Improved ESOP-based Synthesis of Reversible Logic

N. M. Nayeem
Dept. of Math & Computer Science
University of Lethbridge
Lethbridge, AB, Canada
noor.nayeem@uleth.ca

J. E. Rice
Dept. of Math & Computer Science
University of Lethbridge
Lethbridge, AB, Canada
j.rice@uleth.ca

***Abstract*—This paper presents an improved ESOP-based reversible logic synthesis which utilizes cubes shared by multiple outputs and ensures that the implementation of each cube requires just one Toffoli gate. Thus it has the potential of minimizing the gate count and quantum cost. Experimental results show that this technique can reduce the quantum cost up to 75%, compared to the existing algorithm.**

I. Introduction

Landauer [1] showed that traditional logic computation generates a certain amount of heat for every bit of information that is lost or discarded regardless of underlying technology. This dissipated heat will cause problems in the near future if Moore's law holds true. Reversible logic, on the other hand, dissipates no energy theoretically as it does not erase any information during computation. According to Bennett [2], it would be possible for a circuit to dissipate zero energy if it is implemented using reversible gates. Frank [3] states that the amount of dissipated heat in reversible logic will become very close to zero with the development of the technology. It is interesting to note that reversible logic has a direct relationship with quantum computing as all quantum gates are reversible [4]. Moreover, reversible computing has applications in diverse technologies such as ultra-low power CMOS design, optical computing, nanotechnology, and bioinformatics.

Synthesis of reversible logic is far different from synthesis of irreversible logic since fan out and loops are not permitted. As a result, design approaches used for traditional Boolean logic cannot be directly applied to reversible logic. There are a number of reversible logic synthesis techniques such as transformation [5], positive polarity Reed-Muller expressions (PPRM) [6], exclusive-or sum-of-products (ESOP) [7], [8], and shared PPRM [9] techniques. Synthesis based on ESOP representations of functions is of interest because of the easy transformation of ESOP terms into a cascade of Toffoli gates, as well as the ability to handle functions with large numbers of inputs. In this paper, we present an optimized shared cube synthesis approach which also works with the ESOP representation of a function. Comparisons with work in the literature [10], [11] seem to indicate that our approach performs better than the existing methods when ESOP terms are shared by more than two outputs.

II. Background

A. Reversible Logic

A function is reversible if it is bijective (*i.e.*, one-to-one and onto) [12]. In other words, a reversible function has a one-to-one correspondence between its input and output vectors. A reversible gate realizes a reversible function and has the same number of inputs and outputs. A reversible circuit consists of only reversible gates which are interconnected without fanout and feedback [4].

Traditional logic gates such as AND, OR, NAND, NOR and EXOR are not reversible. However, the NOT gate is reversible. The most popular reversible gates are the Toffoli gate and the Fredkin gate. An *n*-bit Toffoli gate maps the input vector [x_1, x_2, ..., x_{n-1}, x_n] to the output vector [x_1, x_2, ..., x_{n-1}, $x_1x_2...x_{n-1} \oplus x_n$] as shown in Figure 1(a). The first (n-1) bits are known as controls and the last bit is the target. 2-bit Toffoli gate is also known as the CNOT gate. In general, the target is affected only if all of the control lines have the value 1; however negative-control Toffoli gates have recently been proposed. The use of negative-control Toffoli gates simplifies a circuit by reducing the number of NOT gates [13], [14]. A 3-bit Toffoli gate with a single negative control in its first input is shown in Figure 1(b).

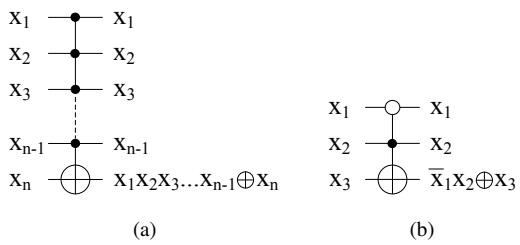


Fig. 1. (a) An *n*-bit Toffoli gate and (b) a 3-bit negative-control Toffoli gate.

Gate count is a popular cost metric used to evaluate reversible circuits. Another common metric is quantum cost. The quantum cost refers to the number of elementary (quantum) gates required to implement the circuit. A common way to calculate the quantum cost of a reversible circuit is to add the quantum costs of its gates. For evaluation, we use the costs of

Toffoli gates given in [15]. The negative-control Toffoli gate with at least one positive control has the same quantum cost as a Toffoli gate. However, if all controls are negative, the negative-control Toffoli gate has an extra cost of 2. For more information, please see [13], [14].

B. Exclusive-or Sum-of-products (ESOP)

In a sum-of-products (SOP) representation of a switching function terms, also referred to as products or cubes, are created by ANDing one or more literal (variable in either its complemented or non-complemeted form). To create a SOP expression one or more of these terms are then combined using OR gates. The exclusive-or sum-of-products (ESOP) representation is slightly different from a SOP as the OR (+) operators are replaced with exclusive-or ($\oplus$) operators. For example, a function $f = xy + yz$ in the SOP form can be rewritten as $f = xy \oplus \overline{x}yz$ which is in an ESOP form. We note that ESOP forms can represent any Boolean functions, and both $+$ and $\oplus$ are associative operators.

III. Previous Work

A brief overview of different ESOP-based methods is given in this section. The basic ESOP-based synthesis proposed in [7] works with the ESOP cube-list and generates a circuit by transforming the ESOP cubes into a cascade of Toffoli gates, where a gate is generated for each cube of each output. The resultant circuit requires 2*n*+*m* lines, where *n* is the number of inputs and m is number of outputs. 2*n* lines are required for the input variables and their negated forms. By observing that all the negated lines are not utilized in all cases, and since it is easy to get a negated line by inserting a NOT gate when necessary, the authors in [7] proposed a heuristic algorithm using the alpha/beta cost metric, which considerably reduces the number of lines to *n*+*m*. Rice and Suen [8] proposed another heuristic based on autocorrelation coefficients.

A newer technique, the shared-cube synthesis algorithm [10], [11] generates one Toffoli gate for each cube irrespective of the number of outputs and adds CNOT gates to transfer the shared functionality to other outputs containing the cube. However, this algorithm may generate multiple Toffoli gates for a single cube if it is shared by more than two outputs. The next section addresses this issue and presents an efficient shared cube synthesis algorithm.

IV. Improved Shared Cube Synthesis

Shared cube synthesis works with multi-output functions if the ESOP terms (cubes) are shared by more than one output. For instance, given a multi-output function, $f_1 = ab \oplus cd$ and $f_2 = abc$, shared cube synthesis cannot improve the circuit as there is no shared term between f_1 and f_2. The algorithm presented in [10], [11] takes the best advantage of shared functionality if the ESOP terms are shared by only two outputs. However, if the shared terms exist in more than two outputs, transformation of each term may require more than one Toffoli gate, which is inefficient. The following two examples show that the existing algorithm can be further optimized. The optimized shared cube synthesis presented in this paper produces one Toffoli gate for a cube and hence has the potentiality to reduce the gate count as well as quantum cost.

Example 1: Given a cube-list of 3-input 3-output function shown in Figure 2(a), a Toffoli cascade generated by the algorithm from [10], [11] is shown in Figure 2(b). This cascade requires two Toffoli gates for each of the cubes. An equivalent network depicted in Figure 2(c) generates one Toffoli gate for each cube, and hence minimizes the gate count by 4. Moreover, the quantum cost is reduced from 66 to 34.

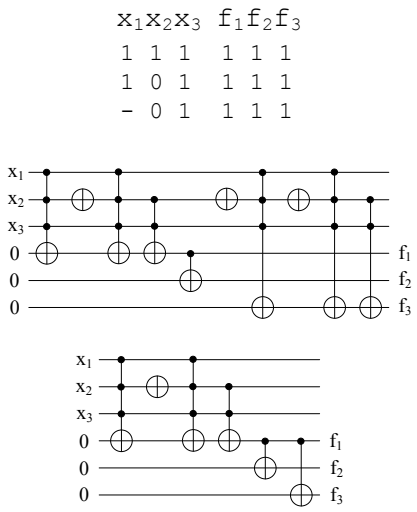


Fig. 2. (a) An example cube-list, (b) the Toffoli cascade generated by the algorithm in [10], [11], and (c) an improved Toffoli cascade.

Example 2: Consider the cube-list given in Figure 3(a). The algorithm described in [10], [11] generates a Toffoli network containing three Toffoli gates for the first cube and two Toffoli gates for the second cube, a total of 8 gates as shown in Figure 3(b). However, an efficient synthesis optimizes the network as shown in Figure 3(c). The quantum cost of this network is 27, in contrast to the former approach which costs 56. This example also shows an efficient way to make use of the shared functionality even if the cubes are not shared by all the outputs.

A. Our Approach

Some cubes are not shared by multiple outputs. If the number of 1s in the output part of a cube is one, then only one output contains this cube and no other output shares it. This cube, called an ungrouped cube, will be dealt with individually. The optimized synthesis technique proposed in this paper consists of the following two phases:

Phase 1: Generation of sub-lists
Phase 2: Transformation of sub-lists into gate-lists

Generation of sub-lists: This phase takes the original *cube-list* as its input and generates sub-lists as follows:

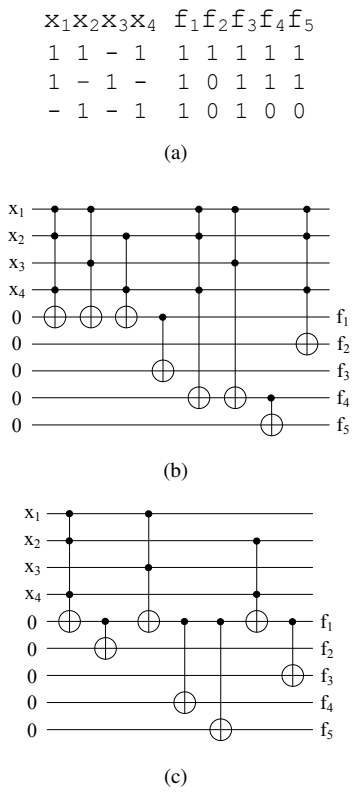


Fig. 3. (a) An initial cube-list, (b) the Toffoli cascade generated by the algorithm in [10], [11], and (c) an improved Toffoli cascade.

Step 1: Move ungrouped cubes from the *cube-list* into the *ungrouped-list*.

Step 2: Repeat Step 3 and Step 4 until *cube-list* is empty. Initialize the value of k with a 1 and increase its value by 1 after each iteration.

Step 3: Select a cube from the modified *cube-list* which is shared by the largest number of functions, *i.e.* has the maximum number of 1s in its output part. This cube and every other cube with an identical output part are moved to the *sub-list*$_k$.

Step 4: Select a cube from the *cube-list* which has the maximum number of 1s and which must have 0 at the output position at which the cubes in *sub-list*$_k$ have 0. In other words, the outputs that share this cube must also share all the cubes in *sub-list*$_k$. Afterwards this cube along with the cubes having identical output parts is moved from the *cube-list* to *sub-list*$_k$. This step continues until no such cube exists in the *cube-list*.

Transformation of sub-lists into gate-lists: In this phase, the sub-lists generated by phase 1 are transformed into a cascade of Toffoli gates. The *total-gate-list*, which is initially empty, will contain the final circuit at the end of this phase.

Step 1: For each *sub-list*$_k$, do Step 2 - Step 6.

Step 2: An output line p is selected as the Toffoli target line if the corresponding output contains all cubes in *sub-list*$_k$. If multiple such lines are found, choose one line arbitrarily that has not yet been used as a control or target of any Toffoli gate. If all such lines are occupied by other gates, choose the same line targeted in the last iteration, or any line arbitrarily.

Step 3: The *gate-list*$_k$ is initially empty. For each of the cubes in *sub-list*$_k$ perform steps 4-5 which add gates to the gate-list$_k$.

Step 4: Add a Toffoli gate that has a target on line p. The controls of this Toffoli are the input lines for which the input part of the corresponding cube contains zeros and ones. If the input part contains at least one zero, use a negative-control Toffoli gate. After that add a CNOT gate to transfer the gates to other output line(s) only if this cube is the last cube in the list that the output contains.

Step 5: If the line p has already hosted gates before the beginning of Step 2, adding CNOTs in Step 4 transfers those gates to other outputs as well. To remove this unwanted effect, also add CNOTs at the beginning of the *gate-list*$_k$. Note that insertion of this gate may cancel out another CNOT in the *total-gate-list*. If so, remove both of these gates.

Step 6: Append the *gate-list*$_k$ at the end of *total-gate-list*.

Step 7: Generate one Toffoli for each cube in *ungrouped-list* and append the gates to *total-gate-list*.

Example 3: An ESOP *cube-list* of six cubes with four input variables (x_1, x_2, x_3, and x_4) and five output variables (f_1, f_2, f_3, f_4, and f_5) is shown in Figure 4(a). The cubes are labeled C_1 to C_6. Among all the cubes only C_1 is ungrouped since the number of 1s in its output portion is 1 and so it is therefore separated from the *cube-list*. The resultant lists are shown in Figure 4(b). Now in the modified *cube-list*, C_3 has the highest number of 1s in its output part. Thus it is moved to the *sub-list*$_1$. Note that C_3 is not shared by f_5. Now from the remaining cubes (C_2, C_4, C_5, and C_6), a cube is selected whose output portion contains the highest number of 1s and which is not shared by output f_5 since f_5 does not contain C_3. Although C_2, C_5, and C_6 have the same number of 1s in their output parts, C_6 is not allowed to move in this iteration since it is shared by f_5. Between the cubes C_2 and C_5, suppose that C_2 has been selected. C_2 along with C_5 is moved to the end of *sub-list*$_1$ since the output patterns of these two cubes are identical. There are no other cubes which can be moved to *sub-list*$_1$. Figure 4(c) shows the cubes in *sub-list*$_1$ and *cube-list*.

Now the current *cube-list* consists of C_4 and C_6. Both cubes have the same number of output ones. Consider that C_4 is chosen and moved to *sub-list*$_2$. We see that f_1 shares C_6 but not C_4. As a result, C_6 is not allowed to make a group with C_4. Figure 4(d) shows the *sub-list*$_2$. Now only one cube C_6 is remaining in the *cube-list*; thus in the next iteration moving this cube to *sub-list*$_3$ shown in Figure 4(e) completes the phase 1.

In phase 2, we transform three sub-lists and ungrouped-list into a cascade of Toffoli gates. For *sub-list*$_1$ shown in Figure 4(c), outputs f_1, f_2, and f_3 have all the cubes in this list. Moreover, there are no gates on any of these output lines. Consequently, any of these lines can be used as a target line. Let f_1 be chosen as the target line. A Toffoli gate for C_3 targeting at f_1 is generated. Since f_4 does not share any cube other than C_3 in *sub-list*$_1$, one CNOT is required to transfer

	$x_1x_2x_3x_4$	$f_1f_2f_3f_4f_5$
C_1	1 0 - 0	0 0 0 1 0
C_2	1 1 - -	1 1 1 0 0
C_3	1 1 1 -	1 1 1 1 0
C_4	1 0 0 1	0 1 0 0 1
C_5	1 - 1 1	1 1 1 0 0
C_6	1 - 0 -	1 0 1 0 1

Cube-list

(a) An initial cube-list.

	$x_1x_2x_3x_4$	$f_1f_2f_3f_4f_5$
C_1	1 0 - 0	0 0 0 1 0

Ungrouped-list

	$x_1x_2x_3x_4$	$f_1f_2f_3f_4f_5$
C_2	1 1 - -	1 1 1 0 0
C_3	1 1 1 -	1 1 1 1 0
C_4	1 0 0 1	0 1 0 0 1
C_5	1 - 1 1	1 1 1 0 0
C_6	1 - 0 -	1 0 1 0 1

Current Cube-list

(b) Separation of ungrouped cubes from the cube-list.

	$x_1x_2x_3x_4$	$f_1f_2f_3f_4f_5$
C_3	1 1 1 -	1 1 1 1 0
C_2	1 1 - -	1 1 1 0 0
C_5	1 - 1 1	1 1 1 0 0

Sub-list$_1$

	$x_1x_2x_3x_4$	$f_1f_2f_3f_4f_5$
C_4	1 0 0 1	0 1 0 0 1
C_6	1 - 0 -	1 0 1 0 1

Current Cube-list

(c) Generation of sub-list$_1$.

	$x_1x_2x_3x_4$	$f_1f_2f_3f_4f_5$
C_4	1 0 0 1	0 1 0 0 1

Sub-list$_2$

	$x_1x_2x_3x_4$	$f_1f_2f_3f_4f_5$
C_6	1 - 0 -	1 0 1 0 1

Current Cube-list

(d) Generation of sub-list$_2$.

	$x_1x_2x_3x_4$	$f_1f_2f_3f_4f_5$
C_6	1 - 0 -	1 0 1 0 1

Sub-list$_3$

(e) Generation of sub-list$_3$.

Fig. 4. Cube-list and its sub-lists

C_3 from f_1 to f_4. Next two Toffoli gates are generated for C_2 and C_5. Again, to transfer all the gates from f_1 to f_2 and f_3, two CNOTs are added. The Toffoli cascade for *sub-list*$_1$ is shown in Figure 5(a).

The *sub-list*$_2$, shown in Figure 4(d), has just one cube C_4 which is shared by f_2 and f_5. From Figure 5(a) we see that f_2 is already occupied by another gate. Since the line f_5 is empty, this is chosen as the target line for *sub-list*$_2$. One negative-control Toffoli gate is added for C_4, which is transferred to f_2 via a CNOT. Gates generated for this list are appended at the end of the cascade in Figure 5(a), which results in the circuit shown in Figure 5(b).

Next we consider the *sub-list*$_3$ in Figure 4(e), which contains one cube C_6. Outputs f_1, f_3, and f_5 share C_6, and the corresponding output lines are not empty (see Figure 5(b)). Let the target line be f_1. Since f_1 has gates on it, in order to eliminate the effect of these unexpected gates while transferring C_6 to f_3 and f_5, two more CNOTs are needed before generating the Toffoli gate for C_6. However, adding a new CNOT from f_1 to f_3 will cancel out previously the inserted CNOT (from f_1 to f_3) in the circuit shown in Figure 5(b). Therefore, this CNOT is removed rather than adding a new one (from f_1 to f_3). However a CNOT from f_1 to f_5 is required. Afterwards one negative-control Toffoli gate for C_6 and two CNOTs for sharing with f_3 and f_5 are added as shown in Figure 5(c). Finally, the ungrouped cube C_1 shown in Figure 4(b) is transformed directly into a negative-control Toffoli gate. The final cascade is shown in Figure 5(d).

V. Experimental Results

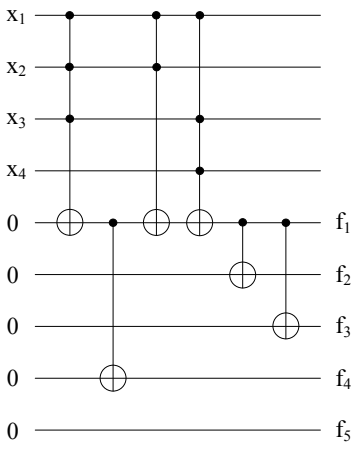
The proposed algorithm has been developed in C++. As is done in the original shared cube synthesis approach from [10], [11], the tool EXORCISM-4 [16] is used to generate the ESOP cube-lists for the benchmark circuits. The implemented program has been run on a 2.4GHz Intel core 2 duo based system with 4GB RAM using the same set of benchmark circuits (except dalu as it is not available) reported in [10], [11]. The execution time of the program is negligible, and the results of this experiment are compared to the existing algorithm [10], [11] in Table I.

In Table I GC and QC stand for gate count and quantum cost, respectively. In the first column, the name of the function is given. Columns two to five and columns six to nine show the number of NOT gates, number of Toffoli gates (excluding the NOT gates), total number of gates and the quantum cost of the circuit generated by the existing algorithm and the proposed algorithm, respectively. The last two columns indicate the improvement in percentage of the proposed algorithm over the existing one. Negative values indicate that the existing algorithm is better than the proposed one for that function. We note that for fair comparison, Toffoli gate count (not total gate count) and their quantum costs (excluding the extra cost of negative-control Toffoli gate if applicable) are considered during the calculation of improvement.

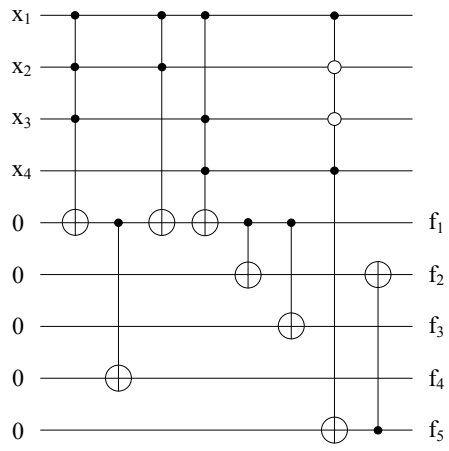
We also note that it appears that the calculation of quantum cost in [10], [11] may be incorrect for cordic.esop and chkn.esop (indicated by * in Table I). Shared cube synthesis requires at least one Toffoli gate for each cube. If we just consider one Toffoli for a cube and do not count the necessary CNOT gates, then quantum costs required for cordic.esop and chkn.esop are at least 187,563 and 33,620 respectively, which are greater than the values reported in [10], [11].

It can be seen from the improvement column of Table I that our approach reduces the quantum cost of Toffoli gates for all functions except the last one (z4ml). Circuits bw and apex3 are improved by more than 75% in terms of quantum cost. Moreover, a significant improvement is noticed for the functions risc, bc0, and in2. The table also shows that the average reduction of quantum cost is 25%.

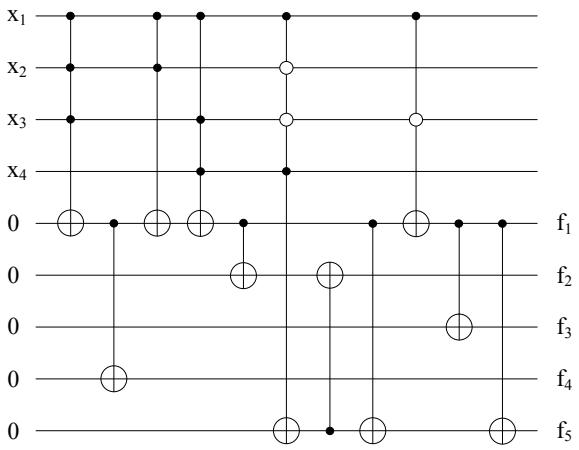
However, the opposite trend is found in the improvement column of the Toffoli gate count. We have investigated to find the reason. The algorithm in [11] is somewhat different from [10] since the former utilizes the shared functionality until one of the output lines is empty, while the latter algorithm inserts CNOT gates to remove the impact of other gates when the output lines are not empty. It is expected that the insertion of necessary CNOT gates will increase the Toffoli gate count as well as the total gate count, but the values (*e*.g., the number



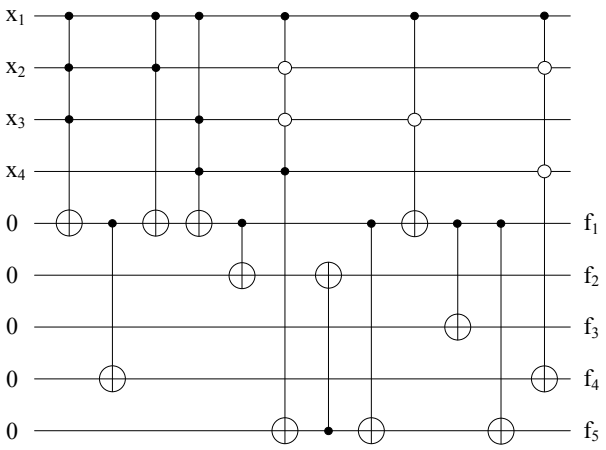
(a) A circuit equivalent to sub-list$_1$.



(b) Appending the circuit for sub-list$_2$ to the circuit of Figure 5(a).



(c) Appending the circuit for sub-list$_3$ to the circuit of Figure 5(b).



(d) A circuit equivalent to the cube-list in Figure 4(a).

Fig. 5. Improved shared cube synthesis process.

of Toffoli gates and total gate count) reported in [11] and [10] are identical. In addition, the problem found in comparing quantum costs also exists in gate counts. For example, the cube-list of the function term1 contains 540 cubes. Thus the Toffoli circuit must generate at least 539 Toffoli gates (not 540 because one cube contains all don't care values, which will be transformed into NOT gates) even if the CNOT gates, which are required to transfer the gates to other outputs and remove the effect of other gates, are not taken into account. However, the number of Toffoli gates reported in [10] and [11] is 506. Because of these problems, we have difficulties with an accurate comparison of the gate count metric between our approach and the results as reported in [10], [11].

VI. Conclusion

In this paper we have proposed an improved shared cube synthesis algorithm. Our experimental results for the quantum cost of the resulting circuits are compared to those reported in [10], [11]. Our new algorithm results in improvement in quantum costs for nearly all circuits, although we note that we did not have similar improvements when considering the gate counts. As indicated in Section V we are still investigating why this is. We have thus focused our analysis on the quantum costs of the circuits generated by our approach, although future work will include further investigation and discussion with a view to more accurate comparisons.

Acknowledgment

This research was funded by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, pp. 183–191, 1961.

[2] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[3] M. P. Frank, "Introduction to reversible computing: motivation, progress, and challenges," in *Proceedings of the 2nd Conference on Computing Frontiers*, (Ischia, Italy), pp. 385–390, 4-6 May 2005.

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

[5] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proceedings of the 40th annual Design Automation Conference (DAC)*, pp. 318–323, 2003.

[6] P. Gupta, A. Agrawal, and N. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2317–2330, 2006.

[7] K. Fazel, M. Thornton, and J. E. Rice, "ESOP-based Toffoli gate cascade generation," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, (Victoria, BC, Canada), pp. 206–209, 22-24 Aug. 2007.

[8] J. E. Rice and V. Suen, "Using autocorrelation coefficient-based cost functions in ESOP-based Toffoli gate cascade generation," in *Proceedings of 23rd Canadian Conference on Electrical and Computer Engineering (CCECE)*, (Calgary, Canada), May 2010.

[9] Y. Sanaee, M. Saeedi, and M. S. Zamani, "Shared-PPRM: A memory-efficient representation for Boolean reversible functions," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, (Washington, DC, USA), pp. 471–474, 2008.

[10] Y. Sanaee, "Generating Toffoli networks from ESOP expressions," Master's thesis, University of New Brunswick, 2010.

[11] Y. Sanaee and G. W. Dueck, "Generating Toffoli networks from ESOP expressions," in *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, (Victoria, BC, Canada), pp. 715–719, 13-18 June 2009.

[12] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 710–722, 2003.

[13] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, "Quantum circuit simplification and level compaction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 3, pp. 436–444, 2008.

[14] M. Arabzadeh, M. Saeedi, and M. Zamani, "Rule-based optimization of reversible circuits," in *Proceedings of Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 849–854, 2010.

[15] D. Maslov, "Reversible logic synthesis benchmarks page." `http://www.cs.uvic.ca/~dmaslov/`.

[16] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive sum-of-products," in *Proceedings of the 5th International Reed-Muller Workshop*, (Starkville, Mississippi), pp. 242–250, August 2001.

TABLE I
EXPERIMENTAL RESULTS

Function	Previous algorithm [10], [11]				Proposed algorithm				Improvement (%)	
	NOT GC	Toffoli GC	Total GC	Total QC	NOT GC	Toffoli GC	Total GC	Total QC	Toffoli GC	Toffoli QC
bw	118	175	293	2942	0	322	322	676	-84	76.42
xor5	2	5	7	7	0	5	5	9	0	0
5xp1	48	52	100	1053	0	54	54	782	-3.85	22.59
risc	64	95	159	1564	6	141	147	758	-48.42	50.67
cordic	2361	776	3137	176868*	1	776	777	187620	0	-
vg2	430	199	629	20256	0	192	192	19576	3.52	1.26
bc0	488	401	889	31874	0	444	444	14092	-10.72	55.11
in7	68	26	94	3096	4	53	57	2589	-103.85	14.76
chkn	359	146	505	3351*	1	149	150	33628	-2.05	-
term1	109	506	615	61827	2	549	551	57355	-8.5	7.08
in2	359	202	561	17174	0	182	182	9530	9.9	43.32
apex3	1819	1433	3252	80878	0	2969	2969	18718	-107.19	76.33
e64	96	129	225	29655	0	144	144	24402	-11.63	17.45
example2	183	255	438	12045	4	273	277	10169	-7.06	14.36
x4	210	404	614	18502	6	439	445	14999	-8.66	18.27
apex5	422	513	935	40119	13	552	565	33766	-7.6	14.99
apex6 (x3)	355	524	879	23140	4	575	579	21776	-9.73	4.86
sqr6	39	55	94	797	0	59	59	611	-7.27	19.92
misex2	61	46	107	1986	1	48	49	1470	-4.35	24.21
z4ml	26	32	58	592	0	33	33	567	-3.13	-0.18
Average	380.85	298.7	679.55	19305.94*	2.1	397.95	400.05	22654.65	-20.73	25.63

*Incorrect?