Logic Synthesis of Ternary Quantum Circuits with Minimal Qutrits

Xiaoyu Li, Guowu Yang, Desheng Zheng
School of Computer Science and Engineering, University of Electronics Science and Technology of China;
Department of Electrical Engineering, University of California at Los Angeles, USA;
Email: erin.xiaoyu.li@gmail.com

Abstract— **This paper investigates a universal approach of synthesizing arbitrary ternary logic circuits in quantum computation based on the truth table technology. It takes into account of the relationship of classical logic and quantum logic circuits. By adding inputs with constant value and garbage outputs, the classical non-reversible logic can be transformed into reversible logic. Combined with group theory, it provides an algorithm using the ternary Swap gate, ternary NOT gate and ternary Toffoli gate library. Simultaneously, the main result shows that the numbers of qutrits we use are minimal compared to other methods. We also illustrate with two examples to test our approach.**

Index Terms— **ternary quantum computation, reversible logic, circuit synthesis, group theory**

I. Introduction

Fast synthesis of reversible logic circuits in quantum computation has been a hot research topic [1]–[4], in recent years. Reversible logic circuit is one of its important subclass. Its realization is the fundamental theory of quantum computer [5], [6]. Since 1980, there has been a fairly extensive literature about reversible computation [7], [8]. Reversible computation and energy consumption of calculation have a profound relationship, as implied by the Landauer Principle [9] and Moore's Law [10]. This is the reason that the application of reversible operation becomes more attractive [11]. Synthesis of reversible logic is mainly motivated in low-power computing, quantum computing, DNA computing, and nanotechnologies.

In quantum computation and quantum cryptography, multi-valued quantum computing is gaining importance, because it represents one kind of $n-dimensional$ quantum systems with the basis state $|0\rangle, |1\rangle, \ldots, |n-1\rangle$. Generally, the information's unit in a multi-valued quantum system is called a qudit, for the binary case, qubit, for the ternary case, qutrit. If there are n qutrits in a quantum register, it can hold 3^n simultaneous values. Meanwhile, the n qubits can only hold 2^n values. Zilic and Radecka have proposed that for the same using memory, the method of ternary Quantum Fourier Transform (QFT) can improve the approximation and add state space in a factor: $(3/2)^n$ [12]. The ternary valued reversible circuit has been experimentally feasible under the context using linear ion trap schemes of quantum computation [13]. In the literatures, the universality has been assumed that the gates input is set as constant value, and ancilla bit has been used [14], [15]. When synthesizing a non-reversible logic circuit, we should use ancilla bits and some garbage outputs [16]–[18] to transform it to its reversible circuit firstly. A proposal for how to realize the $n-bit$ binary non-reversible logic circuits or even reversible circuits using the library of Controlled-Not, Not and Peres gates (CNP) has been presented in [16]. It constructively has been proved that all $n-qutrit$ ternary reversible circuits could be constructed by ternary Swap, ternary Not and ternary Toffoli ($(n-1)$ qutrits control another qutrits, so it is also an $n-qutrit$ gate) gates without ancilla qutrits, or by ternary NOT, controlled-NOT, multiply-two and Toffoli gates [17]. It has been demonstrated the $2-qutrit$ ternary reversible gates' universality [18]. Then, it lists two conjectures for the properties of $d-level$ reversible gates.

Group theory is found particularly useful in analysis of reversible logic circuits [19]. Numerous studies have proposed the methods about how to apply group theory on reversible logic gates synthesis [2], [3], [20]–[22]. Based on group theory, the purpose of the paper is about synthesis of ternary non-reversible logic circuits using the library of ternary Swap gates, ternary NOT gates and ternary Toffoli gates (SNT). We found the relationship about the number of ancilla input and garbage output necessary for achieving reversibility. Moreover, we design an algorithm 'Synthesize the $t \times s$ ternary non-reversible logic circuits' (STNC) for our main result that is provably convergent.

The structure of this paper is as follows. In section II, the definitions of basic ternary reversible gates are given. The main results of the paper are given in section III. The required notations and terms of group theory are introduced in the following section. Using group theory, we develop an algorithm to synthesize reversible logic circuits with the SNT library. In section V, we give two examples to demonstrate our results and illustrate how the algorithm synthesizes ternary non-reversible logic circuits. We also discuss the differences among our results, [11], [23] and [4]. We conclude the paper in section VI.

This work was supported by the Fundamental Research Funds for the Central Universities under Grant No. *ZYGX*2011*YB*022, in part by the National Science Foundation of China under Grant No. 60773205 and 973 Foundation (2010CB328004).

doi:10.4304/jcp.8.8.1941-1946

II. Ternary Reversible Gates

Definition 1: (ternary reversible gate) [24] Given a ternary logic function f with n inputs and outputs, B_1, B_2, ..., B_n and P_1, P_2, ..., P_n, respectively, function f is $f : B^n \rightarrow P^n$, where $(B_1, B_2, \ldots, B_n) \in B^n$ and $(P_1, P_2, \ldots, P_n) \in P^n$ are the input and output vectors, respectively. The ternary logic function f is defined be reversible if it is one-to-one and onto (bijection).

Definition 2: (ternary NOT gate) [24] Given a ternary NOT Gate N_j, it has been defined as: $P_j = B_j \oplus_3 1$, where $\oplus_3$ is addition modulo 3; $P_i = B_i$, if $i \neq j$, $1 \leq j \leq n$.

Definition 3: (ternary $n \times n$ Toffoli gate) [24] Given a ternary $n \times n$ Toffoli Gate, it has been defined as: if $B_2 = B_3 = ... = B_n = 2$, then $P_1 = B_1 \oplus_3 1$; otherwise, $P_1 = B_1$, where $P_i = B_i$, for $i \neq 1$ (shown as Fig. 1). That is, it maps $d_1 \rightarrow d_2$, $d_2 \rightarrow d_3$, $d_3 \rightarrow d_1$, ... respectively, where $d_1 = (0, 1, \ldots, 1)$, $d_2 = (1, 1, \ldots, 1)$, $d_3 = (2, 1, \ldots, 1)$, and without changing other assignments.

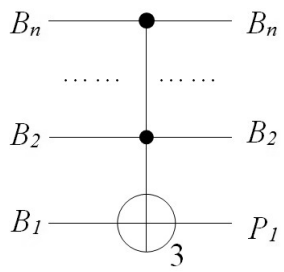

Figure 1. ternary $n \times n$ Toffoli gate

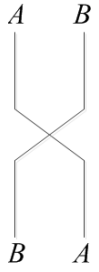

Figure 2. Swap gate

Definition 4: (swap gate) [24] Given a swap gate $E_{i,j}$, it exchanges the i^{th} bit and the j^{th} bit, A_i and A_j, respectively. For example, $P_i = A_j$, $P_j = A_i$; $P_r = A_r$, if $r \neq i$, j (shown as Fig. 2).

Definition 5: (library) [24] Given a $w - library$, it is a set of $w \times w$ reversible gates that has been used to synthesize $w \times w$ reversible circuits, simply as L. A reversible circuit f can be synthesized by L, that is, f can be constructed by elements in L.

By far, there is no universal gate library. When synthesizing different logic circuits, L could be constructed by different gates. In our paper, we use SNT library to synthesize the quantum ternary reversible circuits. Furthermore, in [11] the gates library is made up of the New gate, Feynman Gate and the Toffoli Gate. Meanwhile, the library of [23] contains the Generalized Ternary Gate and the newly defined permutative quantum ternary C^2NOT gate.

III. Main Results

Previous works [16] have presented some synthesis properties of binary non-reversible logic circuits as blow.

Lemma 1: Let f be binary classical non-reversible logic circuits, where f stands for a function. Suppose that the t inputs are B_1, ..., B_t, the s outputs are P_1, ..., P_s ($s \leq t$). M should be the truth table of function f. From the table of M, there exits r rows with the same outputs. This class of circuits could be synthesized by adding $\lceil \log_2 r \rceil$ $s - t$ inputs with constant 0 and $\lceil \log_2 r \rceil$ garbage output. Meanwhile, this could be realized by the Controlled-Not gate, Not gate and Peres gate (CNP) library.

Proof: The exactly proof of Lemma 1 could be found in [16]. □

Further generalized this Lemma, we can consider the case of ternary values. Different from the binary values, here we use SNT instead of CNP library to deal with the arbitrary ternary non-reversible logic circuit.

Lemma 2: Every $n \times n$ ternary reversible logic circuit could be generated by the ternary Swap gates, NOT gates and Toffoli gates.

Proof: It could be found that the proof of Lemma 2 is in [17]. □

Theorem 1: Let f be ternary classical non-reversible logic circuits, where f stands for a function. Suppose that the t inputs are B_1, ..., B_t, the s outputs are P_1, ..., P_s ($s \leq t$). M should be the truth table of function f. From the table of M, there exits r rows with the same outputs at most. This class of circuits could be synthesized by adding $\lceil \log_3 r \rceil$ $s-t$ (if $\lceil \log_3 r \rceil > (t-s)$, else 0) inputs with constant values. Meanwhile, with $\lceil \log_3 r \rceil$ garbage output, the function f could be realized by the SNT library of ternary Swap gates, NOT gates and Toffoli gates .

Proof: Firstly, based on Lemma 1 and 2, it could be seen that with the SNT library the classical ternary non-reversible logic function f could be realized.

Second, we have to proof that the amount of garbage outputs can be expressed by $\lceil \log_3 r \rceil$in two cases.

Case a: consider that in the truth table of M, there exits r ($r = 3^k$, $k \geq 1$, and k is integer) rows which are of the same output values. Obviously, for identifying the r outputs with same values, it needs $\log_3 r$ variables. As a result, $\log_3 r$ garbage outputs would be produces.

Case b: consider in M, the case is that there are r rows($r = m + 3^k$, where m and k are integers, and $k \geq 1$, $3^k > m \geq 1$) which have the same output values. If the same outputs could be divided into different outputs, it should add $k + 1$ variables. In the other hands, in order to identify these outputs, $\lceil \log_3 r \rceil$ garbage outputs are required.

Third, refer to the amount of adding input values. Following the property of a logic reversible circuit, the number of inputs and outputs must be the same. In our transformation it needs a certain number of inputs and garbage outputs of $\lceil \log_3 r \rceil$. Based on this relationship, we can calculate that $\lceil \log_3 r \rceil$ $s - t$(if $\lceil \log_3 r \rceil > t - s$, else 0) inputs are needed.

Therefore, the proof process of the theorem is completed. □

Remark 1: Not all gates in the library SNT will be used to synthesize some circuits. However, the SNT library is the maximal set for all the ternary reversible circuits. This can be seen in example 1 where the Swap gate is not used.

Corollary 1: Suppose that the function f stands for the ternary non-reversible circuit. According to Theorem 1, it

can be transformed to reversibility by adding $s+\lceil\log_3 r\rceil-t$ (if $(t-s) < \lceil\log_3 r\rceil$, else zero) inputs with constant values and $\lceil\log_3 r\rceil$ garbage outputs. The qutrits of ancilla inputs and garbage outputs are minimal.

Proof: From the intuitive insight of the proof for Theorem 1, we obtain that the garbage outputs' number is minimal directly. The basic property of reversible logic is that between inputs and outputs have an one to one correspondence mapping. That means the amount of input and output is same. So the number of ancilla inputs is minimal also.

More explicitly, suppose that f can be synthesized with ($\lceil\log_3 r\rceil$-1) qutrits. This means ($\lceil\log_3 r\rceil$-1) qutrits could identify the same r outputs. If this is true, the rule of one-to-one correspondence mapping between inputs and outputs is violated. Therefor the numbers are minimal.

The result of the corollary is proved. □

IV. Permutation Group and Synthesis Algorithm

Definition 6: ($j-cycle$) [24] Given S_k as a symmetric group of symbols $\{d_1, d_2, d_3, \ldots, d_k\}$, then $(d_{i_1}, d_{i_2}, d_{i_3}, ..., d_{i_k})$ is a $j-cycle$, where $j \leq k$. For particularly, a $(d_i, d_{i+1}, d_{i+2}, ..., d_{i+j-1})$ of $j-cycle$ is a neighbor $j-cycle$ of S_k, for $\forall 1 \leq i \leq k-j+1$.

Definition 7: (permutation) [24] The bijection of M onto itself is a permutation for M, where $M = \{d_1, d_2, d_3, ..., d_k\}$. The sets of all permutations on M compose a group under mapping composition, called as a symmetric group on M [19]. A permutation group is simplified as a subgroup [25] of a symmetric group.

Definition 8: (even permutation and odd permutation) [24] Given a permutation, it is a product of an even number of $2-cycles$ and odd if it is a product of an odd number of $2-cycles$.

Remark 2: Furthermore, as the similarly rules, it could be summarized as followed. An odd permutation could be obtained either by the product of odd numbers of odd permutation or odd numbers of odd permutation with even numbers of even permutation. An even permutation could be obtained either by the product of even numbers of even permutation or even numbers of odd permutation.

Some other notations are also needed. For instance, a mapping relationship of s: $M \rightarrow M$ could be expressed as

$$s = \begin{pmatrix} d_1, & d_2, & ..., & d_k \\ d_{i_1}, & d_{i_2}, & ..., & d_{i_k} \end{pmatrix}, \tag{1}$$

Here another notation will be written as the product of disjoint cycles [25]. Take an example as followed,

$$\begin{pmatrix} d_1, & d_2, & d_3, & d_4, & d_5, & d_6, & d_7, & d_8, & d_9 \\ d_1, & d_4, & d_7, & d_2, & d_5, & d_8, & d_3, & d_6, & d_9 \end{pmatrix}, \tag{2}$$

could be written as $(d_2, d_4)(d_3, d_7)(d_6, d_8)$. The identity mapping relationship of '( )' (direct wiring) is named the basic unity cell in some certain permutation groups. For convenient, it would be denoted as s^{-1} for inverse mapping of s. As per convention, the product $s \times t$ means two permutations applying mapping the s before t.

Algorithm 1 Synthesis of Arbitrary $t \times s$ Ternary Non-reversible Logic Circuits of f (STNC)

Input: Library L and Non-reversible logic circuit f.
Output: Synthesis of Reversible logic circuit f';

1: According to Theorem 1, transform a $t \times s$ ternary non-reversible f to reversible f';
2: $f' = apply\_group\_theory(f)$
3: **if** ($f' = odd\_permutation$) **then**
4: $E_{1,2} * f' = L_1 * L_2 * \ldots * L_h$, $L_i \in \{ternary\ Swap\ gate, NOT\ gate, Toffoli\ gate\}$
5: **return** $f' = E_{1,2} * L_1 * \ldots * L_h$.
6: **else if** ($f' = even\_permutation$) **then**
7: $f' = C_1 * C_2 * \ldots * C_s$, C_i is $3-cycle$
8: $C_i = L_{i,1} * L_{i,2} * \ldots * L_{i,t_i}$; $1 \leq i \leq s$, $L_{i,j} \in \{ternary\ Swap\ gate, NOT\ gate, Toffoli\ gate\}$
9: **return** $f' = [L_{1,1} * L_{1,2} * \ldots * L_{1,t_1}] * \ldots * [L_{s,1} * L_{s,2} * \ldots * L_{s,t_s}]$.
10: **end if**

According to the Definition 6, 7 and 8, we could obtain that the relationships of permutations and reversible logic circuits as followed. Now we would order 3^n different of n inputs assignment vectors as Eq. 3 shown

$$\begin{gathered}(0,0,\ldots,0),(1,0,\ldots,0),(2,0,\ldots,0),\\ (0,1,\ldots,0),\ \ldots,\ (2,2,\ldots,2),\end{gathered} \tag{3}$$

and denote them by the order of a_1, a_2, a_3, …, a_m, where $m = 3^n$. Thus arbitrary $n \times n$ ternary reversible logic circuit could be expressed just by a permutation in S_m (where $m = 3^n$). As the basic principal, if the corresponding permutation of a ternary logic reversible circuit is an odd (or even) permutation, it could be called the named odd (or even) reversible logic circuit. Multiplying arbitrary two permutations stands for cascading two logic gates. In what follows, there is no distinctions between an arbitrary $n \times n$ reversible logic gate and a permutation which should be in S_m (here $m = 3^n$).

The synthesis algorithm of any $t \times s$ ternary non-reversible logic circuit (STNC) will be described next. Obviously, when dealing with $t \times s$ ternary non-reversible logic circuits, we should use Theorem 1 firstly, and then synthesize it with the SNT Library based on group theory. This algorithm has been implemented in terms of the lemmas in [17].

V. Examples and Discussions

We would like to illustrate our main results with two examples of $t \times s$ non-reversible logic circuits. We also compare the results with other methods at the end.

A. Example 1: Half Adder

First, let's consider a half adder of ternary. a $2-qutrit$ half adder's truth table could be seen in Table I. It is non-reversible. Let x_0, x_1 be the inputs; sum and C_{out}, the outputs; f, the classical non-reversible half adder.

TABLE I.
TRUTH TABLE OF A 2 – *qutrit* HALF ADDER

No.	x_1	x_0	C_{out}	*sum*
1	0	0	0	0
2	0	1	0	1
3	0	2	0	2
4	1	0	0	1
5	1	1	0	2
6	1	2	1	0
7	2	0	0	2
8	2	1	1	0
9	2	2	1	1

TABLE II.
TRUTH TABLE OF THE TERNARY REVERSIBLE FUNCTION f'

	Inputs				*Outputs*		
No.	x_2	x_1	x_0	**No.**	*s*	*carry*	*sum*
1	0	0	0	**1**	0	0	0
2	0	0	1	**2**	0	0	1
3	0	0	2	**3**	0	0	2
4	0	1	0	**11**	1	0	1
5	0	1	1	**12**	1	0	2
6	0	1	2	**4**	0	1	0
7	0	2	0	**21**	2	0	2
8	0	2	1	**13**	1	1	0
9	0	2	2	**5**	0	1	1
10	1	0	0	**10**	1	0	0
11	1	0	1	**6**	0	1	2
12	1	0	2	**9**	0	2	2
13	1	1	0	**14**	1	1	1
14	1	1	1	**8**	0	2	1
15	1	1	2	**15**	1	1	2
16	1	2	0	**16**	1	2	0
17	1	2	1	**17**	1	2	1
18	1	2	2	**18**	1	2	2
19	2	0	0	**19**	2	0	0
20	2	0	1	**20**	2	0	1
21	2	0	2	**27**	2	2	2
22	2	1	0	**22**	2	1	0
23	2	1	1	**23**	2	1	1
24	2	1	2	**24**	2	1	2
25	2	2	0	**25**	2	2	0
26	2	2	1	**26**	2	2	1
27	2	2	2	**7**	0	2	0

There are 3 rows with the same outputs C_{out}: 0 *sum*: 2. This output configuration has maximal number of repeats. So according to Theorem 1, we need to add one constant ancilla input x_2 and one garbage output *s*. Then, f could be transformed to the reversible logic function f'. f' is reversible now, and its truth table is given in Table II. With the library of SNT, we combine the approach of [18] with Alg. 1 to synthesize the function f'.

The permutations of $(7, 21, 27)$, $(4, 11, 6)$, $(8, 13, 14)$ and $(5, 12, 9)$ could be got from Table II. Here, every number stands for the row number. Using the approach of [18], we have to calculate every permutation group separately, then multiply the results in the order of $(7, 21, 27)$, $(4, 11, 6)$, $(8, 13, 14)$ and $(5, 12, 9)$. Because the procedure is the same for each permutation, we will only demonstrate the calculation with permutation $(4, 11, 6)$.

The 4^{th} row stands for (010), 11^{th} for (101) and 6^{th} for (012), respectively. This can be expressed by Eq. 4. This permutation could be decomposed into the product

TABLE III.
THE RELATIONSHIP OF ROW'S NUMBER

			Row No.		Row No. of table II
0	1	2	b_1	→	6
0	1	0	b_2	→	4
1	1	0	b_3		
1	0	0	b_4		
1	0	1	b_5	→	11

of some neighboring $3-cycles$ [18]. From Table III, the result is $(b_1, b_2, b_5) = (b_2, b_4, b_3) * (b_3, b_4, b_5) * (b_1, b_2, b_3)$.

$$\begin{pmatrix} 4 \\ 11 \\ 6 \end{pmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \rightarrow \begin{pmatrix} b_2 \\ b_5 \\ b_1 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_5 \end{pmatrix} \quad (4)$$

Remark 3: The decomposition is not unique. This suggests the synthesis approach is not optimal.

Firstly, from Eq. 5 we get $(b_2, b_4, b_3) = N_1 * N_2 * N_3^{\wedge 2} * C_1^{\wedge 2} * C_2 * C_1^{\wedge 2} * C_2^{\wedge 2} * C_1 * N_3 * N_2^{\wedge 2} * N_1^{\wedge 2}$. The latter part $C_1^{\wedge 2} * C_2^{\wedge 2} * C_1 * N_3 * N_2^{\wedge 2} * N_1^{\wedge 2}$ is the inverse of the former part $N_1 * N_2 * N_3^{\wedge 2} * C_1^{\wedge 2} * C_2$. Similarly, $(b_3, b_4, b_5) = N_1 * N_2^{\wedge 2} * N_3^{\wedge 2} * C_2 * C_3^{\wedge 2} * C_2^{\wedge 2} * C_3 * C_2^{\wedge 2} * N_3 * N_2 * N_1^{\wedge 2}$, $(b_1, b_2, b_3) = N_1^{\wedge 2} * N_2 * N_3^{\wedge 2} * C_3^{\wedge 2} * C_1^{\wedge 2} * C_3 * C_1 * C_3 * N_3 * N_2^{\wedge 2} * N_1$.

$$\begin{pmatrix} b_2 \\ b_4 \\ b_3 \end{pmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \xrightarrow{N_1 N_2 N_3^2} \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 2 \end{bmatrix} \xrightarrow{C_1^2}$$
$$\begin{bmatrix} 0 & 2 & 2 \\ 2 & 1 & 2 \\ 1 & 2 & 2 \end{bmatrix} \xrightarrow{C_2} \begin{bmatrix} 0 & 2 & 2 \\ 2 & 2 & 2 \\ 1 & 2 & 2 \end{bmatrix} \xrightarrow{C_1^2} \begin{bmatrix} 2 & 2 & 2 \\ 1 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix} \quad (5)$$

Following the same method, we can compute the synthesis result of $(5, 12, 9)$, $(7, 21, 27)$ and $(8, 13, 14)$. Finally:

$f' = N_1 * N_2 * N_3^{\wedge 2} * C_1^{\wedge 2} * C_2 * C_1^{\wedge 2} * C_2^{\wedge 2} * C_1 * N_3 * N_2^{\wedge 2} * N_1^{\wedge 2} * N_1 * N_2^{\wedge 2} * N_3^{\wedge 2} * C_2 * C_3^{\wedge 2} * C_2^{\wedge 2} * C_3 * C_2^{\wedge 2} * N_3 * N_2 * N_1^{\wedge 2} * N_1^{\wedge 2} * N_2 * N_3^{\wedge 2} * C_3^{\wedge 2} * C_1^{\wedge 2} * C_3 * C_1 * C_3 * N_3 * N_2^{\wedge 2} * N_1 * N_1^{\wedge 2} * N_2 * C_2 * C_3 * C_2 * C_3^{\wedge 2} * C_2^{\wedge 2} * N_2^{\wedge 2} * N_1 * N_1^{\wedge 2} * N_2^{\wedge 2} * C_1^{\wedge 2} * C_2^{\wedge 2} * C_1 * N_2 * N_1 * N_3^{\wedge 2} * C_1 * C_3 * C_1^{\wedge 2} * C_3^{\wedge 2} * C_1^{\wedge 2} * N_3 * C_2 * C_3^{\wedge 2} * C_2^{\wedge 2} * C_3 * C_2^{\wedge 2} * N_1 * N_3 * C_1^{\wedge 2} * C_2 * C_1 * C_2^{\wedge 2} * C_1 * N_3^{\wedge 2} * N_1^{\wedge 2} * N_1 * N_2 * N_3 * C_2 * C_3 * C_2 * C_3^{\wedge 2} * C_2^{\wedge 2} * N_3^{\wedge 2} * N_2^{\wedge 2} * N_1^{\wedge 2}$

Here we use the N_i to denote ternary Not gate, i stands for the i^{th} column which from left to right, $N_i^{\wedge 2}$ represents two cascaded Not gates. Meanwhile, C_i means thatin the ternary Toffoli gate, the i^{th} line will be the controlled under the order from left to right. The half adder verifies our approach directly.

Remark 4: When synthesizing the ternary circuit, it wasn't considered the most optimal case in Algorithm 1 and example 1. It was know that the non-reversible ternary logic circuit could be transferred into different gate library's products. Even if the most optimal reversible functio could be gotten, the non-reversible circuit could not be implemented optimally.

Meanwhile, the quantum implementation costs of various quantum logic gates are different, such as, the CNOT gate costs less than Toffoli gate. Hung et al. presented a methods of optimally synthesizing the circuits with

TABLE IV.
TRUTH TABLE OF A 2 – *qutrit* FULL ADDER

No.	x_1	x_0	C_{in}	C_{out}	*sum*
1	0	0	0	0	0
2	0	0	1	0	1
3	0	0	2	0	2
4	0	1	0	0	1
5	0	1	1	0	2
6	0	1	2	1	0
7	0	2	0	0	2
8	0	2	1	1	0
9	0	2	2	1	1
10	1	0	0	0	1
11	1	0	1	0	2
12	1	0	2	1	0
13	1	1	0	0	2
14	1	1	1	1	0
15	1	1	2	1	1
16	1	2	0	1	0
17	1	2	1	1	1
18	1	2	2	1	2
19	2	0	0	0	2
20	2	0	1	1	0
21	2	0	2	1	1
22	2	1	0	1	0
23	2	1	1	1	1
24	2	1	2	1	2
25	2	2	0	1	1
26	2	2	1	1	2
27	2	2	2	2	0

minimize cost, which also applied on the half adder and full adder circuits [4]. The comparison between our method and the method of Hung et al. will be given in subsection C comparison of results and discussions.

B. Example 2: Full Adder

As another example, we consider the ternary full adder. The truth table of it is shown in Table IV. C_{in} denotes the carrier of the full adder. Actually its value is either 0 or 1, but never 2. Here, we suppose the ideal situation where the outputs can be 0, 1 and 2. In Table IV, we can see that the maximum number of repeated output configuration is '01', which occurs seven times. According to Theorem 1, we have to add one ancilla input and two garbage outputs. Then, the non-reversible function of the full adder f could be transformed to the reversible logic function f''. As a result of its four inputs, there are 81 rows in the truth table of f''. Due to the length of the transformed truth table, we won't show it here.

We use the same method to synthesize f'' with the SNT library. Because there are more than 150 gates to be used, the result is not shown here. This example only illustrates the correctness of our main results, it does not consider the optimization.

C. Comparison of Results and Discussions

Our examples verify Theorem 1 and Corollary 1. Compared with other methods, its advantage is that the number of ancilla input bits and garbage outputs are minimal. This can be seen from Table V directly. In [11], for the binary full adder case, three different gates are used for synthesis. Two ancilla input bits and three garbage outputs are added, whereas our method uses one ancilla input bit and two garbage outputs. Meanwhile, [11] defined a NG gate. Even if this increases the cost of the circuit, it has lower cost than our method because we have to use more than 150 gates for a full adder. Furthermore, we make a more intuitive comparison between the ternary cases. S. Mandal et al. [23] had proposed a method of expressing the logic function of ternary with the operations of L_i and J_i. And the definition of a minterm was also presented. With this method, the synthesis of ternary half-adder and ternary full-adder logic circuits have fewer gates but more ancilla bits and garbage bits. Our examples show that our method uses 50% fewer bits for a half adder and 66.7% fewer bits for a full adder in comparison with [23]. However, the numbers of gates used are 30.5 times and 20+ times of those used in [23] in the half and full adder cases, respectively.

TABLE V.
THE COMPARISON BETWEEN DIFFERENT APPROACHES OF SYNTHESIS HALF AND FULL ADDER

Circuits	Gate Library	Ancilla Input Bits	Garbage Outputs	Gates Number Used in Synthesis Circuits
Binary Full Adder in [11]	Feynman Gate Toffoli Gate New Gate	2	3	3
Ternary Half Adder in [23]	Ternary Feynman Gate Ternary C^2NOT Gate Ternary Toffoli Gate	2	2	4
Ternary Half Adder of Our	Ternary NOT Gate Ternary Toffoli Gate	1	1	122
Ternary Full Adder in [23]	Ternary Feynman Gate Ternary C^2NOT Gate Ternary Toffoli Gate Generalized Ternary Gate	4	5	8
Ternary Full Adder of Our	Ternary Swap Gate Ternary NOT Gate Ternary Toffoli Gate	1	2	> 150

In this paper, the optimal quantum cost of these examples is not considered. There is a shallow analysis why it is not considered. J. Smolin et al. [4] introduced a method of cost evaluation for quantum gates . Commonly, every 2-qubit logic gates has a cost of 1 for quantum implementation. Similarly, if the quantum XOR gate or quantum C-V gate are worked on the two qubits for a symmetric pattern, it is considered the total cost as 1 also. In [3] it gives two examples of quantum half-adder circuit of cost 4, and quantum full-adder of cost 6, respectively. However, our method focuses on the minimal number of ancilla input bits and garbage outputs. Hence, it is unnecessary to take account into the quantum cost of these examples. As a result, there is a trade off among the number of gates, ancilla bits and garbage outputs. Our future work will investigate more optimization of the synthesis algorithm for ternary logic circuits.

VI. CONCLUSIONS

In summary, we demonstrate a universal method for synthesizing arbitrary classical ternary non-reversible circuits. This work takes into account the combination of classical logic and quantum logic circuits. With the library

of ternary Toffoli gate, ternary NOT gate and ternary Swap gate, out method presents how to transform these non-reversible circuits. Combining with their group theory notations, we can synthesize these ternary non-reversible logic circuits by adding $s+\lceil log_3 r\rceil - t$ input with constant values and garbage output of $\lceil log_3 r\rceil$. According to the main result, it could be seen that the quantum ternary logic circuit is various of the quantum binary reversible circuit. Simultaneously, the qutrits we use are minimal compared to other methods [11], [23]. The SNT library is no the unique one to synthesize the ternary quantum logic circuit. As show in [18], the library of ternary of $1-CNOT$ gate, ternary NOT gate and ternary Swap gate has more properties and characteristics to transform the ternary circuits, which is more universal to realize arbitrary $n-qutrit$ logic circuits with non ancilla qutrits. Furthermore, out approach of realizing arbitrary ternary logic circuits is constructive. Despite is is not the most optimal case. It could be used further to develop methods of synthesizing $d-level$ circuits with minimal qutrits.

Acknowledgments

The authors also would like to express their sincere thanks to Dr. Hsien-Hang Shieh for useful discussions, Rahul and Dr. Kang L. Wang for their help of the structure of this paper.

References

[1] R. Korf, *Artificial intelligence search algorithms*. Chapman & Hall/CRC, 2010.

[2] A. De Vos, B. Raa, and L. Storme, "Generating the group of reversible logic gates," *Journal of Physics A: Mathematical and General*, vol. 35, no. 33, p. 7063, 2002.

[3] V. Shende, A. Prasad, I. Markov, and J. Hayes, "Reversible logic circuit synthesis," in *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*. ACM, 2002, pp. 353–360.

[4] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 9, pp. 1652–1663, 2006.

[5] R. Feynman, "Quantum mechanical computers," *Foundations of physics*, vol. 16, no. 6, pp. 507–531, 1986.

[6] M. Nielsen and I. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[7] T. Toffoli, "Reversible computing," *Automata, Languages and Programming*, pp. 632–644, 1980.

[8] C. Bennett, "Notes on the history of reversible computation," *IBM Journal of Research and Development*, vol. 32, no. 1, pp. 16–23, 1988.

[9] ——, "Notes on landauer's principle, reversible computation, and maxwell's demon," *Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics*, vol. 34, no. 3, pp. 501–510, 2003.

[10] R. Keyes, "The impact of moore's law," *Solid-State Circuits Newsletter, IEEE*, vol. 11, no. 5, pp. 25–27, 2006.

[11] H. Babu, M. Islam, S. Chowdhury, and A. Chowdhury, "Synthesis of full-adder circuit using reversible logic," in *VLSI Design, 2004. Proceedings. 17th International Conference on*. IEEE, 2004, pp. 757–760.

[12] Z. Zilic and K. Radecka, "Scaling and better approximating quantum fourier transform by higher radices," *Computers, IEEE Transactions on*, vol. 56, no. 2, pp. 202–207, 2007.

[13] A. Muthukrishnan and C. Stroud Jr, "Multivalued logic gates for quantum computation," *Physical Review A*, vol. 62, no. 5, p. 052309, 2000.

[14] D. Miller, D. Maslov, and G. Dueck, "Synthesis of quantum multiple-valued circuits," *JOURNAL OF MULTIPLE VALUED LOGIC AND SOFT COMPUTING*, vol. 12, no. 5/6, p. 431, 2006.

[15] P. Kerntopf, M. Perkowski, and M. Khan, "On universality of general reversible multiple-valued logic gates," in *Multiple-Valued Logic, 2004. Proceedings. 34th International Symposium on*. IEEE, 2004, pp. 68–73.

[16] G. Yang, X. Song, W. Hung, and M. Perkowski, "Bidirectional synthesis of 4-bit reversible circuits," *The Computer Journal*, vol. 51, no. 2, pp. 207–215, 2008.

[17] G. Yang, X. Song, M. Perkowski, and J. Wu, "Realizing ternary quantum switching networks without ancilla bits," *Journal of Physics A: Mathematical and General*, vol. 38, no. 44, p. 9689, 2005.

[18] G. Yang, F. Xie, X. Song, and M. Perkowski, "Universality of 2-qudit ternary reversible gates," *Journal of Physics A: Mathematical and General*, vol. 39, no. 24, p. 7763, 2006.

[19] J. Dixon and B. Mortimer, *Permutation groups*. Springer Verlag, 1996, vol. 163.

[20] L. Storme, A. Vos, and G. Jacobs, "Group theoretical aspects of reversible logic gates," *Journal of Universal Computer Science*, vol. 5, no. 5, pp. 307–321, 1999.

[21] G. Yang, W. Hung, X. Song, and M. Perkowski, "Majority-based reversible logic gates," *Theoretical computer science*, vol. 334, no. 1, pp. 259–274, 2005.

[22] X. Song, G. Yang, M. Perkowski, and Y. Wang, "Algebraic characterization of reversible logic gates," *Theory of Computing Systems*, vol. 39, no. 2, pp. 311–319, 2006.

[23] S. Mandal, A. Chakrabarti, and S. Sur-Kolay, "Synthesis techniques for ternary quantum logic," in *Multiple-Valued Logic (ISMVL), 2011 41st IEEE International Symposium on*. IEEE, 2011, pp. 218–223.

[24] X. Li, G. Yang, and D. Zheng, "Synthesis of ternary non-reversible logic circuits," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010, pp. 1–4.

[25] M. Kargapolov and J. Merzljakov, *Fundamentals of the Theory of Groups*. Springer-Verlag New York, 1979.

Xiaoyu Li received her M.S. degree in computer application technology. She has been pursuing the PhD. degree with University of Electronic Science and Technology of China, Chengdu, 611731, since 2009. She has been the visiting scholar of University of California at Los Angles, since 2011. Her current research interests include quantum computation and quantum information.

Guowu Yang received his PhD. degree in Dept. of electrical and computer engineering, Portland State University, USA, 2005. Now he is the Professor in University of Electronic Science and Technology of China. His current research interests include hardware formal verification, floor planing, reversible logic.

Desheng Zheng received his M.S. degree in computer application technology. He has been pursuing the PhD. degree with University of Electronic Science and Technology of China, Chengdu, 611731, since 2009. His current research interests include formal methods, model checking, biological computation and its application on biological systems.