Asymptotically Optimal Regular Synthesis of Quantum Networks

Dmitri Maslov
Faculty of Computer Science
University of New Brunswick
Fredericton, NB, E3B 5A3 Canada
dmaslov@unb.ca

Gerhard W. Dueck
Faculty of Computer Science
University of New Brunswick
Fredericton, NB, E3B 5A3 Canada
gdueck@unb.ca

ABSTRACT

We propose a network of generalized Toffoli gates with multiple EXORs for the realization of reversible functions. If implemented as a quantum circuit, the cost of such gates is shown to be only marginally higher than the cost of a Toffoli gate with a single EXOR and the same number of controls. The main result is a regular synthesis procedure which allows to create asymptotically optimal networks. However, asymptotic optimality does not necessarily mean absolute optimality. Thus, when the algorithm terminates, and a network is created, simplification procedures that may reduce the number of gates in the network can be applied.

1. INTRODUCTION

Reversible logic will play a more significant role as technology evolves. Landauer's principle [6] states that for every bit of information lost, an amount of heat equal to $kT\ln 2$ Joule is dissipated. Although current processors dissipate 500 times this amount of heat [11] every time a bit of information is lost, this amount will become relevant in the future. Assuming that every transistor out of more than $4 * 10^7$ [5] for Pentium-IV technology dissipates heat at a rate of the processor frequency, for instance 1 GHz (10^9 Hz), the figure becomes $2 * 10^{19} * kT\ln 2$ J/sec. The processor's working temperature is greater than 300 degrees, which brings us to $6 * 10^{21} k\ln 2$. Although this amount of heat is still small ($k \approx 1.38 * 10^{-23}$), i.e. only around 0.05 J/sec, Moore's law predicts exponential growth of the heat generated due to the information loss, which will be a noticeable amount of heat loss in the next decade. Bennet [2] showed zero energy dissipation would be possible only if the network consists of reversible gates. Thus reversible logic will become an essential component in future circuit design.

To our best knowledge, not much has been done in regular reversible logic synthesis of a multiple output Boolean function. Mishchenko and Perkowski [10] introduce a reversible structure which is based on an EXOR PLA, thus allowing strait forward implementation of EXOR synthesis results. Unfortunately, this synthesis method requires a large number of garbage bits which are expensive in some technologies, especially quantum. Authors of [7], [3], [9] suggest regular synthesis methods, but they usually produce large networks when applied as formulated.

2. DEFINITIONS

DEFINITION 1. *For the set of domain variables* $\{x_1, x_2, ..., x_n$ **generalized Toffoli gate** *is a gate of a form* $TOF(C, t)$, *where* $C = \{x_{i_1}, x_{i_1}, ..., x_{i_k}\}, t = \{x_j\}$ *and* $C \cap t = \emptyset$ *which map a Boolean pattern* $\{x_1^0, x_2^0, ..., x_n^0\}$ *to* $\{x_1^0, x_2^0, ..., x_{j-1}^0, x_j^0 \oplus x_{i_1}^0 x_{i_1}^0 ... x_{i_k}^0, x_{j+1}^0, ..., x_n^0\}$. *Further, C will be called* **control** *set (or a set of controls), and bit* x_j *the* **target**.

A multiple EXOR Toffoli gate (mEXOR) is defined analogously by allowing the target to be a set. Formally,

DEFINITION 2. *An* **mEXOR gate** $TOF(C, T)$, *where* $C \cap T = \emptyset$ *and* $T = \{x_{j_1}, x_{j_1}, ..., x_{j_m}\}$ *is a single gate that is equivalent to the network* $TOF(C, x_{j_1})\ TOF(C, x_{j_2})...\ TOF(C, x_{j_m})$.

Pictorial representation of a gate mEXOR is shown in Figure 1. The notation does not reflect the actual structure of the gate, which is discussed in section 3.

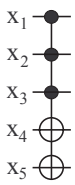

Figure 1: An example of mEXOR Toffoli gate.

3. QUANTUM COST OF THE NEW MODEL

Quantum transformations are necessarily reversible, this follows from the only condition used to determine whether a transformation can be accomplished: it must be a unitary operator on the set of amplitudes. This condition does not

mention how difficult it is to realize a given unitary transform, it only states the theoretical possibility.

In conjunction with reversible logic synthesis, the following transforms can be realized as one gate with unit cost:

- NOT gate (also known as quantum X gate). For Boolean entities it acts as the conventional NOT gate.
- CNOT gate (proposed by Feynman [4]), which acts as $TOF(x_1, x_2)$. In other words, in the Boolean case it flips x_2 iff $x_1 = 1$.

The set of gates NOT, CNOT is not complete since they only realize linear functions. Thus, in order to make the set complete (as a set of Boolean functions), the Toffoli gate [12], $TOF(x_1 + x_2, x_3)$ was added. Unfortunately, this gate cannot be realized as one elementary quantum operation. Even worse, the more controls a generalized Toffoli gate has, the larger its cost in terms of the number of elementary quantum transformations required.

The problem of building quantum blocks to realize Toffoli gates was investigated by many authors. For the comparison of quantum cost of Toffoli and mEXOR Toffoli gates we will use results of [1]. For other implementations the costs can easily be recalculated.

DEFINITION 3. *The quantum cost of a gate G, $|G|$ is the number of basic operations required to realize function given by G.*

No particular realization of a gate (for most of the gates) was proven to be optimal, so the numeric value of the quantum cost may change as soon as better gate realizations are proposed.

THEOREM 1. *For a mEXOR Toffoli gate $TOF(C, T)$, $T = \{x_{j_1}, x_{j_1}, ..., x_{j_m}\}$ its quantum cost is $|TOF(C, t)|+2(m-1)$, where $TOF(C, t)$ is a Toffoli gate with a single target.*

PROOF. The picture in Figure 2 illustrates the construction of a circuit for $TOF(C, T)$ given a circuit for generalized Toffoli gate, shown as a box. Sometimes, a Toffoli gate realization requires additional garbage bits to produce a smaller network. These bits are used in the box, but the initial states of the bits do not have to be set to 0 first, and the output on these bits is reset to the initial input values (as it is done in [1]). For example, the generalized Toffoli gate with 8 controls can be realized with cost 509 if no garbage bits are used, or with cost of 172 elementary quantum operations if 4 garbage bits are allowed as shown in Table 3.

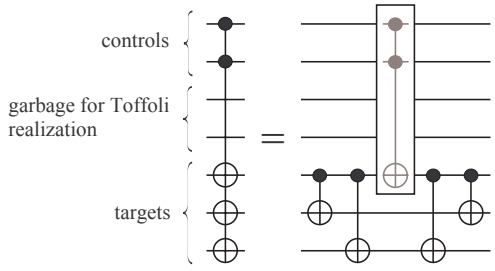

Figure 2: Construction of a single mEXOR gate.

The procedure of building an mEXOR gate $TOF(C, T)$ does not require any additional garbage bits and uses $2(m-1)$ CNOT gates. The procedure also does not require the presetting of garbage bits and returns them unchanged (according to computations done in [1]). □

Table 3 shows the cost comparison for Toffoli and mEXOR gates using the above theorem as a basis for the calculations. The quantum cost of Toffoli gates is taken from [1]. The absolute quantum complexities of Toffoli and 2-target, 4-target and 8-target mEXOR gates with the same number of controls are given in columns **Toffoli**, **2**, **4** and **8**. The relative values of mEXOR gate complexities with respect to the cost of the corresponding Toffoli gate are shown in columns **Rel 2**, **Rel 4** and **Rel 8**. The cost of mEXOR gates with zero or one control is m, since it can be implemented with m NOT or CNOT gates (each has a cost of one).

The following examples illustrate how Theorem 1 can be used to calculate the quantum complexities of mEXOR gates.

EXAMPLE 1. *The mEXOR gate shown in Figure 1 has 3 controls and 2 targets. Thus, its quantum cost is 15, which is approximately 1.154 of the cost of Toffoli gate with the same number of controls. The strait-forward realization of this mEXOR gate as a set of two Toffoli gates would have cost $2 * 13 = 26$, which is approximately 1.733 times higher than the cost in suggested approach.*

EXAMPLE 2. *The higher the number of controls, the more optimistic the results of the comparison. For an mEXOR gate with 10 controls and 10 targets its cost would be 286, which is approximately 1.067 of the cost of correspondent Toffoli gate, and almost 10 (9.37) times better than the strait forward realization.*

4. ASYMPTOTICALLY OPTIMAL SYNTHESIS METHOD

The organization of this section is as follows: describe the synthesis method, then define the Shannon function and prove that its lower boundary has the same cost order as the upper boundary for the cost of the algorithm. This allows us to say that the synthesis algorithm is asymptotically optimal.

Building a cascade is considered to be a standard procedure for creating reversible networks. Initially there are no gates in network. At each step of the algorithm new gates are added to the end of cascade. Note, that if there exists a cascade that realizes the inverse of a reversible function f^{-1}, the reverse order the cascade realizes the function itself. We will need this observation since the way a cascade is constructed produces a network for the inverse function.

#of controls	garbage	Toffoli	2	4	8	Rel 2	Rel 4	Rel 8
2	0	7	9	13	21	1.286	1.857	3
3	0	13	15	19	27	1.154	1.462	2.077
4	0	29	31	35	43	1.069	1.207	1.483
5	0	61	63	67	75	1.033	1.098	1.23
6	0	125	127	131	139	1.016	1.048	1.112
6	4	112	114	118	126	1.018	1.054	1.125
7	0	253	255	259	267	1.008	1.024	1.055
7	3	124	126	130	138	1.016	1.048	1.113
8	0	509	511	515	523	1.004	1.012	1.028
8	4	172	174	178	186	1.012	1.035	1.081

Table 1: Cost comparison.

We assume that the function to be realized is given as the truth table where the left part consists of the inputs and the right part consists of the output patterns. The input patterns are sorted in lexicographical order. The synthesis algorithm proceeds as follows. Build a network for the reverse of f and then read it in reverse order to get a network for f. Do this in a series of steps. At each step a few gates are added to the end of cascade. Each step brings one entity of the output part of the truth table so that it corresponds to the input. This has to be accomplished without changing the entities that were processed during the previous steps. The procedure terminates when each input pattern is equal to the corresponding output pattern. The network built during the process realizes f^{-1}. This algorithm is similar to the one proposed in [9].

Step 0. The top part of the left side of the truth table consists of the input pattern with the lowest order, $(0, 0, ..., 0)$ which represents integer 0 as a binary expression. The corresponding pattern in the output side, $(b_1, b_2, ..., b_n)$ does not necessarily consist of all zero's, therefore it should be brought to the form when it will be equal to the input part. To do so use one mEXOR gate, $TOF(, b_{i_1} + b_{i_2} + ... + b_{i_k})$, where $\{b_{i_1}, b_{i_2}, ..., b_{i_k}\} = \{b_j | b_j = 1, 1 \leq j \leq n\}$.

Step k. The input part of the truth table has pattern $(a_1, a_2, ..., a_n)$, which represents binary expansion of the integer number k. The output part has the pattern $(b_1, b_2, ..., b_n)$ which, in general, differs from $(a_1, a_2, ..., a_n)$. For any Boolean pattern $(x_1, x_2, ..., x_n)$ define the set $X^1 = \{x_j | x_j = 1, 1 \leq j \leq n\}$ - all one bits of pattern $(x_1, x_2, ..., x_n)$. In order to bring $(b_1, b_2, ..., b_n)$ to the form $(a_1, a_2, ..., a_n)$ we need at most two mEXOR gates only:

1. **Increase order.** Apply mEXOR gate $TOF(B^1, B^1 \setminus A^1)$ to bring $(b_1, b_2, ..., b_n)$ to the form $(c_1, c_2, ..., c_n) = (a_1 \vee b_1, a_2 \vee b_2, ..., a_n \vee b_n)$: change the output part of the truth table as dictated by the gate.

2. **Decrease order.** Apply mEXOR gate $TOF(A^1, C^1 \setminus A^1)$ to bring $(c_1, c_2, ..., c_n)$ to the form $(a_1, a_2, ..., a_n)$: change the output part of the truth table as dictated by the gate.

Note, that during this step all the patterns previously put at their places were not altered:

- $(b_1, b_2, ..., b_n) \succeq (a_1, a_2, ..., a_n)$ since all the patterns with the order less than $(a_1, a_2, ..., a_n)$ are already at their correct places in the upper part of the truth table.
- It follows from the definition of $(c_1, c_2, ..., c_n)$, $(c_1, c_2, ..., c_n) \succeq (b_1, b_2, ..., b_n) \succeq (a_1, a_2, ..., a_n) \Rightarrow (c_1, c_2, ..., c_n) \succeq (a_1, a_2, ..., a_n)$.

Step $2^n - 1$. Actually, there are no operations at the last step, since if all of the $2^n - 1$ patterns with lower order are on their places, there is automatically only one spot available for the last pattern, $(1, 1, ..., 1)$.

Complexity analysis. An upper bound on the complexity of presented algorithm output is given by the formula $2^{n+1} - 4$. Since there are 2^n steps to do, and each requires at most 2 gates to be added to the network, it sums up to $2 * 2^n$. A more accurate analysis shows that the first step adds at most one gate, the last step never adds a gate, and the step before last uses at most one gate (symmetrically to the first step), therefore the complexity decreases to $2^{n+1} - 4$. This bound is reachable, so it cannot be any smaller.

EXAMPLE 3. *Take a reversible function given as a truth table (columns* **Input** *and* **Output** *of the Table 3) with the variables named x_1, x_2, x_3, and x_4. Its output is the input with permuted pattern 0011 and 1100.*

The algorithm proceeds as follows:

- **Steps 0-2.** *Since the patterns match, do nothing.*
- **Step 3.** *Input pattern 0011 does not match output pattern 1100.*
  - *Increase order. Apply mEXOR gate $TOF(x_1 + x_2, x_3 + x_4)$ to bring pattern 1100 to the form 1111. The result is shown in column 3I.*
  - *Decrease order. Apply mEXOR gate $TOF(x_3 + x_4, x_1 + x_2)$ to bring pattern 1111 to the desired form of 0011. The result is shown in column 3D.*
- **Steps 4-6.** *Nothing to do there, everything matches.*
- **Step 7.** *Pattern 1011 does not match the desired 0111.*
  - *Increase order. Apply mEXOR gate $TOF(x_1 + x_3 + x_4, x_2)$. The result is shown in 7I.*

Input	Output	3I	3D	7I	7D	11D	12D
0000	0000	0000	0000	0000	0000	0000	0000
0001	0001	0001	0001	0001	0001	0001	0001
0010	0010	0010	0010	0010	0010	0010	0010
0011	**1100**	**1111**	0011	0011	0011	0011	0011
0100	0100	0100	0100	0100	0100	0100	0100
0101	0101	0101	0101	0101	0101	0101	0101
0110	0110	0110	0110	0110	0110	0110	0110
0111	0111	0111	**1011**	**1111**	0111	0111	0111
1000	1000	1000	1000	1000	1000	1000	1000
1001	1001	1001	1001	1001	1001	1001	1001
1010	1010	1010	1010	1010	1010	1010	1010
1011	1011	1011	0111	0111	**1111**	1011	1011
1100	0011	0011	1111	1011	1011	**1111**	1100
1101	1101	1110	1110	1110	1110	1110	1101
1110	1110	1101	1101	1101	1101	1101	1110
1111	1111	1100	1100	1100	1100	1100	1111

Table 2: Circuit building process for a four variable function.

  - *Decrease order. Apply mEXOR gate* $TOF(x_2 + x_3 + x_4, x_1)$ *see column 7D.*

- **Step 11.** *1111 does not match desired 1011. Decrease the order by applying* $TOF(x_1+x_3+x_4, x_2)$*. The result of operation is shwon in 11D.*

- **Step 12.** *Finally, match the last pattern by applying mEXOR gate* $TOF(x_1 + x_2, x_3 + x_4)$*.*

The 6 gates obtained above are shown as a network in Figure 3. Note, the gates are in reverse order.

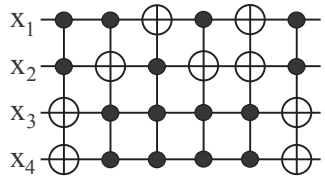

Figure 3: mEXOR gate network.

In quantum technology the cost of Toffoli and mEXOR gates grows quadratically as the number of controls increases and no garbage is allowed and linearly as $48n - 212 (n \geq 7)$ if certain amount of garbage is permitted [1]. It is clearly beneficial to have few controls. In the following we provide a modification of the algorithm which favors smaller gates.

Quantum modification. Update the "increase order" step as follows. Apply mEXOR gate $TOF(D^1, B^1 \setminus A^1)$ to bring $(b_1, b_2, ..., b_n)$ to the form $(c_1, c_2, ..., c_n) = (a_1 \vee b_1, a_2 \vee b_2, ..., a_n \vee b_n)$. Where the set D^1 is defined as follows: $D = \min\{D = (d_1, d_2, ..., d_n) | D^1 \subseteq B^1, (d_1, d_2, ..., d_n) \succeq (a_1, a_2, ..., a_n)\}$.

The above simplification procedure allows us to choose a smaller gate for the "increase order" step. Although it does not necessarily mean that the cost of the network will decrease, but it may decrease if the operation is used wisely. Some other modifications of the algorithm, when not necessarily minimal subsets D are chosen may lead to a simpler networks, but it is hard to specify which set D to choose. Exhaustive search is almost impossible, since the branching factor will be too high, so we propose a heuristics approach for further modifications.

We illustrate the algorithm and its quantum modification with an example where usage of quantum modification is beneficial. Examples where it will not be beneficial also exist. Take function with truth table specified in **Input-Output** columns, with the names of variables a, b, c written left-to-right (see Table 4).

For the basic approach find the first pattern of the output part of truth table which differs from the corresponding input pattern. It is 101. Increase the order by applying $TOF(a + c, b)$ (result is shown in column **BI**). The pattern becomes 111. Decrease the order by applying $TOF(b+c, a)$. Update the information (column **BD1**). Finally, decrease the order of 111 to the desired input pattern 011 by applying $TOF(a + c, b)$.

For the quantum modification select the smaller gate (first step), $TOF(a, b)$ to increase the order of 101 (column **QI**). At the second step we decrease the order of 111 by applying $TOF(b + c, a)$ (result is shown in **QD1** column). Finally, use $TOF(a, b)$ to decrease the order of 110 to match input pattern 100.

Quantum cost analysis: the basic approach uses 3 Toffoli gates, which results in the total cost of 21, whereas the quantum modification has cost 9.

Note, the function given in the above example can be realized with one gate, namely the Fredkin gate.

DEFINITION 4. **Shannon function**, $L(n)$ *is maximal num-*

Input	Output	BI	BD1	BD2	Input	Output	QI	QD1	QD2
000	000	000	000	000	000	000	000	000	000
001	001	001	001	001	001	001	001	001	001
010	010	010	010	010	010	010	010	010	010
011	**101**	**111**	011	011	011	**101**	**111**	011	011
100	100	100	100	100	100	100	110	**110**	100
101	011	011	**111**	101	101	011	011	111	101
110	110	110	110	110	110	110	100	100	110
111	111	101	101	111	111	111	101	101	111

Table 3: Circuit for the basic approach.

ber of gates requires to realize a reversible function of n variables with an optimal network.

We just proved an upper bound $L(n) \leq 2^{n+1} - 4$, which is $L(n) \preceq 2^n$. Prove a lower bound $L(n) \geq C * 2^n$ for a positive constant C.

LEMMA 1. *The number of distinct mEXOR gates with n variables is* $(3^n - 2^n)$.

PROOF. Each of the variables may participate in a gate as control, target or do not be a part of a gate. This gives possibility of 3^n gates. But, among those 3^n some will not contain any target bits, therefore will not be a mEXOR gate. Number of the last will be 2^n (each bit is allowed to be either in control or does not present). Thus, the answer is given by formula $3^n - 2^n$. □

It is interesting to note that an mEXOR gate with zero controls is used at most once in the algorithm (Step 0), but the number of them is exponential, namely 2^n. It happens that considering these gates as a set of NOTs does not lead to a change of asymptotic behavior, which is the focus of this section.

THEOREM 2. $L(n) \geq \frac{2^n}{\ln 3} + \underline{o}(2^n)$.

PROOF. The number of all reversible functions of n variables is $2^n!$ (as the number of permutations of 2^n elements). The total number of mEXOR gates is $3^n - 2^n$. Suppose that by taking all the possible cascades of from mEXOR we get different functions (which is, of course, not true), what will be the complexity of the hardest to realize function. The answer can be given by formula $\log_{(3^n - 2^n)}(2^n!)$, which can be simplified (using Stirling formula) to the form $\frac{2^n}{\ln 3} + \underline{o}(2^n)$. □

5. FUTURE WORK

In the future we want to incorporate the bidirectional approach [9] as well as design a template simplification tool analogous to the one proposed in [9] and [8] in order to be able to produce reasonable networks for the benchmark functions.

6. REFERENCES

[1] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *The American Physical Society*, 1995.

[2] C. Bennett. Logical reversibility of computation. *I.B.M. J. Res. Dev.*, 17:525–532, 1973.

[3] G. W. Dueck and D. Maslov. Reversible function synthesis with minimum garbage outputs. In *6th International Symposium onRepresentations and Methodology of Future Computing Technologies*, March 2003.

[4] R. Feynman. Quantum mechanical computers. *Optic News*, 11:11–20, 1985.

[5] G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel. The microarchitecture of the pentium 4 processor. *Intel Technology Journal*, 2001.

[6] R. Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res.*, 5:183–191, 1961.

[7] D. Maslov and G. W. Dueck. Garbage in reversible design of multiple output functions. In *6th International Symposium on Representations and Methodology of Future Computing Technologies*, March 2003.

[8] D. Maslov, G. W. Dueck, and D. M. Miller. Templates for toffoli network synthesis. Submitted to *International Workshop on Logic Synthesis*, May 2003.

[9] D. M. Miller, G. W. Dueck, and D. Maslov. A transformation based algorithm for reversible logic synthesis. Submitted to *DAC*, June 2003.

[10] A. Mishchenko and M. Perkowski. Logic synthesis of reversible wave cascades. In *International Workshop on Logic Synthesis*, June 2002.

[11] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[12] T. Toffoli. Reversible computing. *Tech memo MIT/LCS/TM-151, MIT Lab for Comp. Sci*, 1980.