Moving Forward: A Non-Search Based Synthesis Method toward Efficient CNOT-Based Quantum Circuit Synthesis Algorithms

Mehdi Saeedi, Morteza Saheb Zamani, Mehdi Sedighi
Quantum Design Automation Group
Amirkabir University of Technology, Computer Engineering Department, Tehran, Iran
Email: {msaeedi, szamani, msedighi}@ aut.ac.ir

Abstract- Quantum information processing is in the beginning stages. Among open research problems, quantum circuit synthesis has recently received significant attention. In this paper, we propose a new non-search based moving forward synthesis algorithm (MOSAIC) for CNOT-based quantum circuits. Compared with the widely used search-based methods, MOSAIC is guaranteed to produce a result and can lead to a solution with much fewer steps. To evaluate the proposed algorithms, different circuits taken from the literature are used. The experimental results show the efficiency of the proposed algorithm.

I. Introduction

The current ever-shrinking transistor approach will reach its fundamental limits in the near future [1] as the laws of classical physics will not be valid at atomic dimensions. To overcome this limitation, various new computational models have recently been proposed [1]. Among them, quantum computing has the potential to increase the rate of advances in computing power drastically, at least for some problems [2]. The promise of the exponential speedups of quantum algorithms running on quantum computers has intensified the attempts for using quantum algorithms [3], [4].

While quantum mechanics and quantum computing are established research areas [2]-[5], automated quantum circuit synthesis is in the beginning stage and no fully optimized method for quantum circuit synthesis has been proposed yet. To address the problem, in the rest of this paper, we introduce a new moving forward strategy to propose a fast quantum circuit synthesis algorithm. Compared with the recently proposed search-based synthesis methods (as [6], [7] and [9]) our synthesis algorithm can produce a gate-level implementation of the circuit from its matrix representation within much fewer steps.

The rest of the paper is organized as follows: in Section II, a brief introduction to quantum computation is presented. Previous work is reviewed in Section III. Our synthesis method is presented in Section IV. Experimental results are reported in Section V and finally, Section VI concludes the paper.

II. Basic Concepts

Quantum computation uses quantum mechanics to perform a task. A quantum bit (or qubit) is typically derived from the state of a two-level quantum system such as the ground and excited states of an atom. The common notation of a qubit denotes one of these states as $|0\rangle$ and the other as $|1\rangle$. A quantum system with a collection of n qubits is called a quantum register of size n.

Unlike a classical bit, the state of a qubit can take not only two pure states $|0\rangle$ and $|1\rangle$, but also any linear combinations of these pure states, also called superposition, resulting in exponentially larger state space. In other words, the state of a qubit ψ can be written as $\psi=\alpha|0\rangle+\beta|1\rangle$ where α and β are complex numbers and $\alpha^2+\beta^2=1$.

It is common to denote the state of a single qubit by a 2×1 vector as $[\alpha\ \beta]^T$. So, the state of a quantum register of size n can also be shown by an $2^n\times1$ vector $[\alpha_1\ \ \alpha_2\ \ \cdots\ \ \alpha_{2^n}]^T$ where each α_i ($i=1,2,..,2^n$) is a complex number and $\alpha_1^2+\alpha_2^2\ldots+\alpha_{2^n}^2=1$. If only one α_i ($i=1,2,..,2^n$) is set to be one, a pure quantum state of size n is formed.

An n-qubit *quantum gate* is a device which performs a specific unitary operation on its qubits in a fixed period of time. An n-qubit quantum gate has a unitary $2^n\times2^n$ matrix, also called QMatrix [9], describing its functionality. A matrix M is unitary if $MM^+=I$ where M^+ is the conjugate transpose of M and I is the identity matrix. The QMatrix of a quantum circuit is derived from its gate QMatrices using matrix multiplication.

Previously, various quantum gates with different functionalities have been proposed [5]. Among them, CNOT-based gates comprise an important class of quantum gates and often appear in the quantum computing literatures (for example see [5]-[15]) and defined as follows:

.

Definition 1: An n-input, n-output CNOT gate $CNOT_n(x_1,x_2,\ldots,x_n)$ passes the first $n-1$ lines unchanged. These

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

lines are referred to control lines. This gate flips the n^{th} line if the control lines are all one. In other words, we have: $x_{i(out)}=x_i$ $(i<n)$, $x_{n(out)}=x_1x_2\ldots x_{n-1}\oplus x_n$. A general $CNOT_k$ $(k\leq n)$ gate is called *CNOT-based gate* in this paper.

Some authors (for example [12]) assume that complementation can also be internal to a CNOT-based gate. Therefore, it is possible to have a $CNOT_3(a',b',c)$ gate to refer to $c(new)=c\oplus a'b'$, $a(new)=a$ and $b(new)=b$. This assumption is also used in this paper.

It has been shown that CNOT-based quantum circuits, including Boolean reversible circuits, are described by a unique QMatrix representation, called *well-formed QMatrix*, which contains only a single 1 in all columns and rows [9]. Note that an n-input, n-output, fully specified Boolean function is called reversible if it maps each input pattern to a unique output pattern [6], [7]. For the class of Boolean reversible circuits its matrix representation can also be derived from its truth table by inserting a single 1 at row i and column j where the row and column numbers are specified by output and input patterns, respectively.

Quantum circuit synthesis is defined as the ability to automatically generate a quantum circuit from a given specification. As each quantum circuit, including but not limited to Boolean reversible circuits, has a unique QMatrix representation, we use QMatrix as the given specification to propose an efficient synthesis method for CNOT-based quantum and Boolean reversible circuits. In the next section, previous work on automated quantum (reversible) circuit synthesis is reviewed.

III. Previous Work

Several algorithms have recently been proposed to synthesize a quantum circuit. Toffoli in [10] presented an algorithm to implement a function using CNOT-based gates. As this algorithm uses many extra qubits, it cannot be used to synthesize a general quantum circuit efficiently. In [11], a new incremental approach was presented which uses shared binary decision diagrams for representing a reversible function and measuring circuit complexity. The proposed algorithm selects reversible gates based on the complexity of the rest of logic.

Some authors used transformation-based algorithms for quantum circuit synthesis [12]-[15]. However, these algorithms usually use local transformations to optimize the results of other algorithms. In [16], an approach to synthesize a quantum circuit was proposed which uses symbolic reachability analysis where the primary inputs are assumed to be purely binary. In [17], Shende et al. presented a top-down structure based on Cosine-Sine decomposition to introduce quantum multiplexer and used it to propose a synthesis algorithm in terms of quantum multiplexers. In [18] a quantum decision diagram (QDD) structure was introduced and used to synthesize quantum circuits using $R_x(\theta)$ rotation gates. However, they assumed that the QDD control variables are pure quantum states.

As the size of a quantum circuit can be large, a practical algorithm for quantum circuit synthesis may become extremely difficult. Due to the lack of a mature systematic method, search-based algorithms are widely used [6]-[9] for quantum and Boolean reversible circuit synthesis where an extensive exploration is required to find a possible implementation of the circuit. This method was used in [6] to propose a synthesis algorithm for reversible circuits based on the Reed-Muller expansion. This algorithm was further improved in [7] to reach a result within fewer searches.

The authors of [19] presented an algorithm to decompose the matrix of a quantum circuit into the unitary matrices of elementary gates. However, these methods are not practical to synthesize a general quantum circuit of arbitrary size. The authors of [20] studied the maximum number of required gates to synthesize a reversible circuit. They also proposed a decomposition method for linear reversible circuits. In [9], the matrix characterizations of CNOT-based reversible circuits are studied and used to propose a multi-stage synthesis methodology for quantum circuits where a search-based algorithm was used to reach a result. So, their results are also limited to small and medium size CNOT-based quantum circuits.

As the size of a quantum circuit increases drastically, a practical algorithm for quantum circuit synthesis becomes extremely difficult. In the following section, we propose a new non-search based synthesis algorithm which uses several predefined steps to attain a result from its matrix representation.

IV. Synthesis Algorithm

As stated before, several authors assumed that complementation can also be internal to a CNOT-based gate [12]. On the other hand, several others (for example [6], [7] and [9]) use only CNOT-based gates with positive control lines. In this section, we propose a *moving forward synthesis algorithm with internal complementation*, *MOSAIC,* to synthesize CNOT-based quantum circuits efficiently. However, we also introduce a transformation method to attain a positive control CNOT-based circuit.

As only CNOT-based quantum and Boolean reversible circuits are used in this paper, for the following sections, the terms gates and circuits are used for CNOT-based gates and CNOT-based circuits, respectively. Moreover, well-formed QMatrix is denoted as QMatrix in short. Consider the following definitions:

Definition 2: The application of a k-qubit gate with matrix G on a quantum circuit with a QMatrix M is called L_k *QTranslation.*

It can be seen that the result of using an L_k QTranslation is the same as multiplication of M by G, i.e. MG. As each k-qubit gate has a well-formed QMatrix [9], the result of using an L_k QTranslation is also well-formed.

Definition 3: Consider the j^{th} and the i^{th} rows of a QMatrix M. These two rows form a *quantum pair* ($QPair_{i,j}$) if the numbers i and j differ in only one bit position. For example, the 2nd (010) and the 6th (110) rows belong to $QPair_{2,6}$.

Definition 4: The 2^k rows of a QMatrix the row numbers of which have the same value on their n-k bit locations form a single group called C^kQPair. For the case of $k=1$, each C^kQPair contains only one QPair.

Lemma 1 and Lemma 2 explain the results of using an L_k QTranslation on a given QMatrix.

Lemma 1: (a) Consider an n-qubit quantum circuit with QMatrix M. The application of an L_k QTranslation on M leads to 2^{n-k-1} row exchanges. (b) Equally, exchanging the locations of 2^k QPairs of the same C^kQPair is equivalent to applying an L_{n-k-1} QTranslation. □

Lemma 2: Consider a C^kQPair of a $2^n \times 2^n$ QMatrix having 2^k rows $r_1, r_2, ..., r_{2^k}$ where the n-bit numbers $r_1, r_2, ..., r_{2^k}$ have the same value on their n-k bits $b_1, b_2, \ldots, b_{n-k}$ and two QPair rows differ from each other only in one bit position b_m. Exchanging the locations of each $QPair_{i,j}$ ($QPair_{i,j} \in C^kQPair$, $i, j \in \{r_1, r_2, ..., r_{2^k}\}$) has the same result as applying an L_{n-k+1} QTranslation $CNOT_k(x_{b_1}, x_{b_2}, ..., x_{b_{n-k}}, x_{b_m})$. For the case of $n=k$, a simple NOT gate is constructed. □

Due to the paper page limit, the proofs of the previous lemmas are omitted and are available from the authors upon request. In the following subsection, our synthesis algorithm is proposed.

A. The MOSAIC algorithm

The goal of MOSAIC is to decompose a given QMatrix into several elementary QMatrices of CNOT-based gates efficiently. After the decomposition, the resulted QMatrices are translated into the final circuit using Lemma 2. Fig. 1 shows the proposed MOSAIC algorithm. As shown in this figure, the algorithm starts with the first bit of the first column and finishes when the i^{th} column has a value of 1 in its i^{th} row for all $i \in (1..2^n)$. Consider the following example for more details. To save space, the QMatrix A is denoted as $A(x_1, x_1, ..., x_{2^n})$ where x_i $(i \in [1,2^n])$ is the row number of an element with the value of 1 in the i^{th} column.

Example 1: Consider a 3-qubit circuit with QMatrix A(7,0,1,2,3,4,5,6). Fig. 2 shows the results of the proposed algorithm. In this figure, the numbers p, q and the exchanged QPair rows enclosed in an '{}' symbol are shown. Furthermore, the resulted $C^kQPairs$ for each step are also demonstrated. It can be verified that the final QMatrix after each step are A_1=(6,1,0,3,2,5,4,7), A_2=(4,1,2,3,0,5,6,7) and A_3=(0,1,2,3,4,5,6,7). The synthesized circuit of this example is *{NOT(c), CNOT(c',b), C2NOT(b',c',a)}*. □

Consider the 2^k rows $r_1, r_2, ..., r_{2^k}$ of a QMatrix which belong to the same C^kQPair set. It can be checked that $r_1, r_2, ..., r_{2^k}$ have the same value in the specific n-k bit locations. However, the value of each bit could be 0 or 1 independent of the other n-k-1 bit locations. As a result, the final circuit could have both positive and negative control lines. In the following, we introduce a transformation method to attain a positive control gate.

Algorithm MOSAIC

Input: A $2^n \times 2^n$ QMatrix M
Output: A CNOT-based decomposition of M

```
1.  b=1;
2.  repeat
3.    reset all rows of M to be unvisited;
4.    flag=true;
5.    for each column index c of M, c∈(1..2^n)
6.      set r to be the c row number which has a value of 1;
7.      if the r^th row is not marked as visited then
8.        if the b^th bits of r and c are not equal then
9.          flag=false;
10.           find the number p ∈ QPair_{r,p} which differs with r in its b^th bit;
11.           set q to be the column number of row p which has a value of 1;
12.           if q = p  and p < r then
13.              do nothing;
14.           else
15.              exchange the locations of the p^th and r^th rows;
16.              mark the p^th and r^th rows as visited;
17.           end if
18.       end if
19.     end if
20.   end for
21.   b = (b + 1) mod n;
22.   Identify each C^kQPair group whose QPairs are exchanged;
23.   Extract an L_k translation for each C^kQPair based on Lemmas 1 and 2;
24. until flag=true;
```

Fig. 1- The proposed MOSAIC synthesis algorithm

Consider a general CNOT-based gate with both positive and negative controls. As the control and target qubits have no shared variable, it is possible to decompose this gate into several NOT gates followed by one positive control gate. Consider the following example for more detail:

Example 2: Consider the QMatrix A(7,0,1,2,3,4,5,6) of Example 1 and the resulted circuit *{NOT(c), CNOT(c',b), C2NOT(b',c',a)}*. The results of using the above method to reach positive control gates are:

$$NOT(c)$$
$$CNOT(c', b) \rightarrow NOT(c), CNOT(c, b)$$
$$C2NOT(b', c', a) \rightarrow NOT(b), NOT(c), C2NOT(b, c, a)$$

Therefore, the final circuit is *{NOT(c), NOT(c), CNOT(c,b), NOT(b), NOT(c), C2NOT(b,c,a)}*. □

It can be verified that the application of the transformation method may lead to several redundant gates. For example, the result of Example 2 could be reduced to *{CNOT(c,b), NOT(b), NOT(c), C2NOT(b,c,a)}*. Therefore, after each transformation, a simple redundancy elimination technique may also be needed.

B. The Algorithm Convergence

The MOSAIC algorithm uses an iterative approach to reach a result. Theorem 1 guaranties the algorithm convergence:

$$\overbrace{c=0 \rightarrow r=7 \rightarrow p=6 \rightarrow q=7 \Rightarrow \{7,6\}}^{b=0}$$
$$c=1 \rightarrow r=0 \rightarrow p=1 \rightarrow q=2 \Rightarrow \{0,1\}$$
$$c=2 \rightarrow r=1 \text{(visited)}$$
$$c=3 \rightarrow r=2 \rightarrow p=3 \rightarrow q=4 \Rightarrow \{2,3\}$$
$$c=4 \rightarrow r=3 \text{(visited)}$$
$$c=5 \rightarrow r=4 \rightarrow p=5 \rightarrow q=6 \Rightarrow \{4,5\}$$
$$c=6 \rightarrow r=5 \text{(visited)}$$
$$c=7 \rightarrow r=6 \text{(visited)}$$
$$C^3QPair=[0,1,2,3,4,5,6,7]$$

$$\overbrace{c=0 \rightarrow r=6 \rightarrow p=4 \rightarrow q=6 \Rightarrow \{6,4\}}^{b=1}$$
$$c=1 \rightarrow r=1$$
$$c=2 \rightarrow r=0 \rightarrow p=2 \rightarrow q=4 \Rightarrow \{0,2\}$$
$$c=3 \rightarrow r=3$$
$$c=4 \rightarrow r=2 \text{(visited)}$$
$$c=5 \rightarrow r=5$$
$$c=6 \rightarrow r=4 \text{(visited)}$$
$$c=7 \rightarrow r=7$$
$$C^2QPair=[0,2,4,6]$$

$$\overbrace{c=0 \rightarrow r=4 \rightarrow p=0 \rightarrow q=4 \Rightarrow \{4,0\}}^{b=2}$$
$$c=1 \rightarrow r=1$$
$$c=2 \rightarrow r=2$$
$$c=3 \rightarrow r=3$$
$$c=4 \rightarrow r=0 \text{(visited)}$$
$$c=5 \rightarrow r=5$$
$$c=6 \rightarrow r=6$$
$$c=7 \rightarrow r=7$$
$$C^1QPair=[0,4]$$

Fig. 2- The results of applying the MOSAIC algorithm on the specification of Example 1.

Theorem 1: The MOSAIC algorithm will converge to a possible implementation after a finite number of steps.

Proof: Consider a QMatrix M of size 2^n. Assume that after a number of steps, several rows represented as a set Σ, are placed at their right positions. Furthermore, suppose that the algorithm is working on the k^{th} bit (*i.e. b=k and k≤n)* of the c^{th} column and sets *r* to the column *c* row number with the value of 1. Moreover, consider the case where *r* differs from *c* in its k^{th} bit (i.e. $r \notin \Sigma$). Accordingly, the algorithm finds a row number *p* that differs from *r* only in its k^{th} bit.

If $p \in \Sigma$ and $p<r$ (line 9 in Fig. 1), the algorithm does nothing to avoid instability in row locations. However, as the r^{th} row is placed at a wrong position (for example, the position of the t^{th} row, $t \notin \Sigma$), there must be another row, i.e. the t^{th} row, which should be exchanged with the r^{th} row during the next steps. Therefore, the algorithm does not finish at the current step and it will reach the other cases, i.e. $p \notin \Sigma$ or *$(p \in \Sigma$ and $p>r)$.*

Consider the other cases *$(p \notin \Sigma$ or $(p \in \Sigma$ and $p>r))$* where the algorithm exchanges the location of the p^{th} row with that of the r^{th} row. Then, the k^{th} bit of the row *r* will be correct and the algorithm moves forward to check other rows as well as other bits. As each QTranslation does not change the results of the previous ones, i.e. each QTranslation changes only one qubit, the algorithm will gradually place all rows at their right positions. Therefore, the algorithm will lead to a valid result after several steps. □

Based on Theorem 1, it can be said that the MOSAIC algorithm always converges to a possible synthesized result. In the following section, we propose a worst-case analysis to compare the time complexity of the MOSAIC algorithm with that of search-based methods proposed in literature.

C. The Time Complexity of the Algorithm

In order to compare the MOSAIC algorithm with search based methods with respect to their time complexity, assume that a possible implementation of a $2^n \times 2^n$ QMatirx *M* needs at most *h* CNOT-based gates.

Theorem 2: A search-based synthesis method needs at most $O(n \times 2^n)^h$ steps to reach a result.

Proof: For a quantum circuit of size *n*, there are C_n^1 possible NOT gates and C_n^2 possible C2NOT gates in which one of its two inputs can be the target output. On the other hand, as each of the C2NOT inputs could be used as the target qubit, the total number of $2 \times C_n^2$ gates can be attained.

In contrast, for a *k*-qubit gate, $k \in (3, 4, ..., n-1)$, there are C_{n-1}^k possible gates when target can be any i^{th} $(i \in [1,n])$ qubit. Considering all possible qubits as the target variable leads to the total number of $n \times C_{n-1}^k$ *k*-qubit gates. Therefore, the total number of gates are $n \times 2^{n-1}$ as $C_n^1 + 2 \times C_n^2 + n \times (C_{n-1}^3 + ... + C_{n-1}^{n-1}) = n \times 2^{n-1}$. Since at most *h* steps are required to reach a result, it can be said that search-based methods need at most $(n \times 2^{n-1})^h$ or $O(n \times 2^n)^h$ node searches to synthesize a given specification. □

Theorem 3: The proposed MOSAIC algorithm needs at most $O(h \times 2^n)$ steps to reach a result.

Proof: It can be verified that except the lines 2 and 5 in Fig. 1, the other lines take only O(1) time complexity. On the other hand, the time complexity of line 5 (internal loop on column index) is $O(2^n)$. Furthermore, the outer loop (line 2) has O(h) time complexity. As a result, the MOSAIC algorithm needs at most $O(h \times 2^n)$ step to reach a result. Moreover, the proposed transformation technique and the final elimination method need at most $O(h)$ steps. Therefore, the time complexity of our algorithm is $O(h \times 2^n)$. □

Compared with the search-based methods [6]-[7], [9], the MOSAIC algorithm needs much fewer steps to synthesize a given specification. In the following section, the experimental results are shown.

V. Experimental Results

The proposed algorithm was implemented in MATLAB and all of the experiments were done on an Intel Pentium IV 3GHz computer with 1GB memory. To evaluate the MOSAIC algorithm, we use the same sixteen circuits of [7] and several new randomly generated circuits. Furthermore, we compare the results of our algorithm with two recent papers [6] and [7] in terms of the number of searched nodes and the produced gate counts. The results of these comparisons are shown in Table 1. As shown in this table, the proposed MOSAIC algorithm not only has the ability to produce a result for all of the attempted specifications but also can reach a result with much fewer steps (more than 100 times faster on average). Since the specification of search space affects the number of searched nodes, for a few benchmarks, i.e. 3, 6 and 7, MOSAIC requires more steps to reach a result. It is important to note that both primitive operations (i.e. node search for search-based methods and step for MOSAIC) have the same *O(1)* time complexity. While MOSAIC needs one second to synthesize each benchmark on average, the search-based methods consume much more times to lead a result. Furthermore, it can be verified that the MOSAIC algorithm can also reach a circuit with comparable cost (9.81 vs. 7.62) within a small number of steps.

To test the scalability of the proposed algorithm for different input sizes, several other experiments were made. We generated 10 random circuits of different input sizes (from 1 to 12) and use the MOSAIC algorithm to synthesize each circuit, separately. For each input size, the number of required steps and the number of gates are shown in Table 2. The CPU time for each input size is also reported in this table. It can be seen that the MOSAIC algorithm can synthesize the attempted circuits quickly.

Compared with the best possible implementation of each selected QMatrix, the gate counts of several circuits are more than the results of an exhaustive search-based method for about 2.2% on average. The natural next step for future work seems to be working on the improvement of the resulting synthesized circuits possibly by combining the proposed approach and the search-based methods. Efforts to reach this goal are under way.

VI. Conclusions

In this paper, a new non-search based synthesis algorithm was proposed which requires a few steps to synthesize a given specification. To evaluate the algorithm, we used sixteen examples taken from the literature and compared the results with those from two recent search-based methods. It was shown that the presented algorithm can lead to a result for all of the circuits 113 times faster, on average. In addition, to evaluate the scalability of the algorithm, several random circuits with up to 12 inputs were generated. The results of our experiments illustrated that for large QMatrices, our algorithm needs about 1 minute to reach a result.

References

[1] International Technology Roadmap for Semiconductors, 2005 Edition.

[2] P. E. Black, D. R. Kuhn, and C. J. Williams, "Quantum Computing and Communication," *Advances in Computers*, Academic Press, vol. 56, pp. 189-244, 2002.

[3] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," *in 28th Annual ACM Symposium on Theory of Computing*, pp. 212-219, 1996.

[4] P. W. Shor, "Polynomial Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, 26(5), pp. 1484-1509, 1997.

[5] M. A. Nielsen, I. L. Chuang, "Quantum Computation and Quantum Information," *Cambridge University Press*, 2000.

[6] P. Gupta, A. Agrawal, and N. K Jha, "An Algorithm for Synthesis of Reversible Logic Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, November 2006.

[7] M. Saeedi, M. Saheb Zamani, M. Sedighi, "On the Behavior of Substitution-Based Reversible Circuit Synthesis Algorithms: Investigation and Improvement," *International Symposium on VLSI*, 2007.

[8] D. M. Miller, "Spectral and Two-Place Decomposition Techniques in Reversible Logic," *45th Midwest Symposium on Circuits and Systems*, pp. II-493-II-496, 2002.

[9] M. Saeedi, M. Sedighi, M. Saheb Zamani, "A New Methodology for Quantum Circuit Synthesis: CNOT-Based Circuits as an Example," *International Workshop on Logic Synthesis*, 2007.

[10] T. Toffoli, "Reversible Computing," MIT, Tech. Rep., 1980.

[11] P. Kerntopf, "A New Heuristic Algorithm for Reversible Logic Synthesis," *Design Automation Conference*, pp. 834-837, 2004.

[12] D. Maslov, C. Young, D. M. Miller, and G. W. Dueck, "Quantum Circuit Simplification Using Templates," *Design Automation and Test in Europe*, pp. 1208-1213, 2005.

[13] K. Iwama, and Y. Kambayashi, and S. Yamashita, "Transformation Rules for Designing CNOT-Based Quantum Circuits," *Design Automation Conference*, pp.419-424, 2002.

[14] D. Maslov, C. Young, D. M. Miller, and G. W. Dueck, "Quantum Circuit Simplification Using Templates," *Design Automation and Test in Europe*, pp. 1208-1213, 2005.

[15] D. M. Miller, D. Maslov, and G. W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," *Design Automation Conference*, pp. 318-323, 2003.

[16] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Quantum Logic Synthesis by Symbolic Reachability Analysis," *Design Automation Conference*, pp. 838-841, 2004.

[17] V. V. Shende, S. S. Bullock, I. L. Markov, "Synthesis of Quantum Logic Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 25(6), pp. 1000-1010, June 2006.

[18] A. Abdollahi, and M. Pedram, "Analysis and Synthesis of Quantum Circuits by Using Quantum Decision Diagrams," *Design Automation and Test in Europe*, pp. 317-322, 2006.

[19] J. J. Vartiainen, M. Mottonen, and M. M. Salomaa. "Efficient Decomposition of Quantum Gates," *Phys. Rev. Let.*, 92:177902, 2004.

[20] K. N. Patel, I. L. Markov, J. P. Heyes, "Efficient Synthesis of Linear Reversible Circuits," *arXiv:quant-ph/0302002v.*

Table 1- The results of using the proposed synthesis methods compared with two recent methods [6],[7], [9]

Circuit #		QMatrix	Number of Searched Nodes [6],[7], [9] & Steps (MOSAIC)			Number of Gates	
			MOSAIC	[6], [9]	[7]	MOSAIC	[6],[7], [9]
[7]	1	(1,0,3,2,5,7,4,6)	40	11	15	4	4
	2	(7,0,1,2,3,4,5,6)	24	761	300	3	3
	3	(0,1,2,3,4,6,5,7)	32	7	10	3	3
	4	(0,1,2,4,3,5,6,7)	64	156	786	7	5
	5	(0,1,2,3,4,5,6,8,7,9,10,11,12,13,14,15)	160	9515	8256	9	7
	6	(1,2,3,4,5,6,7,0)	24	4	4	3	3
	7	(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0)	64	5	5	4	4
	8	(0,7,6,9,4,11,10,13,8,15,14,1,12,3,2,5)	64	230	139	4	4
	9	(3,6,2,5,7,1,0,4)	56	-	66	8	7
	10	(1,2,7,5,6,3,0,4)	48	-	77	8	6
	11	(4,3,0,2,7,5,6,1)	56	-	4387	6	7
	12	(7,5,2,4,6,1,0,3)	32	-	352	6	7
	13	(6,2,14,13,3,11,10,7,0,5,8,1,15,12,4,9)	192	-	678	19	15
	14	(9,7,13,10,4,2,14,3,0,12,6,8,15,11,1,5)	240	-	9712	23	14
	15	(6,4,11,0,9,8,12,2,15,5,3,7,10,13,14,1)	192	-	74521	21	17
	16	(13,1,14,0,9,2,15,6,12,8,11,3,4,5,7,10)	352	-	85191	29	16
Average			102	-	11531	9.81	7.62

Table 2- The results of using the proposed synthesis method to synthesize different size QMatrices

Inputs	Number of Steps	Number of Gates	CPU Time (seconds)	Inputs	Number of Steps	Number of Gates	CPU Time (seconds)
1	1	1	0	2	7	2	0
3	34	4	0	4	155	9	0.01
5	624	17	0.05	6	2265	30	0.17
7	7731	55	0.51	8	24422	84	1.65
9	72960	133	5.46	10	225280	206	17.27
11	581632	259	45.39	12	1277952	312	61.50