Majority-Based Reversible Logic Gate

Guowu Yang, William N. N. Hung*, Xiaoyu Song and Marek Perkowski
Portland State University, Portland, Oregon, USA

ABSTRACT

Reversible logic plays an important role in application of adiabatic low power CMOS computing and quantum computing. In this paper we introduce families of reversible gates based on majority and we prove their properties in reversible circuit synthesis. These gates can be used to synthesize reversible circuits of minimum "scratchpad register width" for arbitrary reversible functions. We show that, given a majority Boolean function f with $2k+1$ inputs, f can be implemented by a reversible logic gate with $2k+1$ inputs and $2k+1$ outputs, i.e., without any constant inputs. The problem is formulated in terms of group theory and solved by using the algebraic software GAP for logic synthesis.

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

1 Introduction

Reversible logic circuits play an important role in application of adiabatic low power CMOS computing and quantum computing [3, 4, 21]. Majority (voting) gates are used in many fault-tolerant computing applications [13]. Various reversible gates that realize polarized majority function have relatively inexpensive quantum realizations [13]. There have been extensive work in constructing reversible gates which have certain properties

* William N. N. Hung is now working at Intel Architecture Group, Intel Corporation, Hillsboro, Oregon 97124, USA.

such as universality, symmetry, etc. [1-7, 9, 12, 17, 19, 20, 22]. In particular, there are the synthesis algorithms by composition [15, 12], decomposition [15], factorization [23]., EXOR logic [4, 8, 15, 18, 24], group-theoretic methods [19, 20], synthesis to regular structures [14, 16, 17, 22], synthesis of various forms of reversible cascades [2, 7, 8, 9, 10, 11, 12, 15] and spectral methods [10, 11].

The Miller's gate [10] was proposed for quantum logic realizations or in emerging reversible technologies. We present a novel family of gates, of which the Miller gate is a special case. We show that the new gates can be used in synthesis without constant inputs. More specifically, we show that, given a majority Boolean function f with $2k+1$ inputs, f can be implemented by a reversible logic gate with $2k+1$ inputs and $2k+1$ outputs, i.e., without any constant inputs. The new gates can be used to synthesize reversible circuits of minimum "scratchpad register width" for arbitrary reversible functions. The problem is formulated in terms of group theory and solved by using the algebraic software GAP for logic synthesis [19, 20]. Our result is not only interesting theoretically in comparison with other families of gates [18], but also of practical importance in realization of current quantum computers due to the small possible width of the scratchpad register (this width is limited by 7 in 2002). The new family of gates can realize quantum logic circuits with the smallest possible width.

This paper deals with synthesis of arbitrary reversible functions of n inputs and n outputs that as described by permutation cycles. Our goal is to realize circuits of the smallest width (scratchpad register width, quantum register width). We realize n-input functions with the width of n. Our method is based on group theory [19, 20].

2 Quantum Realizations of Reversible Majority Gates

Binary reversible logic gates and circuits have been proposed in quantum, optical, CMOS, nano-mechanical and DNA realizations. Universal systems of reversible gates include Feynman gate and some other 3*3 gate, such as Toffoli or Fredkin [2]. The 1*1 gate, an inverter, can be always added since it practically costs nothing and its addition decreases

the number of gates and the number of constant inputs (when necessary) in the reversible realizations of arbitrary functions. The main practical question is this: what 3*3 gate(s) should be added to make the reversible synthesis practical and the corresponding circuits realizable. Toffoli and Fredkin gates are most often realized and used in synthesis, but this is perhaps mostly for historical reasons. For instance, in quantum computing the Peres gate has similar processing power to Toffoli and has a simpler realization with 2-qubit quantum primitives (3-qubit or 3*3 gates are not realizable directly in quantum, they must be build from 1-qubit and 2-qubit gates). Nobody proved, however, so far, what is the best realization of any 3-qubit universal quantum gate with a given set of quantum 1-qubit and 2-qubit primitives.

Not much is known about realizations of 3*3 reversible functions in optical, CMOS, nano-mechanical and DNA realizations, except for Toffoli and Fredkin, so the logic synthesis researchers can only speculate about the costs of future realizations. In this case it is important to analyze from the mathematical and logical points of view what would be good gate families and their properties in general-purpose logic synthesis and design of self-repairable fault-tolerant systems. When such gates are well understood, the experimentalists can realize them in practical circuits.

Observe that so far, most researchers evaluated the complexity of reversible gates by the complexity of their binary counterparts in technologies like CMOS. The situation is different for quantum logic, in which the so-called pseudo-binary (binary permutation) circuits can be realized. Although the detailed costs depend on any particular realization technology for quantum logic, where the cost relations between gates (e.g. Fredkin and Toffoli) can differ in NMR and ion trap realizations, the assumptions used in the previous work are far too approximate and should be replaced with more precise gate costs for synthesis algorithms. Two assumptions were taken in the past: (1) every k*k gate has the same cost, (2) the cost of a gate is proportional to the number of inputs/outputs. We propose here another cost approximation that is better suited to quantum realizations: the cost is the number of 2-qubit gates – realizable quantum primitives, disregarding their

internal structures. We will explain below with an example why this choice of cost is reasonable for future algorithms.

Using the well-known realization of Toffoli gate (in dotted rectangle from Figure 1a) [21], the cost of Toffoli gate in terms of 2-qubit quantum primitive gates is 5. This design uses two controlled-V, one controlled-V+ and two Feynman gates. Unitary matrix V is the "Square root of not". The reader can check that since V*V = NOT and V*V+ = I (identity), the circuit in dotted rectangle realizes the binary functions of a Toffoli gate: A=a, B=b, C=a*b $\oplus$ c.

Let us consider a function of three majorities investigated by Miller (example 2 in [11]). We realized this function with one Toffoli and four Feynman gates, found it useful in other designs and thus worthy to be a stand-alone quantum gate. We call it *the Miller gate*. As seen in Figure 1a, the Miller gate requires 4 Feynman gates and a Toffoli gate, which would suggest a cost of 5+2×2 = 9 in terms of 2-qubit quantum primitives. However, using transformations and cost evaluations introduced by Smolin [21], as in Figure 1b, we obtain a solution with cost 7 because each dotted 2×2 subgate in Figure 1b has a cost of 1. Another solution with cost 7 is shown in Figure 1c. It is also based on simple EXOR transformation. Both solutions from figures 1b and 1c are practically realizable and their exact costs would depend on a particular realization technology. So in first approximation we can assume that they have the same cost of 7. Again, the Miller gate looks initially much more complicated than the Toffoli gate (cost 5), as interpreted in a binary logic. But a closer inspection in quantum logic proves that it is just slightly more expensive with a cost of 7. This example shows that it is worthy to realize pseudo-binary gates from truly quantum 2×2 primitives to evaluate more accurate costs of such gates for synthesizing circuits. This applies to all gates considered below. Thus, from now on we can assume that the cost of Toffoli gate is 5 and the cost of Miller is 7. Such costs can be used, for instance, to approximately compare quantum realization costs of two variants of a reversible circuit – one using Toffoli and another one using Miller gates.

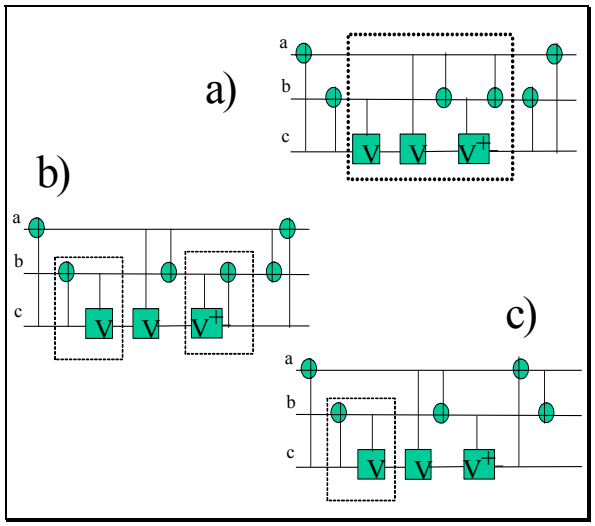

Figure 1. Quantum realizations of Miller gate: (a) with integrated Toffoli gate – cost 9, (b) with integrated 2-qubit gates – cost 7, (c) with one integrated 2-qubit gate – cost 7.

Let us observe (Figure 2a) that, by removing the three rightmost Feynman gates from the gate shown in Figure 1b, we obtain a new reversible gate M_2 that realizes the following functions: $A = a \oplus c$, $B = b \oplus c$, $C = ab \oplus ac \oplus bc$. This gate has the same cost of 5 as the Toffoli gate, and has more processing power. Such gates should be thus used together with *Feynman, NOT* and *Toffoli* gates to obtain exact minimum quantum circuit. Observe for instance, that if the synthesis task would be to minimize the cost of a circuit that realizes a function from Figure 2a, using M_2 gates the solution would use just one M_2 gate and the total gate cost would be 5, but using the standard approach of Toffoli, Feynman, NOT gate base, the cost would be 5+3=8, since two Feynman should be added before and one after the Toffoli gate. Similarly, gate M_3 from Figure 2b can be created, also with cost 5.

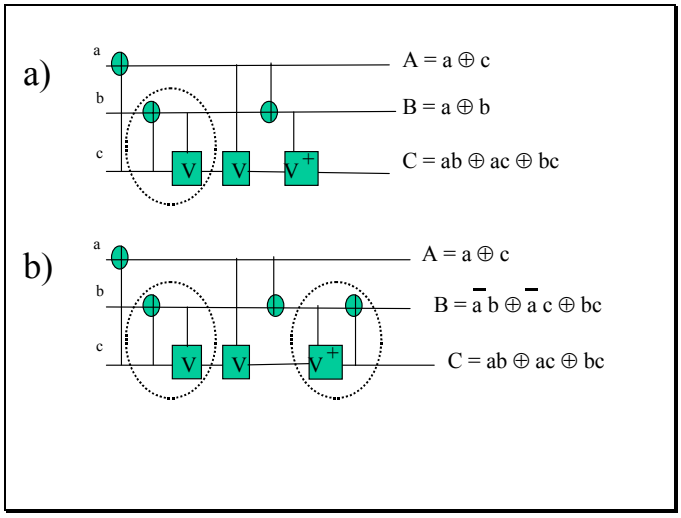

Figure 2. Quantum realizations of Majority gates of cost 5: (a) new gate M_2, (b) new gate M_3.

3 Majority Based Reversible Logic Gates

Given any n-input, n-output, (i.e., $n \times n$) reversible logic gate, we denote its inputs by B_1, B_2, …, B_n. Similarly, we denote its outputs by P_1, P_2, …, P_n. We start with some basic notions on majority.

Definition 1. (Majority Boolean Function) Given a Boolean function f_{MB} with an odd number of inputs, (i.e., $2k+1$ inputs, where k is a natural number), f_{MB} is called a Majority Boolean Function (MBF) when f_{MB} returns TRUE if and only if more than half (i.e., $k+1$ or more) of its inputs are TRUE:

$$f_{MB}(B_{2k+1},\cdots,B_2,B_1) = \sum_{1\le j_1<j_2<\ldots<j_k<j_{k+1}\le 2k+1} Q_{j_1 j_2 \ldots j_k j_{k+1}} \text{ , where } Q_{j_1 j_2 \ldots j_k j_{k+1}} = B_{j_1} B_{j_2} \ldots B_{j_k} B_{j_{k+1}} .$$

Because we want to use majority gates repeatedly in quantum circuit design, and at the same time we do not want to increase the width, an interesting question to ask is whether a majority Boolean function can be implemented in reversible logic using the same number of inputs used by the function, i.e., without introducing any constant inputs and garbage outputs (the technique of adding constant inputs and garbage (useless) outputs is popularly used in reversible logic design as a "last resort" method). To answer this

question, we first establish a necessary and sufficient condition for any Boolean function to be implementable using reversible logic without any garbage inputs.

Lemma 1. Given the truth table of any $n \times n$ reversible logic gate, the output entries must be a permutation of the input entries in the truth table.

Proof: There are 2^n entries (patterns) for inputs, and they are all unique. There are also 2^n entries (patterns) for outputs. Since the logic gate is reversible, all output entries must be distinct. Hence we have 2^n unique entries for all n outputs as well. Thus the 2^n output entries of the truth table must be a permutation of the 2^n input entries. ■

Lemma 2. Given any single-output Boolean function f with n inputs, f can be implemented by a $n \times n$ reversible logic gate (where one of the gate outputs equals f) if-and-only-if the number of 1's and 0's are the same in the output column of the truth table for f.

Proof:
(IF) The input entries of a truth table must have the same number of 1's and 0's. If f has the same number of 1's and 0's in the output column of its truth table, we can construct a logic gate with a truth table such that the output entries are a permutation of the input entries and one of the output column is the same as the output of function f. Since the output entries are a permutation of the input entries, the logic gate is reversible. This truth table is the $n \times n$ reversible logic gate that implements f in one of its outputs.
(ONLY-IF) Let's construct a truth table with all n inputs and all n outputs of the logic gate. Using Lemma 1, the 2^n output entries of the truth table must be a permutation of the 2^n input entries. The input entries of a truth table have the same number of 1's and 0's in each column. Hence, the output entries must have the same number of 1's and 0's in each column. ■

Theorem 1. Given a Majority Boolean Function f with $2k+1$ inputs, f can be implemented by a reversible logic gate with $2k+1$ inputs and $2k+1$ outputs, i.e., without any constant inputs.

Proof: Since *f* is a MBF, its output is 1 if-and-only-if there are *k* or less inputs that are assigned to 0. The number of entries in the truth table with output 1 is:

$$N = C_0^{2k+1} + C_1^{2k+1} + C_2^{2k+1} + \cdots + C_k^{2k+1}$$

Since $C_r^n = C_{n-r}^n$, we have:

$$N = C_{2k+1}^{2k+1} + C_{2k}^{2k+1} + C_{2k-1}^{2k+1} + \cdots + C_{k+1}^{2k+1}$$

By adding the above two equations, we have:

$$2N = C_0^{2k+1} + C_1^{2k+1} + C_2^{2k+1} + \cdots + C_k^{2k+1} + C_{k+1}^{2k+1} + \cdots + C_{2k+1}^{2k+1}$$

This is simply the sum of powers of binomial coefficients. Using the binomial theorem, we have:

$$2N = \sum_{j=0}^{2k+1} C_j^{2k+1} (1)^j = (1+1)^{2k+1} = 2^{2k+1}$$

Since there are $2k+1$ inputs, there are 2^{2k+1} entries in the truth table. From the above equation, half of these entries have output equal to 1. Thus, the other half have output equal to 0. Hence there are equal number of 1's and 0's in the output column for *f*. Using Lemma 2, we know that *f* can be implemented by a $(2k+1)\times(2k+1)$ reversible logic gate.
■

Now that we know that a Majority Boolean Function can be implemented in reversible logic without constant input, we introduce a special reversible logic gate for this function.

Definition 2. (Majority-Based Reversible Logic Gate) A reversible logic gate is called a *Majority-Based Reversible Logic Gate* (**MBRLG**) if it has an odd number of

inputs and outputs, (i.e., $2k$+1 inputs and $2k$+1 outputs), such that at least one output is a Majority Boolean Function of all its inputs.

We use n-MBRLG to denote an n-input, n-output, (i.e., $n \times n$) MBRLG, where n is an odd number. For example, a 3×3 MBRLG is called a 3-MBRLG.

Theorem 2. There are 576 3-MBRLG's where the last output P_3 is the majority Boolean function of all 3 inputs.

Proof: The last output is $P_3 = B_1B_2 + B_1B_3 + B_2B_3$. Its truth table is shown in **Table 1**.

B_3	B_2	B_1	P_3
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 1: Truth table of 3-input majority Boolean function.

There are $(2^3)!=8!$ ways to permute the input entries to form the output entries in the truth table of any 3x3 reversible logic gate. The output of the MBF has four 0's and four 1's. So, there are C_4^8 combinations to arrange these four 0's and 1's. Only one of these combinations would correctly implement the MBF. Hence, the number of 3-MBRLG's where the last output is the majority Boolean function is

$$(8!)\cdot\frac{1}{C_4^8} = (4!)^2 = 576 \qquad \blacksquare$$

Theorem 3. There are $\left[(2^{2k})!\right]^2$ distinct $(2k+1)$-MBRLG's for every natural number k.

Proof: We construct a reversible logic gate such that the last output is the majority Boolean function of all inputs: $P_{2k+1} = \sum_{1 \le j_1 < j_2 < \ldots < j_k < j_{k+1} \le 2k+1} Q_{j_1 j_2 \ldots j_k j_{k+1}}$, where $Q_{j_1 j_2 \ldots j_k j_{k+1}} = B_{j_1} B_{j_2} \ldots B_{j_k} B_{j_{k+1}}$. From Lemma 2 and Theorem 1, we know the numbers of 1's and 0's for the output of this majority Boolean function are the same. There are 2^{2k+1} entries in the truth table of this MBF. So we need to place 2^{2k} 0's in 2^{2k+1} output entries. The total number of different placements is:

$$\frac{(2^{2k+1})!}{C_{2^{2k}}^{2^{2k+1}}} = \frac{(2^{2k+1})!}{\left[\frac{(2^{2k+1})!}{(2^{2k})!(2^{2k+1}-2^{2k})!}\right]} = [(2^{2k})!]^2 \qquad \blacksquare$$

As a sanity check, we can use Theorem 3 when $k = 1$. With this instantiation, there are $[(2^{2(2)})!]^2 = 576$ distinct 3-MBRLG's, which agrees with Theorem 2.

We now construct a (2k+1)-MBRLG, M_{2k+1}, with the following outputs:

$P_{2k+1} = f_{MB}(B_{2k+1}, \cdots, B_1) = \sum_{1 \le j_1 < j_2 < \ldots < j_k < j_{k+1} \le 2k+1} Q_{j_1 j_2 \ldots j_k j_{k+1}}$, where $Q_{j_1 j_2 \ldots j_k j_{k+1}} = B_{j_1} B_{j_2} \ldots B_{j_k} B_{j_{k+1}}$.

$P_i = B_i \oplus B_{2k+1}$, for $i = 1,2,\cdots,2k$.

Theorem 4. M_{2k+1} is a (2k+1)-MBRLG.

Proof: The last output P_{2k+1} matches Definition 1, which means it is a majority Boolean function. There are 2^{2k+1} entries in the truth table. For each entry i, let U_i be the Boolean number encoding of the input pattern, B_{2k+1}, B_{2k}, …, B_2, B_1, in that entry, and let V_i be the Boolean number encoding of the input pattern, P_{2k+1}, P_{2k}, …, P_2, P_1, in that entry. To show reversibility, we only need to prove that $V_1, V_2, \cdots, V_{2^{2k+1}}$ are different numbers. We place these numbers into two sets: $S_1 = \{V_1, \cdots, V_{2^{2k}}\}$, $S_2 = \{V_{2^{2k}+1}, \cdots, V_{2^{2k+1}}\}$. For S_1, the last 2k bits of each number (P_{2k}, …, P_2, P_1) is a counting from (0,0,…,0) to (1,1,…,1). All

these numbers are different. For S_2, the last $2k$ bits of each number (P_{2k}, …, P_2, P_1) is a counting from (1,1,…,1) to (0,0,…,0). These numbers are also different. Thus the numbers within each set are distinct. Now we only need to prove that S_1 and S_2 are disjoint, i.e., $\forall V_i \in S_1 \cdot \forall V_j \in S_2 \cdot V_i \neq V_j$. If the last $2k$ bits of V_i and V_j are different, then they are different of course. Otherwise, the last $2k$ bits of V_i and V_j are the same. We have $V_i = (Q_{2k+1}, P_{2k}, P_{2k-1}, \cdots, P_2, P_1)$ and $V_j = (R_{2k+1}, P_{2k}, P_{2k-1}, \cdots, P_2, P_1)$. We will show that $Q_{2k+1} \neq R_{2k+1}$ for this case. The corresponding inputs for V_i and V_j are: $U_i = (0, P_{2k}, P_{2k-1}, \cdots, P_2, P_1)$ and $U_j = (1, \overline{P_{2k}}, \overline{P_{2k-1}}, \cdots, \overline{P_2}, \overline{P_1})$. Thus, for the majority function outputs, we have:

$$Q_{2k+1} = f_{MB}(0, P_{2k}, P_{2k-1}, \cdots, P_2, P_1) \text{ and } R_{2k+1} = f_{MB}(1, \overline{P_{2k}}, \overline{P_{2k-1}}, \cdots, \overline{P_2}, \overline{P_1})$$

Let C_{ONE} be a function that counts the number of ones in its arguments.

If $Q_{2k+1} = 1$, then $C_{ONE}(P_{2k}, \cdots, P_2, P_1) \geq k+1$

$$\Rightarrow C_{ONE}(\overline{P_{2k}}, \cdots, \overline{P_2}, \overline{P_1}) < (2k+1) - (k+1) = k$$

$$\Rightarrow 1 + C_{ONE}(\overline{P_{2k}}, \cdots, \overline{P_2}, \overline{P_1}) < 1 + k$$

$$\Rightarrow R_{2k+1} = 0$$

Similarly, $(Q_{2k+1} = 0) \Rightarrow (R_{2k+1} = 1)$. Hence, V_i and V_j are different. ■

4 Reversible Logic Synthesis Using MBRLG

We consider logic synthesis of reversible circuits using a collection of reversible logic gates. Let us consider the following two 3-MBRLG's:

M_1: $P_3 = B_1B_2 + B_1B_3 + B_2B_3$, $P_2 = \overline{B_1}B_2 + \overline{B_1}B_3 + B_2B_3$, $P_1 = B_1\overline{B_2} + B_1B_3 + \overline{B_2}B_3$

M_2: $P_3 = B_1B_2 + B_1B_3 + B_2B_3$, $P_2 = B_1 \oplus B_2$, $P_1 = B_1 \oplus B_3$

We are interested in the synthesis of reversible logic circuits using these gates.

Theorem 5. Any 3-input reversible logic gate can be synthesized without input constants using Feynman gates, NOT gates (inverters) and gates of the type M_1.

Theorem 6. Any 3-input reversible logic gate can be synthesized without input constants using Feynman gates, NOT gates (inverters) and gates of the type M_2.

Proof (for both Theorem 5 and Theorem 6): Using Lemma 1, the output entries of any reversible logic gate must be a permutation of all input entries. Thus, we can establish a bijective (one-to-one) mapping of all 3-input reversible logic gates (using their truth table) onto the permutation group $S_{2^3} = S_8$. We can also establish bijective mappings of each gate onto their corresponding subgroups:

Inputs				Outputs			
B_3	B_2	B_1	Encoding	P_3	P_2	P_1	Encoding
0	0	0	1	0	0	0	1
0	0	1	2	0	0	1	2
0	1	0	3	0	1	1	4
0	1	1	4	0	1	0	3
1	0	0	5	1	0	0	5
1	0	1	6	1	0	1	6
1	1	0	7	1	1	1	8
1	1	1	8	1	1	0	7

Table 2. Feynman (Fe_{12}) where $P_1 = B_1$ XOR B_2 *and* Fe_{12} = (3,4)(7,8).

Inputs				Outputs			
B_3	B_2	B_1	Encoding	P_3	P_2	P_1	Encoding
0	0	0	1	0	0	0	1
0	0	1	2	1	0	1	6
0	1	0	3	0	1	0	3
0	1	1	4	1	1	1	8
1	0	0	5	1	0	0	5
1	0	1	6	0	0	1	2
1	1	0	7	1	1	0	7
1	1	1	8	0	1	1	4

Table 3. Feynman (Fe_{31}) where $P_3 = B_1$ XOR B_3 *and* Fe_{31} = (2,6)(4,8).

Inputs				Outputs			
B_3	B_2	B_1	Encoding	P_3	P_2	P_1	Encoding
0	0	0	1	0	0	1	2
0	0	1	2	0	0	0	1
0	1	0	3	0	1	1	4
0	1	1	4	0	1	0	3
1	0	0	5	1	0	1	6
1	0	1	6	1	0	0	5
1	1	0	7	1	1	1	8
1	1	1	8	1	1	0	7

Table 4. NOT gate for input 1: n_1 = (1,2)(3,4)(5,6)(7,8).

Inputs				Outputs			
B_3	B_2	B_1	Encoding	P_3	P_2	P_1	Encoding
0	0	0	1	0	0	0	1
0	0	1	2	0	0	1	2
0	1	0	3	0	1	0	3
0	1	1	4	1	0	0	5
1	0	0	5	0	1	1	4
1	0	1	6	1	0	1	6
1	1	0	7	1	1	0	7
1	1	1	8	1	1	1	8

Table 5. 3-MBRLG: M_1 = (4,5).

Inputs				Outputs			
B_3	B_2	B_1	Encoding	P_3	P_2	P_1	Encoding
0	0	0	1	0	0	0	1
0	0	1	2	0	0	1	2
0	1	0	3	0	1	0	3
0	1	1	4	1	1	1	8
1	0	0	5	0	1	1	4
1	0	1	6	1	1	0	7
1	1	0	7	1	0	1	6
1	1	1	8	1	0	0	5

Table 6. 3-MBRLG: $M_2 = (4,8,5)(6,7)$

We can generate new groups using the above defined subgroups as generators:

g_1 = Group generated by Fe_{12}, Fe_{31}, n_1, M_1.

g_2 = Group generated by Fe_{12}, Fe_{31}, n_1, M_2.

Using GAP software [19], we can compute the size (number of permutations) for each group:

$|S_8| = 40320$, $|g_1| = 40320$, $|g_2| = 40320$.

Since $|S_8| = |g_1| = |g_2|$, and $|S_8|$ contains all possible permutations for 3-input reversible gates, we know that our propositions holds. ■

Notice that without M_1 or M_2, the size of the group generated by Fe_{12}, Fe_{31}, n_1, is only 32, which is smaller than $|S_8|$. Even if we factor in other variants of Feynman gates and inverters (by exchanging wire configurations), the generated group size is only 1344, which is still less than $|S_8|$. So using Feynman gate along with inverters (NOT gates) is not sufficient to synthesize all 3-input reversible logic functions. It is necessary to have some other additional gate, and we concentrate on majority gates such as M_1 or M_2.

We can also use 5-MBRLG's for logic synthesis. Consider the following gate where the last output is the majority Boolean function:

M_5: $P_5 = f_{MB}(B_1, B_2, B_3, B_4, B_5)$, $P_i = B_i \oplus B_5$, for $i = 1, \cdots, 4$.

The subgroup corresponding to the truth table of this gate is:

M_5 = (7,23,24)(11,27,20)(13,29,18)(14,30,17)(15,31,16)(19,28)(21,26)(22,25)

Similarly, we can also find the subgroup for the exchangers (SWAP gates) and inverter (NOT gate):

e_{12} = (2,3)(6,7)(10,11)(14,15)(18,19)(22,23)(26,27)(30,31)

e_{23} = (3,5)(4,6)(11,13)(12,14)(19,21)(20,22)(27,29)(28,30)

e_{34} = (5,9)(6,10)(7,11)(8,12)(21,25)(22,26)(23,27)(24,28)

e_{45} = (9,17)(10,18)(11,19)(12,20)(13,21)(14,22)(15,23)(16,24)

n_1 = (1,2)(3,4)(5,6)(7,8)(9,10)(11,12)(13,14)(15,16)(17,18)
(19,20)(21,22)(23,24)(25,26)(27,28)(29,30)(31,32)

Theorem 7. Any 5-input reversible logic gate can be synthesized using SWAP gates (2×2 gate that exchanges the two wires), NOT gates and gates of the type M_5.

Proof: The size of the group generated by e_{12}, e_{23}, e_{34}, e_{45}, n_1, and M_5 is equal to 32!, which is the same as the size of the symmetry group S_{32}. ■

Theorem 8. Any 5-input reversible logic gate can be synthesized using Feynman gates, NOT gates and gates of the type M_5.

Proof: SWAP gates can be synthesized by Feynman gates. Thus we can deduce this theorem from Theorem 7. ■

5 Generalized Majority-Based Reversible Logic Gates

We can invert any inputs of a majority Boolean function, resulting in Generalized Majority Boolean Function (GMBF). The GMBF is formed by composing a MBF with zero or more inverters at its inputs.

For example, given a 3-input MBF, $f_{MB}(B_1, B_2, B_3) = B_1B_2 + B_1B_3 + B_2B_3$. We can invert its input B_2 to form a 3-input GMBF,

$$g_{MB}(B_1, B_2, B_3) = f_{MB}(B_1, \overline{B_2}, B_3) = B_1\overline{B_2} + B_1B_3 + \overline{B_2}B_3 .$$

Since we can invert 0, 1, 2 or 3 inputs of the MBF, the total number of distinct 3-input GMBF's is: $C_0^3 + C_1^3 + C_2^3 + C_3^3 = 8$.

Definition 3. (Generalized Majority-Based Reversible Logic Gate) A reversible logic gate is called a Generalized Majority-Based Reversible Logic Gate (**GMBRLG**) if it has an odd number of inputs and outputs, (i.e., $2k$+1 inputs and $2k$+1 outputs), such that at least one output is a Generalized Majority Boolean Function of all its inputs.

We use n-GMBRLG to denote an n-input, n-output, (i.e., $n \times n$) GMBRLG, where n is an odd number. For example, a 3×3 GMBRLG is called a 3-GMBRLG.

Theorem 9. (i) There are 4608 3-GMBRLG's where the last output P_3 is a GMBF. (ii) There are 192 3-GMBRLG's where every output is a GMBF. (iii) There are 72 3-MBRLG's where one output is a MBF and the other two outputs are GMBF's.

Proof:

(i) In Theorem 2, we showed there are 576 3-MBRLG's where the last output P_3 is the MBF of all inputs. In the example above, we also showed that the 3-input MBF is one of the 8 GMBF's. Thus the total number of 3-GMBRLG's is $8 \times 576 = 4608$.

(ii) There are 8 choices (3-input GMBF's) for the first output. When the first output

function is determined, we can look at the truth table and realize that 6 out of the remaining 7 GMBF's can be used for the second output and the gate would still be reversible. For the last output, we use the truth table again and realize that 4 out of the remaining 6 choices (GMBF's) can be used and the gate is still reversible. Hence $8 \times 6 \times 4 = 192$.

(iii) We have 3 choices (3 outputs) to assign the MBF. After that, we have 6 GMBF's for the second output and 4 GMBF's for the third output. Thus $3 \times 6 \times 4 = 72$. ■

6 Conclusion

We generalized the Miller gate concept to new families of gates and we proved that such gates can be useful in synthesis without constant inputs. This result is not only interesting theoretically when compared to other families of gates [18], but also of practical importance in realization of current quantum computers since the small possible width of the scratchpad register continues and will continue to be one of the most difficult barriers to overcome. The families of gates introduced here enable us to realize quantum logic circuits with the smallest possible width.

7 References

1. D. Deutsch, "Quantum computational networks," *Proc. Roy. Soc. Lond*. A. 425, 1989, pp. 73-90.

2. Dueck, G. W., and D. Maslov, "Reversible Function Synthesis with Minimum Garbage Outputs," *Proc. International Workshop on Applications of the Reed-Müller Expansion in Circuit Design* (2003).

3. E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, 21, pp. 219-253, 1982

4. K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation Rules for Designing CNOT-based Quantum Circuits," *Proc. DAC 2002,* New Orleans, Louisiana.

5. P. Kerntopf, "Maximally efficient binary and multi-valued reversible gates," *Proceedings of ULSI Workshop*, Warsaw, Poland, May 2001**,** pp. 55-58.

6. P. Kerntopf, "Synthesis of multipurpose reversible logic gates," *Proceedings of EUROMICRO Symposium on Digital Systems Design*, 2002, pp. 259-266.

7. A. Khlopotine, M. Perkowski, and P. Kerntopf, "Reversible logic synthesis by gate composition," *Proceedings of IWLS* 2002. pp. 261 – 266.

8. M. Lukac, M. Pivtoraiko, A. Mishchenko, and M. Perkowski, "Automated Synthesis of Generalized Reversible Cascades using Genetic Algorithms," *Proc.5th International Workshop on Boolean Problems*, Freiberg, Germany, September 19-20, 2002. pp 33-45.

9. Maslov, D., and G. W. Dueck, "Garbage in Reversible Designs of Multiple-Output Functions," *Proc. International Workshop on Applications of the Reed-Müller Expansion in Circuit Design* (2003).

10. Miller, D. M., "Spectral and Two-Place Decomposition Techniques in Reversible Logic," *Proc. Midwest Symposium on Circuits and Systems,* on CD-ROM, August 2002

11. D. M. Miller and G.W. Dueck, "Spectral Techniques for Reversible Logic Synthesis," *Proc. International Workshop on Applications of the Reed-Müller Expansion in Circuit Design* (2003).

12. A. Mishchenko and M. Perkowski, "Logic Synthesis of Reversible Wave Cascades", *Proc. IEEE/ACM International Workshop on Logic Synthesis,* June 2002. pp. 197 – 202

13. L. Nazhandali and K. A. Sakallah, "Majority-Based Decomposition of Carry Logic in Binary Adders", *IEEE/ACM International Workshop on Logic Synthesis,* June 2002.

14. M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, B. Massey, "Regularity and symmetry as a base for efficient realization of reversible logic circuits," *Proceedings of IWLS 2001*, pp. 90-95, 2001.

15. M. Perkowski, L. Jozwiak, P. Kerntopf, A. Mishchenko, A. Al-Rabadi, A. Coppola, A. Buller, X. Song, M. M. H. A. Khan, S. Yanushkevich, V. Shmerko, and M. Chrzanowska-Jeske, "A general decomposition for reversible logic," *Proceedings of RM 2001*. pp. 119 – 138.

16. M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, B. Massey, "Regular realization of symmetric functions using reversible logic," *Proceedings of EUROMICRO Symposium on Digital Systems Design*, 2001**,** pp. 245-252.

17. P. Picton, "A universal architecture for multiple-valued reversible logic," *MVL Journal*, 5, pp. 27-37, 2000.

18. V. V. Shende, A.K. Prasad, I.L. Markov, J.P. Hayes, "Reversible Logic Circuit Synthesis," *Proc. 11th IEEE/ACM Intern. Workshop on Logic Synthesis*, 2002, pp. 125 – 130.

19. M. Schoenert, "GAP", *Computer Algebra Nederland Nieuwsbrief,* 9, 1992, pp. 19 – 28.

20. L. Storme, A. De Vos, and G. Jacobs, "Group theoretical aspects of reversible logic gates," *Journal of Universal Computer Science*, 5, pp. 307-321, 1999.

21. 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, pp. 2855-2856.

22. A. Al-Rabadi, “Novel Methods for Reversible Logic Synthesis and Their Application to Quantum Computing,” *Ph.D. Thesis*, Portland State University, Portland, Oregon, USA, October 24, 2002.

23. M. H. A. Khan, and M. Perkowski , “Multi-Output ESOP Synthesis with Cascades of New Reversible Gate Family,” *Proc. International Workshop on Applications of the Reed-Müller Expansion in Circuit Design* (2003).

24. F. Luccio and L. Pagli, "On a new Boolean function with applications", *IEEE Transactions on Computers,* 48, 3 (1999) 296--310.