Journal of Quantum Information Science, **2012, 2, 61-65**
http://dx.doi.org/10.4236/jqis.2012.23011 Published Online September 2012 (http://www.SciRP.org/journal/jqis)

Permutation Algebra for Constructing Reversible Circuits

Shah Mohammad Bahauddin, Abu Ashik Md. Irfan
Department of Applied Physics, Electronics & Communication Engineering, Faculty of Engineering & Technology,
University of Dhaka, Dhaka, Bangladesh
Email: bahauddin_omar@ieee.org

Received April 17, 2012; revised May 9, 2012; accepted June 4, 2012

ABSTRACT

In this paper, we show that the algebra of permutation group is one of the inherent structures of reversible logic for quantum computation. In this venture, we discuss necessary properties of cycle and transposition to reveal the potential of permutation algebra for reversible logic. Then we present an efficient method which naturally interconnects the structure of reversible logic with the expression of cycle and corresponding transpositions. Finally we discuss several examples which show that the algebra can be effectively used to construct complex gates as well.

Keywords: Control NOT; Taffoli-Fredkin Gate; Full Adder

1. Introduction

In recent years, reversible logic has become more and more prominent technology having its applications in quantum computing and other aspects of nanotechnology. Moreover, reversibility can also be a solution to the energy dissipation issue in chip level integrated systems. Logically reversible gates are those where the inputs can be inferred from knowledge of the outputs, meaning the logic function implemented by the gate has a unique inverse. In this paper, we present a method which interconnects the structure of reversible logic with permutation algebra. By applying the properties of cycle and transposition, we also show that any complex arithmetic circuit such as full adder can be easily achieved and efficiently constructed.

2. Reversibility & Permutation Group

An n-bit reversible logic gate is a block which performs a permutation on the 2^n possible states of bits that pass through it. Hence it is a one-to-one correspondence from a 2^n state onto itself. Thus specifying a reversible logic gate is nothing but specifying the mapping that it performs. In quantum computing, these computational states are represented by quantum mechanical state. For example, let us define an arbitrary finite set S having n-elements (which are in this case well defined quantum mechanical states). Then any reversible logic of such set can be expressed by introducing $\varphi \in S_N$ where φ is a one-to-one mapping of S onto itself. Now, one can represent the set of bits $\{a,b \in (0,1)\}$ with the set of kets $|i\rangle$ (such that i = 0, 1, 2, 3) which are the four computational states expressed by two input bits of the measurement gate. This allows us to represent any logic (in this example XOR or more specifically Control NOT) by means of specific basis state as

$a_{control}$	b_{target}	$\|i_{input}>$	$a_{control}$	b_{target}	$\|i_{output}>$
0	0	0	0	0	0
0	1	1	0	1	1
1	0	2	1	1	3
1	1	3	1	0	2

A closer examination immediately reveals that this representation can be shown as an element under the permutation S_4. Hence we can say C-NOT can be represented as a one-to-one mapping of S_4, and in terms of energy state, the above table can now be represented by the permutation.

$$\varphi = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 3 & 2 \end{pmatrix}$$

Since the S_4 contains 4! permutations it means that there exists 24 different logic gates for a 4-state system. Generalizing the above statements we can say that, any N-level quantum system can be permutated in N! computationally distinct logic sets. In other words, these permutations of degree N form a group of order N! under composite operation of permutation. This permutation group, S_N is the collection of all bijective maps $\varphi : S \to S$ of the interval $S = \{1,2,3 \quad ,N\}$, with composition maps ( ) as the group operation. We now intend to recall some basic definitions for describing permutations [1].

Cycle of a Permutation: For $k > 1$, a k-cycle is a permutation

$$\varphi = \begin{pmatrix} i_1 & i_2 & & i_{k-1} \\ i_2 & i_3 & & i_k \end{pmatrix} = \left(i_1, i_2, \quad , i_{k-1}, i_k\right)$$

that acts on S in the following way

$$\varphi \text{ maps } \begin{cases} i_1 \to i_2 \to \quad \to i_k \to i_1 : i \in \{i_1, i_2, \quad , i_k\} \\ j \to j : j \notin \{i_1, i_2, \quad , i_k\} \end{cases}$$

Here k is called the length of the cycle. Cycles of length 1 are redundant since they reduce to identity. The order of the entries in a cycle matters, but cycle notation is not unique as the following

$$\begin{aligned} & \left(i_1, i_2, \quad , i_{k-1}, i_k\right) \\ &= \left(i_2, i_3, \quad , i_k, i_1\right) \\ &= \left(i_3, i_4, \quad , i_k, i_1, i_2\right) \\ & \\ &= \left(i_k, i_1, \quad , i_{k-2}, i_{k-1}\right) \end{aligned}$$

Transposition: A cycle of length 2 is called a transposition. Every cycle of length k in S_N ($N > 1$) is expressible as a product ($k - 1$) of transpositions in the following manner,

$$\left(i_1, i_2, \quad , i_{k-1}, i_k\right) = \left(i_{1,} i_k\right)\left(i_1, i_{k-1}\right) \quad \left(i_1, i_3\right)\left(i_1, i_2\right)$$

Since, the inverse of a permutation is the permutation obtained by interchanging the rows between them. This results to state that every transposition is the inverse of its own.

$$\left(i_j, i_k\right)\left(i_j . i_k\right) = I \Rightarrow \left(i_j, i_k\right)^{-1} = \left(i_j, i_k\right) = \left(i_k, i_j\right)$$

A product of disjoint cycle or transpositions is commutative.

$$\left(i_j, i_k\right)\left(i_m, i_n\right) = \left(i_m, i_n\right)\left(i_j, i_k\right)$$

An important property which will be particularly interesting for this paper is that one can decompose a transposition in the following product,

$$\left(i_j, i_l\right) = \left(i_j, i_k\right)\left(i_k, i_l\right)\left(i_j, i_k\right)$$

3. Algebra of Reversible Logic

Now we show that the algebra of reversible logic can be introduced in terms of permutation group and the concept of cycle works as the building block of reversible gate. Hence, we adopt the following statements,

For every n-bit reversible logic there exist a unique permutation of degree 2^n*. This means, the properties and the operations of permutation groups are applicable for reversible logic as well.*

This statement is discussed in the beginning of the literature. Recall that as a subgroup of a symmetric group, all the necessary group axioms are satisfied by permutation group; that is it contains the identity permutation, the inverse permutation and the associative and closure property. Since reversible logic mimics a permutation as shown before, the composite operations between permutations can also be adapted for reversible gates.

Since every reversible gate is expressible by corresponding permutation, hence it is possible to express a reversible gate as a product of disjoint cycles and necessarily as a product of transpositions.

Let us examine the logic of an inverter which is $0 \to 1$ and $1 \to 0$ since it is a single bit logic, the gate can be expressed as a permutation of degree 2 which is

$$\varphi_{\text{NOT}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = (0,1)$$

It can be seen that the inverter is logically reversible, but unfortunately is not a universal gate in the sense that any arbitrary Boolean logic is not realizable in terms of inverter alone. For quantum computation, we need universal reversible gates. One of such gates is the Taffoli-Fredkin (T-F) gate [2,3] which is necessarily a 3-bit gate with 3 inputs (a, b, c) and corresponding outputs (a', b', c'). The relationship between the input and output is the following

$$\begin{aligned} a' &= a \\ b' &= b \\ c' &= c \oplus (a \cdot b) \end{aligned}$$

Here $\oplus$ denotes the Exclusive OR and $\cdot$ denotes the AND function. The variables a and b are control bits where c is called the target bit. As seen from the logic, the control bit suffers from no change while passing through the gate. However the target bit flips if both control bits are logic 1. The permutation corresponding to this gate is,

$$\varphi_{\text{T-F}} = \begin{pmatrix} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ 000 & 001 & 010 & 011 & 100 & 101 & 111 & 110 \end{pmatrix}$$

And the corresponding transposition is $\varphi_{\text{T-F}}$ = (110, 111). We can also write the transposition in decimal symbol which would be more convenient for calculation. In case of T-F gate, the transposition is (6, 7).

The similar treatment can be accounted for more complex gates. Such a gate is Peres gate [4], which is another reversible and universal one. Peres gate is mainly interesting due to its combination of T-F gate and C-NOT gate which makes it an attractive candidate to realize complex logic such as addition, subtraction etc. Peres gate is also a 3-bit gate which maintains relationship between the input and output as following,

$$a' = a$$
$$b' = a \oplus b$$
$$c' = c \oplus (a \cdot b)$$

As seen from the logic, the outputs of bit b and c suffer from CNOT and T-F operations respectively while bit a remains unchanged. The permutation corresponding to this gate is,

$$\varphi_{\text{Peres}} = \begin{pmatrix} 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ 000 & 001 & 010 & 011 & 110 & 111 & 101 & 100 \end{pmatrix}$$

The corresponding cycle has a length of 4 and can be decomposed in 3 transpositions,

$$\varphi_{\text{Peres}} = (100,110,101,111)$$
$$= (100,111)(100,101)(100,110)$$

In decimal symbol, the cycle of Peres gate is (4, 6, 5, 7) = (4, 7)(4, 5)(4, 6).

4. Construction of General Gates

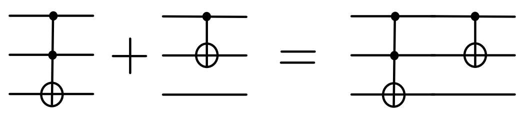
We showed earlier, with example, that the product of cycles of a permutation is sufficient to realize any reversible logic. In an extension, let us introduce an n bit C-NOT gate. In this case, we can realize the C-NOT operation on any one bit of n while the control bits are the other $(n - 1)$ bits or $(n - 2)$ and so on. If $n = 3$, then we can choose any one as a target bit out of three and for each target bit, there exist $2^2 - 1 = 3$ unique combinations of control bits. This means there are 9 possible C-NOT gate for $n = 3$ (**Figure 1**), where each of them can be realized by a unique product of transpositions.

Now we will utilize the properties of cycles, to prove that any reversible logic can be simulated by a combination of elementary gates. In other words, we will prove that the decomposition of reversible logic is possible by exploiting the properties of cycles under permutation algebra.

As an example, we can realize Peres gate in terms of the 3-bit C-NOT gates

$\varphi_{\text{Peres}} = (4, 6, 5, 7)$
$= (6, 5, 7, 4)$
$= (4, 6)(6, 7)(5, 6)$
$= (4, 6)(5, 7)(5, 7)(6, 7)(5, 7)(5, 7)(5, 6)$
$= (4, 6)(5, 7)(5, 6)(5, 7)(5, 6)$
$= (4, 6)(5, 7)(6, 7)$

Now going from right to left, all we have to do is to sum up the corresponding C-NOT gates as per transpositions. Then the decomposition will denote Peres gate through the joining of 3-bit C-NOT gates as shown below:



In the above example, let the C-NOT of cycle (6, 7) and (4, 6) (5, 7) are expressed as C_1 and C_2 respectively. Then clearly C_1, C_2 are one-to-one mapping of $S = \{0,1,2, \quad ,7\}$ onto itself and the Peres gate is merely a composite ($C_1 \quad C_2$) of C_1 and C_2 which can mathematically written by $(C_1 \quad C_2)(x) = C_1(C_2(x))$, $\forall x \in S$. Since this is the closure property of permutation group, it is obvious that $C_1 \quad C_2 \in S_8$.

Similarly, for $n = 4$, we have four options to choose any one as a target bit and for each target bit, there exists $2^3 - 1 = 7$ unique combinations of control bits. Hence there are $7 \times 4 = 28$ numbers of C-NOT gate for $n = 4$ (**Table 1**). Again each of the 28 gates can be realized by a unique product of transpositions as shown in **Figure 2**.

At this juncture, let us imagine that we intend to realize the logic of a full adder, which is evidently irreversible. By adding an extra bit, we can make it reversible, which means it would be a 4-bit reversible logic. But now the question lies where we want to put that extra bit. Since there were 3 bits in an irreversible full adder, we have different positions of significance (from MSB to LSB) to put that extra bit. In our paper, we have introduced 3 logics of such reversible Full Adder where the extra bit is set at three different positions of significance starting from LSB. The permutation corresponding to the gates are given below. Here, in all three cases, the output of a full adder logic is realized only when the extra bit is zero. In addition, the least significant two bits of the outputs are the sum and carry-out.

$$\varphi_{\text{FA-1}} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 & 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ 0000 & 0011 & 0110 & 0100 & 0010 & 0001 & 0101 & 0111 & 1010 & 1011 & 1101 & 1100 & 1001 & 1000 & 1111 & 1110 \end{pmatrix}$$

$$\varphi_{\text{FA-2}} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 & 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ 0000 & 0010 & 0001 & 0011 & 0110 & 0101 & 0111 & 0100 & 1110 & 1101 & 1111 & 1100 & 1001 & 1011 & 1000 & 1010 \end{pmatrix}$$

$$\varphi_{\text{FA-3}} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 & 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ 0000 & 0010 & 0110 & 0001 & 0100 & 0101 & 0011 & 0111 & 1010 & 1001 & 1101 & 1011 & 1100 & 1000 & 1110 & 1111 \end{pmatrix}$$

Now the corresponding cycles in decimal are,

$\varphi_{\text{FA-1}} = (1, 3, 4, 2, 6, 5)(8, 10, 13)(9, 11, 12)(14, 15)$
$\varphi_{\text{FA-2}} = (1, 2)(4, 6, 7)(8, 14)(9, 13, 11, 12)(10, 15)$
$\varphi_{\text{FA-3}} = (1, 2, 6, 3)(8, 10, 13)$

As we stated before, our intention is to decompose these reversible adder logics by 4-bit C-NOT gates only.

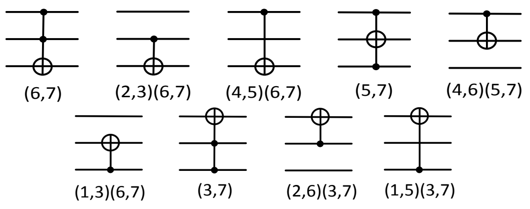


Figure 1. Possible permutations and corresponding transpositions of 3-bit C-NOT gates.

Table 1. Transpositions of 4-bit C-NOT gates corresponding to Figure 2.

Row 1 (target bit at 1st LSB)		Row 3 (target bit at 3rd LSB)	
No.	Transpositions	No.	Transpositions
1	(14,15)	1	(11,15)
2	(6,7)(14,15)	2	(3,7)(11,15)
3	(10,11)(14,15)	3	(9,13)(11,15)
4	(12,13)(14,15)	4	(10,14)(11,15)
5	(8,9)(10,11)(12,13)(14,15)	5	(8,12)(9,13)(10,14)(11,15)
6	(4,5)(6,7)(12,13)(14,15)	6	(2,6)(3,7)(10,14)(11,15)
7	(2,3)(6,7)(10,11)(14,15)	7	(1,5)(3,7)(9,13)(11,15)
Row 2 (target bit at 2nd LSB)		**Row 4 (target bit at MSB)**	
No.	Transpositions	No.	Transpositions
1	(13,15)	1	(7,15)
2	(5,7)(13,15)	2	(3,11)(7,15)
3	(9,11)(13,15)	3	(5,13)(7,15)
4	(12,14)(13,15)	4	(6,14)(7,15)
5	(8,10)(9,11)(12,14)(13,15)	5	(4,12)(5,13)(6,14)(7,15)
6	(4,6)(5,7)(12,14)(13,15)	6	(2,10)(3,11)(6,14)(7,15)
7	(1,3)(5,7)(9,11)(13,15)	7	(1,9)(3,11)(5,13)(7,15)

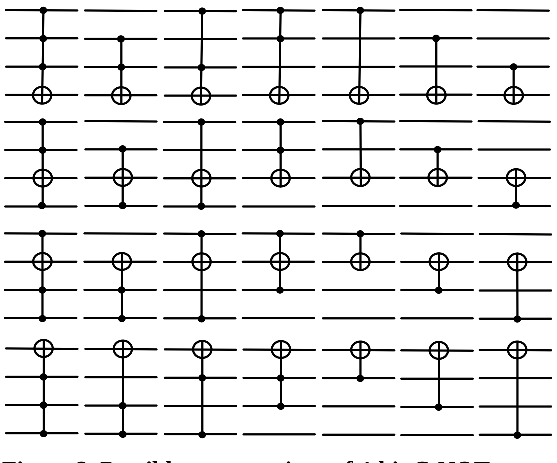


Figure 2. Possible permutations of 4-bit C-NOT gates.

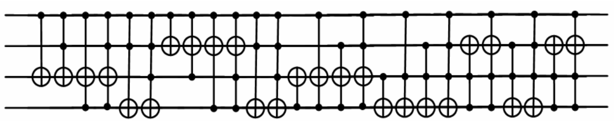


Figure 3. Realization of full adder φ_{FA-3} expressed by 26 4-bit C-NOT gates.

Here, we will utilize the expression of different C-NOT gates in terms of corresponding transpositions and we will operate permutation algebra on these transpositions according to the lemmas to obtain the logic of reversible adder. So, let us first examine the reversible logic of φ_{FA-3} and then decompose it into elementary transpositions which are available for C-NOT operation in following manner,

$$\begin{aligned}\phi_{FA-3} &= (1, 2, 6, 3)(8, 10, 13)\\ &= (3, 1, 2, 6)(8, 10, 13)\\ &= (3, 6)(3, 2)(3, 1)(8, 13)(8, 10)\\ &= (3, 7)(6, 7)(3, 7)(2, 3)(1, 3)(12, 13)(8, 12)(12, 13)\\ &\quad (8, 10)\end{aligned}$$

Now, if we give a closure look at the figures of C-NOT gates previously sketched before with their corresponding product of transpositions, we can immediately reveal that the first transposition (3, 7) cannot be realized alone by any C-NOT gate rather it can be construted by a combination of (3, 7) (11, 15) and (11, 15). According to the previous statement, we can say that both of the transpositons (6, 7) and (12, 13) also require a combination of two C-NOT gates while each of the four transpositions (1, 3), (2, 3), (8, 10) and (8, 12) requires a combination of four distinct C-NOT gates. In summary, we have to add a total of 26 C-NOT gates to realize the logic of φ_{FA-3} (**Figure 3**).

It is now evident that we could also simulate the logic of φ_{FA-2} (shown in **Figure 4**) and φ_{FA-1} (not shown) as well. Our calculation shows that the φ_{FA-2} can be constructed by 40 C-NOT gates while φ_{FA-1} will be comprised of 53 C-NOT gates.

5. Conclusion

The decomposition of complex gate by means of the product of transposition and corresponding *n*-bit C-NOT gate gives us further opportunity to express the logic in more elementary gates. This is because any *n*-bit C-NOT gate and corresponding unitary operations can be expressed as a composition of 2 bit C-NOT and the T-F

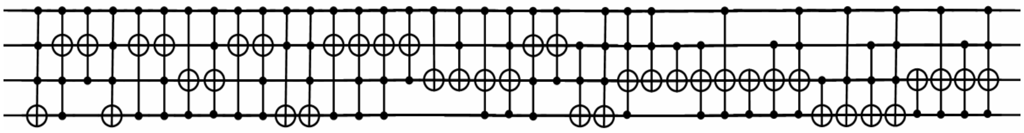


Figure 4. Realization of full adder φ_{FA-2} expressed by 40 4-bit C-NOT gates.

gate [5]. Moreover, the number of basic operations to perform all these steps showed in our examples is mathematically efficient and unique and thus can be easily estimated. Hence we conclude that permutation algebra can be employed as an inherent structure to construct complex reversible circuit for quantum computational networks.

REFERENCES

[1] J. F. Humphreys, "A Course in Group Theory," 5th Edition, Oxford University Press, Oxford, 1996.

[2] T. Taffoli, "Reversible Computing," *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, Vol. 85, 1980, pp. 632-644. doi:10.1007/3-540-10003-2_104

[3] E. Fredkin and T. Taffoli, "Conservative Logic," *International Journal of Theoretical Physics*, Vol. 21, No. 3-4, 1982, pp. 219-253. doi:10.1007/BF01857727

[4] A. Peres, "Reversible Logic and Quantum Computers," *Physical Review A*, Vol. 32, No. 6, 1985, pp. 3266-3276. doi:10.1103/PhysRevA.32.3266

[5] A. Barenco, C. H. Bennett, C. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin and H. Weinfurter, "Elementary Gates for Quantum Computation," *Physical Review A*, Vol. 52, No. 5, 1995, pp. 3457-3467. doi:10.1103/PhysRevA.52.3457