2004

Exact Synthesis of 3-qubit Quantum Circuits from Non-binary Quantum Gates Using Multiple-Valued Logic and Group Theory

Guowu Yang
Portland State University

William N. N. Hung
Portland State University

Xiaoyu Song
Portland State University

Marek Perkowski
Portland State University

Citation Details

Guowu Yang, William Hung, Xiaoyu Song, and Marek Perkowski, "Exact synthesis of 3-qubit quantum circuits from non-binary quantum gates using multiple-valued logic and group theory," submitted to International Journal of Electronics, 2004.

Exact Synthesis of 3-qubit Quantum Circuits from Non-binary Quantum Gates Using Multiple-Valued Logic and Group Theory

Guowu Yang, William N. N. Hung, Xiaoyu Song and Marek A. Perkowski

Dept. ECE, Portland State University, Oregon, USA.

ABSTRACT

We propose an approach to optimally synthesize quantum circuits from non-permutative quantum gates such as Controlled-Square-Root–of-Not (i.e. Controlled-V). Our approach reduces the synthesis problem to multiple-valued optimization and uses group theory. We devise a novel technique that transforms the quantum logic synthesis problem from a multi-valued constrained optimization problem to a permutable representation. The transformation enables us to utilize group theory to exploit the symmetric properties of the synthesis problem. Assuming a cost of one for each two-qubit gate, we found all reversible circuits with quantum costs of 4, 5, 6, etc, and give another algorithm to realize these reversible circuits with quantum gates. The approach can be used for both binary permutative deterministic circuits and probabilistic circuits such as controlled random number generators and hidden Markov models.

Index Terms: Reversible Logic, Boolean Functions, Logic Synthesis, Group Theory.

1. Introduction

Reversible logic plays an important role in the synthesis of quantum computing circuits. The synthesis of reversible logic circuits using elementary quantum gates is different from classical (non-reversible) logic synthesis. In this paper, we propose a novel approach to optimally synthesize quantum circuits by group theory where the primary inputs are purely binary (outputs are not necessarily binary, they may be random binary after measurement of mixed or superpositioned states). There are some works [5,6,8,10] on reversible logic synthesis using permutative reversible gates (Toffoli, Fredkin, or Feynman gates). However, these gates have different quantum costs (e.g. the cost of

Feynman is lower than that of Toffoli). Therefore, finding the smallest number of gates to synthesize a reversible circuit does not necessarily result in a quantum implementation with the lowest cost (in terms of quantum gates). The exact minimal costs for NMR [1] realization of several quantum gates from truly quantum (not permutative) gates such as Pauli Rotations or Controlled-Square-Root-of-Not have been calculated [4]. They can be also calculated for other quantum technologies. In this paper, we focus on synthesizing reversible circuits to quantum implementations with the lowest cost. The method is general and enumerative. It can be adapted to any particular numerical values of costs. These circuits include common reversible gates that can be used at higher levels of logic synthesis or for technology mapping. We reduce the problem of minimum-cost synthesis of quantum circuits from certain subset of quantum gates and binary inputs (in particular, the permutative circuits) to multiple-valued/binary logic formulation. This approach reduces the search space and the search algorithm complexity. We formulate the quantum logic synthesis problem via group theory. Our method guarantees to find the minimum quantum-cost implementation with truly quantum gates (given a set of specified component gates). In contrast to previous works, which either use permutative reversible gates to design permutative circuits or universal quantum gates to design quantum circuits, we use a subset of quantum gates to design permutative binary circuits that can be either deterministic (when output symbols are restricted to pure states) or probabilistic (when there is no such constraint imposed on the output symbols). The paper is self-contained and no knowledge of quantum mechanics or group theory is necessary to understand it.

2. Background

In quantum computing [1], the fundamental information unit is a qubit. The state of a qubit is a superposition of 0 and 1 states, also denoted as $|0\rangle$ and $|1\rangle$ respectively. The qubit state q can be represented by the equation: $q = \alpha|0\rangle + \beta|1\rangle$, where α and β are both complex numbers and $|\alpha|^2+|\beta|^2=1$. $|\alpha|^2$ and $|\beta|^2$ are probabilities of measurements of pure states 0 and 1, respectively. Observe thus that if the complex numbers are $1/\sqrt{2}$ then pure states 0 and 1 are generated with equal probabilities of ½. This way, synthesis of probabilistic logic and finite state machines in quantum is as complicated as for combinational circuits, which is different from standard circuits. Moreover, machines with entangled states [1] can also be synthesized.

The classical state of 0 corresponds to the case where α=1 and β=0. Similarly, the classical state of 1 corresponds to α=0 and β=1. The effect of quantum gates on a qubit can be described as vector operations on α and β, where the quantum gates are represented by unitary matrices. A unitary matrix is a n×n complex matrix U with the following property:$U \times U^+ = U^+ \times U = I$, where I is the identity matrix and U^+ is the conjugate transpose (also known as the Hermitian adjoint) of U.

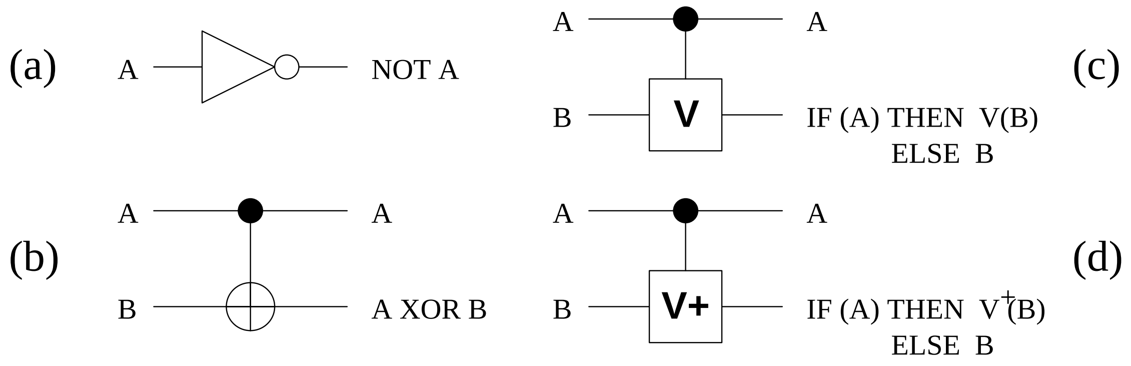


Figure 1: Elementary Quantum Gates

It has been shown that any quantum logic circuit can be constructed using elementary quantum XOR (Feynman gate), and one-qubit gates. It has also been shown [1,15] that arbitrary permutative quantum circuit can be synthesized using only two-qubit gates: controlled-V, controlled-V^+, Feynman gates and single-qubit NOT gates, as shown in Fig. 1. Thus, an entire circuit can be build from 2-qubit gates of finer granularity than Toffoli gates used in the previous research [3,5,6,16,17]. The NOT gates are also known as inverters. The quantum XOR gates are also called Feynman or controlled-NOT (CNOT) gates. The controlled-V gate has two bits: control and data. The data output is the same as the data input (B) when the control input (A) value is 0 (FALSE). If the control value is 1 (TRUE), the data output becomes V(input). Similar rules apply to the controlled-V^+ gate, except that its data output becomes V^+(input), where V^+ is the Hermitian of V.

$$V = \begin{pmatrix} 0.5+0.5i & 0.5-0.5i \\ 0.5-0.5i & 0.5+0.5i \end{pmatrix}, \; V^+ = \begin{pmatrix} 0.5-0.5i & 0.5+0.5i \\ 0.5+0.5i & 0.5-0.5i \end{pmatrix}$$

According to [1,15], the values of V and V^+ are constructed such that they are the square root of NOT (i.e. inverter) gate. Hence, if the signal V(input) is passed through another controlled-V gate with its control value also equal to 1 (TRUE), the output of the second gate becomes the NOT of the input.

$V\times V = V^+\times V^+ = \text{NOT}$, $V^+\times V = V\times V^+ = I$.

The XOR, controlled-V and controlled- V^+ gates are 2×2 gates, also called 2-qubit gates. Similarly, the NOT gate is a 1-qubit gate. For quantum implementation the cost of 2-qubit gates far exceeds the cost of 1-qubit gates. Hence, in a first approximation the quantum cost of 1-qubit gates is usually ignored in the presence of 2-qubit implementations [8]. In this paper, for simplification, we consider each of the 2-qubit gates (XOR, controlled-V, controlled-V^+) to have a quantum cost of 1. All our methods can be however easily modified to take into account the precise NMR [1] costs from [4].

Given a reversible circuit, the quantum logic circuit synthesis problem is to synthesize the circuit using the above elementary quantum logic gates with the minimum cost. Various heuristic methods have been applied to find low cost quantum implementations (using the elementary gates) for the functionality of the Fredkin, Toffoli, and Peres gates [1,3,5,6,8,11,16,17]. We solve the quantum logic circuit synthesis problem using group theoretical permutation representation and related algebraic approaches [6–14,16].

We are interested in synthesizing quantum circuit with pure binary inputs and (sometimes) outputs (1 and 0). The values of the signals in each "quantum wire" are modified after passing through the elementary gates. There are six possible output values when we apply binary (1 and 0) inputs to one of those elementary gates: 0, 1, V_0, V_1, V^+_0, V^+_1, where V_0 represents V(input) when input is 0, and similarly for V_1, V^+_0, V^+_1.

$$V_0 = \begin{pmatrix} 0.5+0.5i & 0.5-0.5i \\ 0.5-0.5i & 0.5+0.5i \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.5+0.5i \\ 0.5-0.5i \end{pmatrix},$$

$$V_1 = \begin{pmatrix} 0.5+0.5i & 0.5-0.5i \\ 0.5-0.5i & 0.5+0.5i \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.5-0.5i \\ 0.5+0.5i \end{pmatrix},$$

$$V_0^+ = \begin{pmatrix} 0.5-0.5i & 0.5+0.5i \\ 0.5+0.5i & 0.5-0.5i \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.5-0.5i \\ 0.5+0.5i \end{pmatrix},$$

$$V_1^+ = \begin{pmatrix} 0.5-0.5i & 0.5+0.5i \\ 0.5+0.5i & 0.5-0.5i \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.5+0.5i \\ 0.5-0.5i \end{pmatrix}.$$

These six possible values are used as input values to gates in subsequent stages. We want to synthesize our circuit such that the inputs of XOR and NOT gates and the "control" input of the controlled-V and controlled-V+ will always be pure binary (0's and 1's), i.e., their input values cannot be V_0 etc. Thus we assume that every wire in a circuit has either pure or mixed values. This separates the quantum wires to two groups: pure and mixed, which is used in our group representation as explained below. Given the above six possible values at the data input of the controlled-V or controlled-V+, their corresponding data output has the same set of six possible values. Hence the input/output of every quantum gate in the circuit can be represented using the above six values. If we look at the complex matrix representation of V_0, V_1, $V^+{}_0$, $V^+{}_1$, we can deduce that $V_0=V^+{}_1$, and $V_1=V^+{}_0$. Thus, it suffices to represent signals in the circuit using four values: 0, 1, V_0, V_1. In this way, the problem of quantum circuit synthesis (that would normally use unitary matrices and Hilbert space to represent signals) is reduced to a simpler synthesis problem in mixed binary/quarternary algebra. (It is so in this particular case, but the method is more general than that and may use any kind of MV algebra).

3. Formulation

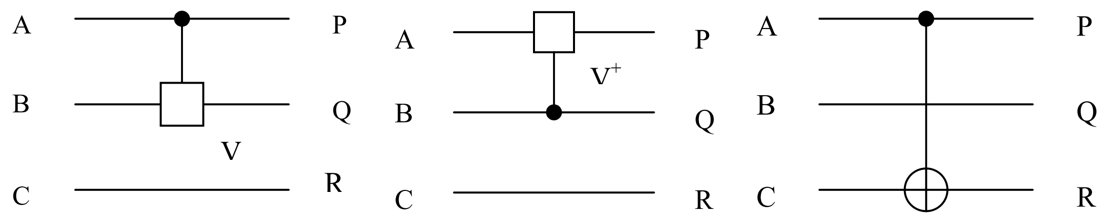
In this section, we will translate the problem realizing a reversible circuit with quantum gates into group theory. First we introduce some basic concepts of permutation group [11-14]. Let $M = \{1, 2,\ldots, n\}$. A bijection (one-to-one mapping) of M onto itself is called a permutation on *M*. We write a permutation as a product of disjoint cycles [11-14]. The identity mapping ( ) is called the unity element in a permutation group. As convention, a product a*b of two permutations *a* and *b* means applying mapping *a* before *b*. Inverse of a: if b*a=a*b=( ), then b is called an inverse of a, denoted as a^{-1}. The image of $s \in M$ under a is denoted as a(s). The image of a subset $S \subseteq M$ is $a(S)=\{a(s) | s \in S\}$.

Consider the truth table of a controlled-V gate on two qubits, as shown in Table 1. We use A and B to denote the control and data inputs, and use P and Q to denote the control and data output respectively.

Table 1: Truth table of Ctrl-V gate

Input			Output		
Label	A	B	P	Q	Label
1	0	0	0	0	1
2	0	1	0	1	2
3	1	0	1	V_0	7
4	1	1	1	V_1	8
5	0	V_0	0	V_0	5
6	0	V_1	0	V_1	6
7	1	V_0	1	1	4
8	1	V_1	1	0	3
9	V_0	0	V_0	0	9
10	V_0	1	V_0	1	10
11	V_1	0	V_1	0	11
12	V_1	1	V_1	1	12
13	V_0	V_0	V_0	V_0	13
14	V_0	V_1	V_0	V_1	14
15	V_1	V_0	V_1	V_0	15
16	V_1	V_1	V_1	V_1	16

The table is enumerated such that pure binary inputs patterns appear at the top of the truth table. The output patterns are computed according to Section 2. The cases where the control input value is non-binary (V_0 or V_1) do not exist. We can thus treat them as don't care cases, because we want to constrain our synthesis such that control inputs are pure binary (as described in Section 2). We specify the output patterns of these don't care cases to be the same as their input patterns. We label the input patterns using natural numbers. The output patterns form a permutation of the input patterns; and the output label reflects that permutation. We can then represent the entire truth table of this gate using the permutation representation: (3, 7, 4, 8). The permutation representations of other orientations (e.g. upside down position) of this gate or other gates or for more qubits can be derived in a similar fashion. Observe that this way any restricted synthesis problem in quantum circuits can be reduced to a problem in some multiple-valued logic. Our approach is thus only an illustration of a category of synthesis problems that reduce representation from unitary (complex) matrices to multiple-valued algebra which allows us to use finite group theory methods and software [14]. The disadvantage of our approach is however that the sizes of these groups grow very quickly.



(a) V_{BA} (b) V^+_{AB} (c) F_{CA}

Figure 2: Arrangements of 3 qubit gates

Consider 3 qubit quantum gates, with inputs A, B, C and outputs P, Q, R, respectively. We define gates for controlled-V, controlled-V^+ and XOR based on their qubit alignments. For example, we denote V_{BA} as the gate shown in Figure 2(a), where the second subscript letter A is the control bit, and the first subscript letter B is the bit which will change (a.k.a. data bit), i.e., P = A; Q = V(B) if A = 1, else Q = B; R = C. Overall, there are six 3-qubit controlled-V gates: V_{BA}, V_{AB}, V_{CA}, V_{AC}, V_{CB}, V_{BC}. Similarly, we denote V^+_{AB} as the gate shown in Figure 2(b). Again, we have six 3-qubit controlled-V^+ gates: V^+_{AB}, V^+_{BA}, V^+_{CA}, V^+_{AC}, V^+_{CB}, V^+_{BC}. We denote F_{CA} as the Feynman gate shown in Figure 3(c), where the first subscript letter C will be changed according to the second subscript letter A, i.e., P = A; Q = B; R = C⊕A. There are six 3 qubit Feynman gates: F_{CA}, F_{AC}, F_{AB}, F_{BA}, F_{CB}, F_{BC}.

In order to use Group Theory, we need to encode the input values. As shown in Section 2, there are four possible values for each qubit: 0, 1, V_0, and V_1. Then we have: $V_0 = V^+_1$, $V_1 = V^+_0$, $V(V_1) = V^+(V_0) = 0$, and $V(V_0) = V^+(V_1) = 1$. Every 3 qubit circuit has $4^3 = 64$ possible input patterns. So we can create a 64 entry truth table for 3 qubit gate similar to the 16 entry truth table for 2 qubit gate in Table 1. But for reversible circuits, we only care about binary input and binary output for each qubit. Considering the property of our elementary quantum gates, every pattern must contain a 1. Otherwise, this pattern will not change after any quantum gate. So we only need to consider 64-27+1 = 38 patterns (including a zero pattern) in our truth table. We can effectively place the remaining 26 (unchangeable) patterns at the bottom of the truth table. So their labels will not permute, and will not show up in our permutation representations. Hence the domain of the permutation representation is reduced from 64 to 38. We arrange these permutable patterns such that the 8 binary patterns will appear first (from small to big), then the other 30 patterns (containing V_0 or V_1) also appear from small to big. Again, in order to make the output patterns a permutation of input patterns, we define that when the control bit is equal to V_0 or V_1, the data bit will keep its value unchanged. The following formulae demonstrate the permutations of the quantum gates.

$V_{BA} = (5,17,7,21)(6,18,8,22)(13,19,15,23)(14,20,16,24)$,

$V^{+}_{AB} = (3,33,7,26)(4,34,8,27)(9,35,15,28)(10,36,16,29)$,

$Fe_{CA} = (5,6)(7,8)(17,18)(21,22)$.

Let S = {1,2,3,4,5,6,7,8} be the set of the index of the binary input patterns that we consider. In the product of multiplying a permutation, we give a banned set for this permutation such that using the corresponding quantum gate is reasonable. A banned set N_A is the set of indices in which the value of the qubit A is V_0 or V_1, i.e., $N_A=\{25,26,27,28,29,30,31,32,33,34,35,36,37,38\}$. Given a quantum circuit *f* (a cascading circuit of some quantum gates), we need to know which quantum gate can be cascaded after *f*. To do this, we let the image of a set of input patterns S in *f* be *f*(S), i.e. *f*(S) is the set of output patterns from *f* that corresponds to the set of input patterns. If the intersection $f(S)\cap N_A$ of *f*(S) and N_A is an empty set $\varnothing$, then $L_A=\{V_{BA}, V_{CA}, V^{+}_{BA}, V^{+}_{CA}\}$ can be cascaded after *f*. The reason is that for any *k* in N_A, the bit A in the k^{th} pattern has value V_0 or V_1, which can not be used as control signal for the next control quantum gate.

Similarly, we have other banned sets:

$N_B = \{11,12,17,18,19,20,21,22,23,24,30,31,37,38\}$, used for $L_B = \{V_{AB}, V_{CB}, V^{+}_{AB}, V^{+}_{CB}\}$,

$N_C = \{9,10,13,14,15,16,19,20,23,24,28,29,35,36\}$, used for $L_C = \{V_{AC}, V_{BC}, V^{+}_{AC}, V^{+}_{BC}\}$.

The same concept applies to Feynman gates. We define N_{AB} is the set of index in which the value of the qubit A or B is V_0 or V_1, i.e., $N_{AB}=\{11,12,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38\}$ used for $L_{AB}=\{ F_{AB}, F_{BA}\}$. If $f(S) \cap N_{AB} = \varnothing$ (empty set), then F_{AB} and F_{BA} can be cascaded after f.

Similarly, we have:

$N_{AC}=\{9,10,13,14,15,16,19,20,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38\}$ used for $L_{AC}=\{ F_{AC}, F_{CA}\}$;

$N_{BC}=\{9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,28,29,30,31,35,36,37,38\}$ used for $L_{BC}=\{ F_{BC}, F_{CB}\}$.

Definition 1. The Reasonable Product $g_1•g_2$ of two permutations g_1 and g_2 on $\{1,2,…,38\}$, where g_2 belongs to the quantum library L (the union of L_A, L_B, L_C, L_{AB}, L_{AB}, L_{BC}), is defined as normal permutation product g_1*g_2 if only if at least one of the following six cases can happen:

i): if $g_2 \in L_A$, then $g_1(S) \cap N_A=\varnothing$;

ii): if $g_2 \in L_B$, then $g_1(S) \cap N_B=\varnothing$;

iii): if $g_2 \in L_C$, then $g_1(S) \cap N_C=\varnothing$;

iv): if $g_2 \in L_{AB}$, then $g_1(S) \cap N_{AB}=\varnothing$;

v): if $g_2 \in L_{AB}$, then $g_1(S) \cap N_{AC}=\varnothing$;

vi): if $g_2 \in L_{BC}$, then $g_1(S) \cap N_{BC}=\varnothing$.

We also define the reasonable product of a permutation g and a library L:

$$g \bullet L = \{g \bullet g' \mid g' \in L\}$$

The meaning of reasonable product of $g_1•g_2$ is that: if g_2 is a control gate, then the value of the control bit is either 0 or 1, (i.e., cannot be V_0 or V_1), so the cascading g_2 after g_1 is reasonable (i.e., satisfies our constraint for multi-valued logic); if g_2 is a XOR gate, then the values of the two bits in the XOR function are either 0 or 1, thus the cascading g_2 after g_1 is reasonable. On the other hand, if g_2 can be cascaded after g_1, then the product

g_1*g_2 is a reasonable product $g_1 \bullet g_2$. Therefore, $g \bullet L$ includes all cases cascading a quantum gate after g.

We now present an algorithm to compute the reversible circuit set G[k] of all reversible circuits which have cost k. The constant cb is the upper-bound cost that we can apply in a particular computer (due to finite memory size). In our computer, cb=7. In our algorithm, we use a useful function Restrictedperm(b,S) in GAP [14] (a group theory computation package), where b is a permutation on a set M, and S is a subset of M. The function is defined as: If the image b(S)=S, then Restrictedperm(b,S) is a permutation b' on S such that for all s in S, the image b'(s)=b(s). Otherwise, the function will return FALSE. The set S can be configured to constrain our synthesis result to circuit that produce binary outputs for binary inputs, or quaternary outputs (complex valued) for binary inputs, or even for quaternary input. The idea of our algorithm is to create a set A[k] of all quantum circuits that can be constructed using k or less quantum gates. B[k] is the set of quantum circuits that can be constructed using k (and at least k) quantum gates. We create pre_G[k]={b'| b'=Restrictedperm(b,S), $b \in B[k]$}, where b(S)=S means if the input pattern is pure binary, then its output is also a pure binary pattern. So the circuit b' is a reversible circuit with cost equal to or less than k. At the end of the loop, we create the set G[k] by subtracting G[k-1], …, G[1] from pre_G[k] because when we compute the b'=Restrictedperm(b, S) circuit, b' may potentially be a member of any G[j], j<k.

Finding_Minimum_Cost_Circuits Algorithm (FMCF):

Input:

$N_A, N_B, N_C, N_{AB}, N_{AC}, N_{BC}$;

$L_A:=\{V_{BA}, V_{CA}, V^+_{BA}, V^+_{CA}\}$; $L_B:=\{V_{AB}, V_{CB}, V^+_{AB}, V^+_{CB}\}$; $L_C:=\{V_{AC}, V_{BC}, V^+_{AC}, V^+_{BC}\}$; $L_{AB}:=\{F_{AB}, F_{BA}\}$; $L_{AC}:=\{F_{CA}, F_{AC}\}$; $L_{BC}:=\{F_{CB}, F_{BC}\}$.

Output: G[1], G[2], …, G[cb].

```
A[0]:= {( )};
B[0]:= {( )};
For 1≤ k≤ cb, do
    A[k]:=A[k-1];
    For all b in B[k-1], do
```

```
            Add b•L in A[k];
    End for;
    B[k]:= A[k]-A[k-1];
    For all b in B[k], do
        Compute b(S);
        If b(S) = S, then
            b':= Restrictedperm(b,S);
            Add b' in pre_G[k];
        End if;
    End for;
    G[k]:= pre_G[k]-G[k-1]-G[k-2]-…-G[1];
End for;
```

Theorem 1. G[k] is the set of all reversible circuits that fulfill our input/output constraints and have quantum cost of at least k without using NOT gate.

Proof:

i): Consider any reversible circuit g (that satisfies our input/output constraints) which has minimal quantum cost k without NOT gate, there are k control gates or Feynman gates a_1, a_2, …, a_k such that $g=a_1*a_2*…*a_k$. From the algorithm, the general permutation Gen(g) of g is in A[k]. So

g = Restrictedperm(Gen(g),S), and g is in pre_G[k]. If g is not in G[k], then there is $r<k$ such that g is in G[r], therefore there are r quantum gates b_1, b_2, …, b_r such that $g=b_1*b_2*…*b_r$. This contradicts with the minimal quantum cost k. Thus g is in G[k].

ii): On the other hand, for any g in G[k], g has minimal quantum cost k. The reason is: According to the algorithm, there exists b in A[k] such that g = Restrictedperm(b,S). So, there are k quantum gates a_1, a_2, …, a_k such that $g=a_1*a_2*…*a_k$. So g has a realization with quantum cost k. If the cost k is not minimal, then there exists a number $r<k$ such that $g\in G[r]$, which contradicts with g being in G[k]:= pre_G[k]-G[k-1]-G[k-2]-…-G[1]. ■

For arbitrary n-qubit reversible circuits, we use N to denote the group realized by NOT gate. The size of N is 2^n, for all $a\in N$, a*a=( ), and for all $a,b\in N$, a*b=( ) iff a=b.

Let G be the set of all n-qubit reversible circuits realized by control gate and Feynman gate. Let H be the set of all n-qubit reversible circuits realized by control gate, Feynman gate and NOT gate. We can deduce that H can be evenly decomposed to 2^n leftcosets of G without intersection, shown in the following theorem.

Theorem 2: $H= \cup_{a\in N}\ a*G$, and $\forall a,b\in N\ (a\neq b) \Rightarrow (a*G)\cap(b*G)=\emptyset$.

Proof: We encode the input patterns from [0,0,…,0] to [1,1,…,1] as the integers from 1 to 2^n. For any $g\in G$, we have $g(1)=1$. For any $a\in N$, if $a\neq(\ )$, then $a(1)\neq 1$. For any $h\in H$, there exists $a\in N$ such that $(a*h)(1)=1$. So $a*h\in G$. Therefore, $H= \cup_{a\in N}\ a*G$.

Given any $a,b\in N$, $a\neq b$ such that $(a*G)\cap(b*G)\neq\emptyset$, then $\exists g_1,g_2\in G$ such that $a*g_1=b*g_2$. Since $a*a=(\ )$, so $a*b=g_1*(g_2)^{-1}$. We are given $a\neq b$, so $a*b\neq(\ )$, thus $(a*b)(1)\neq 1$.But $(g_1*(g_2)^{-1})(1)=1$. This contradiction shows that the assumption there are a, $b\in N$, $a\neq b$ such that $(a*G)\cap(b*G)\neq\emptyset$ is wrong. Thus for any a, $b\in N$, $a\neq b$, $(a*G)\cap(b*G)=\emptyset$. ■

Specifically, when n=3, H is the symmetry group S_8.

G = Groupgeneratedby$\{F_{AB}, F_{BA}, F_{BC}, F_{CB}, Pe_{AB}\}$

$|G| = 5040$,

$|S_8| = 40320$.

Based on the Finding Algorithm and Theorem 2, we formulate the Expressing Algorithm:

Minimum_Cost_Expressing Algorithm (MCE):

Input:

```
Reversible circuit g;
Reversible gates $N_A$, $N_B$, $N_C$, $N_{AB}$, $N_{AC}$, $N_{BC}$;
$L_A:=\{V_{BA}, V_{CA}, V^+_{BA}, V^+_{CA}\}$; $L_B:=\{V_{AB}, V_{CB}, V^+_{AB}, V^+_{CB}\}$; $L_C:=\{V_{AC}, V_{BC}, V^+_{AC}, V^+_{BC}\}$; $L_{AB}:=\{F_{AB}, F_{BA}\}$; $L_{AC}:=\{F_{CA}, F_{AC}\}$; $L_{BC}:=\{F_{CB}, F_{BC}\}$.
```

Output: flag, t, d[0], d[1], …, d[t];

```
flag:= 0; t:= 0;
If g in $\{(\ )\}\cup$NOT then
    flag:= 1;
```

```
   t:= 0;  d[0]:= g;
End if
If flag = 0 then
    Find d[0] in $\{(\,)\} \cup$ NOT such that  $((d[0])^{-1}*g)(1) = 1$;
      g:= $(d[0])^{-1}*g$;
    For $1\leq k\leq cb$, do
         Compute B[k], G[k] as Finding Algorithm
         If g in G[k] then
             flag:= 1; t:= k;
             For all b in B[k] do
                     If g = Restrictedperm(b,S) then stop for;
             End for
         End if
     End for
     For j from t to 1 do
           Find d[j] in L such that
             b:=b*$(d[j])^{-1}$ in B[j-1] and b*d[j] is reasonable.
      End for
End if
```

Theorem 3: If the quantum cost of g does not exceed the bound cb, then the Expresssing Algorithm will return d[0], d[1], …, d[t] such that g = d[0]*d[1]*…*d[t] and t is minimum.

Proof: Assume g has minimum cost $t \leq cb$. If $g(1)\neq 1$, then there exists a NOT gate implementation $d[0]\in N$ such that $a=(d[0])^{-1}*g$, and $a(1)=1$. Otherwise $g(1)=1$, choose d[0]=( ). Based on Theorem 2, $a\in G$. So, a has the same minimum cost t as g.

If the minimum cost s of a is no more than cb, then a will be in G[s], flag=1. According to the algorithm MCE, there exists $t \leq cb$ such that $a\in G[t]$. According the algorithm FMCF, there are quantum gates d[1], …, d[t] such that a= d[1]*…*d[t], i.e., g = d[0]*d[1]*…*d[t]. If a has lower cost $s<t$, then there are c[1], …, c[s] such that a= c[1]*…*c[s]. Then $a\in$pre_G[s], which implies $a\notin G[t]$, which is a contradiction. Thus, t

is the minimum cost of a or g. If flag=0, then a cannot be in G[1], …, G[cb]. The cost of a must be bigger than cb. ■

4. Synthesis of Quantum Automata

Recently quantum random number generators have been introduced to the market as one of first two practical applications of quantum computing [19]. Although these circuits are simple, they are the first example of a new technology to come and for which no formal synthesis methods exist.

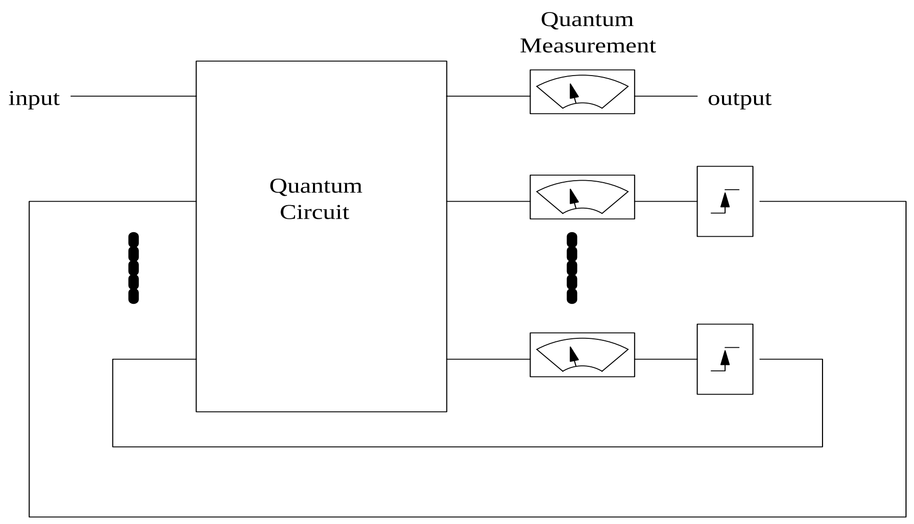


Figure 3: Quantum realized Probabilistic State Machine

Our synthesis approach presented above is also applicable in the construction of quantum-realized probabilistic state machines (quantum automata, etc. [18]) and controlled quantum random number generators [19]. Given a specification, we create a truth table with binary inputs and quaternary outputs. We can apply our approach to synthesize quantum circuit for this specification. With quantum-measurement units this circuit behaves externally as a probabilistic combinational circuit. It has deterministic inputs and probabilistic outputs (which are functions of inputs generated by the quantum gates). So, without any modifications, our approach generates quantum circuits with probabilistic combinational functionality. This logic (quantum plus measurements) can be used as a combinational component of a quantum automaton, just standard memory elements and loop are added – see Figure 3. This automaton is observed externally as a machine with probabilistic and entangled behaviors. The outputs and next states are probabilistically generated binary vectors. This approach will enable us to synthesize minimal quantum automata, Hidden Markov Models and similar concepts, thus extending the application of quantum circuits beyond commercial quantum random number generators.

5. Experiments

We applied our minimum cost algorithm to 3 qubit synthesis, the results are shown in the following table.

Table 2: Number of circuits with cost k

Cost k	0	1	2	3	4	5	6	7
|G[k]|	1	6	30	52	84	156	398	540
|S8[k]|	8	48	240	416	672	1248	3184	4320

For cost up to 3: G[1], G[2] and G[3] consists of the set of the binary input binary output circuits which are the combinations of 1, 2, 3 Feynman gates respectively.

In G[4], there are 60 circuits realized by 4 Feynman gates, the other 24 circuits realized by 3 control gates and 1 Feynman gate. And these 24 circuits exhibit the property of universal gates: all 3-bit binary input and binary output reversible circuits can be realized by NOT gates, Feynman gates and any one of these 24 circuits. To show this, we use g to denote any one of these 24 gates, M=group generated by g, NOT gate and these six Feynman gates. Using GAP, we have the size of M is Size(M)=40320=Size(S_8). And M is a subset of S_8. Therefore M=S_8, namely, S_8 can be realized by g, NOT gate and Feynman gate. There are four representative circuits from these 24 circuits. Each of these four circuits has other five similar circuits with different permutations of the three bits.

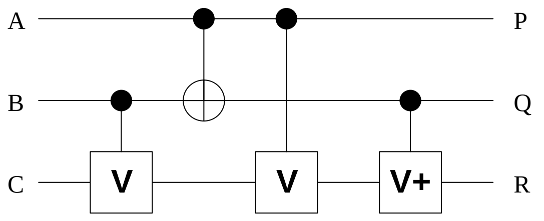


Figure 4: Peres circuit (g_1)

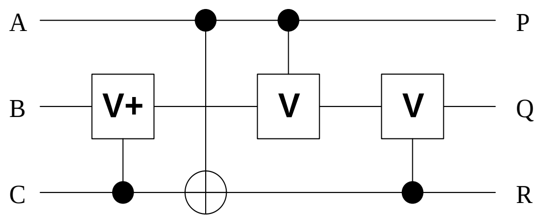


Figure 5: Quantum circuit of g_2

g_1 = (5,7,6,8) = $V_{CB}*F_{BA}*V_{CA}*V^{+}_{CB}$ (Figure 4), P = A, Q = B⊕A, R = C⊕AB; The gate g_1 is a Peres gate.

g_2 = (5,8,7,6) = $V^{+}_{BC}*F_{CA}*V_{BA}*V_{BC}$ (Figure 5), P = A, Q = B⊕AC', R = C⊕A;

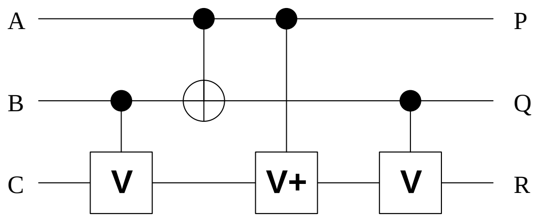


Figure 6: Quantum circuit of g_3

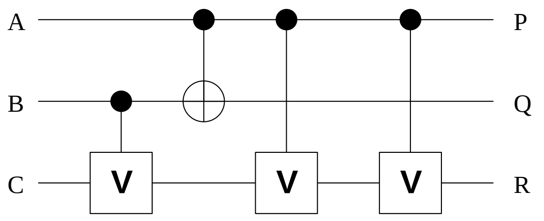


Figure 7: Quantum circuit of g_4

g_3 = (3,4)(5,7)(6,8) = $V_{CB}*F_{BA}*V^{+}_{CA}*V_{CB}$ (Figure 6), P = A, Q = B⊕A, R = C⊕A'B;

g_4 = (3,4)(5,8)(6,7) = $V_{CB}*F_{BA}*V_{CA}*V_{CB}$ (Figure 7), P = A, Q = B⊕A, R = C'⊕A'B'.

Our algorithm can be used to synthesize a quantum circuit from a specified circuit. We applied our algorithm to synthesize the well known Peres and Toffoli circuits. It took 9 CPU seconds (on a 850MHz Pentium® III) to synthesize the Peres circuit (cost=4) and 98 seconds for the Toffoli circuit (cost=5).

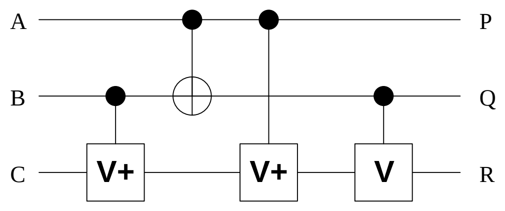
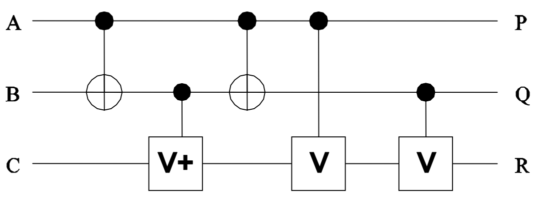
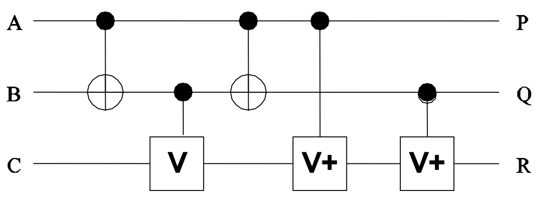


Figure 8: Hermitian adjoint implementation of Peres

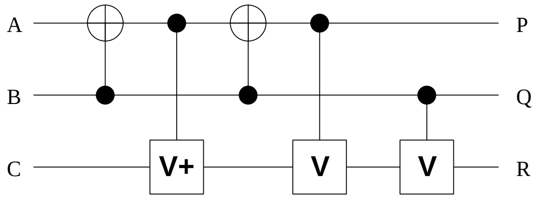
Our synthesis algorithm found two implementations for Peres, one of them is the same as shown in Figure 4. The other one is the hermitian adjoint implementation, which is swapping all control-V and control-V^+ gates, shown in Figure 8.



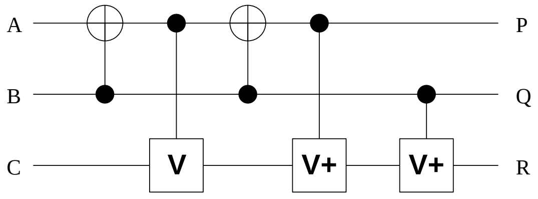
(a) $To=F_{BA}*V^+_{CB}*F_{BA}*V_{CA}*V_{CB}$



(b) $To=F_{BA}*V_{CB}*F_{BA}*V^+_{CA}*V^+_{CB}$



(c) $To= F_{AB}*V^+_{CA}*F_{AB}*V_{CA}*V_{CB}$



(d) $To=F_{AB}*V_{CA}*F_{AB}*V^+_{CA}*V^+_{CB}$

Figure 9: Quantum implementations of Toffoli

For the Toffoli circuit, we found four quantum implementations, shown in Figure 9. Notice that circuits (a) and (b) (in Figure 9) are Hermitian adjoints of each other by simply exchanging V and V^+ gates; and circuits (c) and (d) are also Hermitian adjoints of each other. The difference between these two pairs lies in the qubit where they perform XOR (Feynman) operations. Since Toffoli has two outputs (P and Q) that are both not changing (same as their inputs), we have two choices to perform the XOR operations: qubit A (circuits (a) and (b)) or qubit B (circuits (c) and (d)). Notice that for runtime performance our algorithm does not intend to find all possible implementations of the specified circuit. It only finds some implementations as the synthesis result.

6. Conclusion

In this paper we formulated a method to exactly minimize a subset of quantum circuits by reducing the problem to multiple-valued logic and group theory. As far as we know, this is the first time that such a combined approach has been proposed. Using this method we found many new gates and inexpensive realizations of permutative quantum circuits. For instance, we found a family of Peres-like gates which have all the same lowest cost and can be used to synthesize permutative quantum circuits (section 5). Not only is the Peres gate the cheapest of all NMR realized permutative gates, but we show that there is a large family of such gates with the same smallest possible cost, for which nobody has developed a synthesis method yet. We also demonstrated (in another paper submitted to this conference) that the number of gates using libraries with Peres gates is smaller than using other libraries for all 3-qubit circuits. Taken together, these are strong arguments for Peres-like gates realized using quantum basic gates from this paper.

Our method can also be used to synthesize circuits with pure inputs and mixed outputs. Because the mixed outputs correspond in "quantum measurements" to randomly generated vectors with known probabilities [1], our method is then without any modification a new approach to synthesize a class of binary-input circuits that have random but controlled binary outputs (we remove the constraint that outputs are pure states). In particular, this class includes probabilistic finite state machines and hidden Markov Models [18].

Our future research is on finding efficient heuristics that would allow us to synthesize probabilistic and entangled state machines from examples of their behaviors expressed in multiple-valued logics corresponding to sets of possible complex numbers (each possible complex number encoded by one value of the logic). Such examples correspond to probabilistic input-output behaviors of the machine.

7. References

1. M. A. Nielson and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge Univ. Press, December 2000.
2. W. N. N. Hung et al., "Quantum Logic Synthesis by Symbolic Reachability Analysis", *Proc. ACM/IEEE Design Automation Conference* (DAC), San Diego, CA, June 2004.
3. T. Toffoli. Reversible computing. Tech.Memo MIT/LCS/TM-151, MIT Lab for Comp. Sci, 1980.

4. S. Lee, J.-S. Lee, S.-J. Lee, and T. Kim., "Costs of basic gates in quantum computation", submitted to *Journal of Physics A*, (cited with author's permission from private communication), 2004.
5. V. V. Shende, A.K. Prasad, I.L. Markov, and J.P. Hayes. Reversible logic circuit synthesis. In *Proceedings of ICCAD*, pages 125-132, San Jose, California, USA, Nov 10-14, 2002.
6. A. De Vos, B. Raa and L. Storme. Generating the group of reversible logic gates. *Journal of Physics A: Mathematical and General*, 35, (2002), pages 7063–7078.
7. R. Forf. Artificial intelligence search algorithms. *Algorithms and Theory of Computation Handbook*, CRC Press, 1999.
8. M. Perkowski, M. Lukac, M. Pivtoraiko, P. Kerntopf, M. Folgheraiter. A hierarchical approach to computer aided design of quantum circuits", *6th International Symposium on Representations and Methodology of Future Computing Technology*, pages 201-209, Trier, Germany, March 2003.
9. S. S. Bullock and I. L. Markov. An arbitrary two-qubit computation in 23 elementary gates. *Proceedings of DAC*, pages 324-329 Anaheim, Cal, USA, June 2003.
10. D. M. Miller, D. Maslov and G. W. Dueck. A transformation based algorithm for reversible logic synthesis. *Proceedings of DAC*, page 318-323, Anaheim, Cal, USA, June 2003.
11. L. Storme et al. Group theoretical aspects of reversible logic gates. *Journal of Universal Computer Science*, 5 (1999), pages 307-321.
12. J. D. Dixon, and B. Mortimer. *Permutation Groups*. Springer, New York (1996).
13. M. I. Kargapolov, and J. I. Merzljakov. *Fundamentals of the Theory of Groups*. Springer-Verlag, New York (1979).
14. M. Schonert et.al. *GAP-Group, Algorithms, and Programming*. Lehrstuhl D fur Mathematik, Rheinisch Westfalische Technische Hochschule, Aachen, Germany, fifth, 1995.
15. J. A. Smolin, and D. P. DiVincenzo. Five two-bit quantum gates are sufficient to implement the quantum Fredkin gate. *Physical Review A*, 53 (1996), pages 2855-2856.

16. G. Yang, W. N. N. Hung, X. Song, and M. Perkowski. Majority-based reversible logic gate", 6th *International Symposium on Representations and Methodology of Future Computing Technology* (RM2003), pages 191-200, Trier, Germany, March 2003.
17. V. V. Shende, A. K. Prasad, I. L.Markov, and J. P. Hayes, "Synthesis of Reversible Logic Circuits", *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, Vol. 22, No. 6, June 2003, pp. 710-723.
18. C. Moore and J. P. Crutchfield, "Quantum Automata and Quantum Grammars", *Theoretical Computer Science*, Vol. 237, No. 1-2, pages 275-306, 2000.
19. ID Quantique, "Quantis: Quantum Random Number Generator", http://www.idquantique.com/qrng.html, March 2004.