 62, 022312, 2000.

[12] J.Kim, J-S.Lee, and S.Lee, "Implementing unitary operators in quantum computation," *Physical Review A*, vol. 62, 032312, 2000.