Please note: The methods introduced in this paper are availabe at www.revkit.org.

Reducing Reversible Circuit Cost by Adding Lines

D. Michael Miller
Department of Computer Science
University of Victoria
Victoria, BC, Canada
mmiller@uvic.ca

Robert Wille Rolf Drechsler
Institute of Computer Science
University of Bremen
Bremen, Germany
{rwille,drechsle}@informatik.uni-bremen.de

***Abstract*—Additional lines are required to implement an irreversible function as a reversible circuit. The emphasis, particularly in automated synthesis methods, has been on using the minimal number of additional lines. In this paper, we show that circuit cost reductions can be achieved by adding additional lines. We present an algorithm for line addition that can be targeted to reducing the quantum cost of a circuit or the transistor count for a CMOS implementation. Experimental results show that the cost reduction can be significant even if (1) only a small number of lines (even one) is added and (2) other circuit optimizations have already been applied.**

I. Introduction

The amazing developments in microelectronic technology and the looming limitations are well known and well documented. Both, transistor density and power dissipation are crucial issues for designing current high performance digital circuits. For these reasons, researchers expect that in 10-20 years current technologies will reach their limits (see e.g. [1]). It is thus important to consider alternative technologies. Reversible logic offers one such alternative due to its applications in the domains of low-power design, quantum computation, and bio-computing.

Landauer and Bennett showed in [2], [3], that power dissipation is not only caused by non-ideal behavior of transistors and materials, but also by a more fundamental reason. Each time information is erased (e.g. when the irreversible AND operation is computed), energy is dissipated as well. Only if computation is information-lossless, can zero energy dissipation be achieved. This holds for reversible logic since only bijective operations are allowed (i.e. each input pattern is mapped to a unique output pattern). Practical reversible circuits have been built that exploit these properties [4]. In fact, these circuits were powered by their input signals only and did not need additional power lines.

Quantum computation is of interest, since it potentially allows exponential speed-up of computation for many important problems. This is because qubits, in contrast to traditional bits, can assume a probabilistic superposition of the base states 0 and 1. As a result, a set of qubits can represent multiple states at the same time enabling enormous computational speed-up. Quantum circuits have been developed working with 8, 16, and, recently, 28 qubits (see e.g. [5]). Reversible logic acts as a framework for quantum computation since all quantum gate operations are inherently reversible.

For the reasons cited above, reversible circuits and logic synthesis have in recent years become well studied topics. Many synthesis approaches for reversible logic have been proposed (e.g. [6]–[15]). Except for very small cases, these methods do not produce minimal results. Thus, post-synthesis optimization is applied to reduce the cost of a circuit.

For example, template matching [11], [16], [17] is a search method which looks for gate sequences that can be replaced by alternative cascades of lower cost. The run-time increases with both the number of applied templates and the number of gates in the circuit and as a result can be quite high. But for many circuits substantial improvement are achieved.

As a second example, Zhong and Muzio [18] showed how analyzing cross-point faults can identify redundant control connections in reversible circuits. Removing such control lines reduces the cost of the circuit. However, the computation needed to determine such redundancies is extremely high.

In this paper, we propose a post-synthesis optimization technique which reduces the cost of the circuit by adding further signal lines to the circuit. The general idea is to use the new lines for "buffering" factors of gate control lines so that they can be reused by the other gates in the circuit. This reduces the size of these gates and thus decreases the cost of the circuit. Both, quantum cost (used in quantum circuits) and transistor count (used in CMOS implementations) are considered.

A fast algorithm is presented along with results that show it can be quite effective even if only a small number of lines, even 1, are added. Since we consider the addition of lines to a circuit (not to the functional specification), our approach is applicable to circuits designed by any automated synthesis method and in fact to circuits designed by hand.

Experiments show that applying our approach circuit cost can be reduced by up to 70% with adding only a single line. Thus, even for quantum realizations (where qubits and hence the number of circuit lines are limited) adding a further line is beneficial as it reduces the quantum cost significantly. So, the designer can trade off the additional expenses of a further circuit line for a significant reduction of quantum cost. For CMOS realizations, the observed reductions are more moderate. However, here the cost of adding a new line to the circuit is negligible, so that the proposed optimization is worthwhile for such circuits as well.

In previous work, it has already been observed that more circuit lines usually lead to lower (quantum) cost (see e.g. [19] or recently [20]). Moreover, the authors of [21] even showed that some functions cannot be synthesized for certain gate libraries unless one additional line is added. However, in this paper these observations are exploited for the first time by proposing a constructive post-synthesis optimization approach for reversible logic.

The remainder of this paper is structured as follows: Section II provides the necessary background on reversible circuits for this paper. The general idea of the proposed approach is introduced in Section III while Section IV describes the algorithm in detail. Experimental results are reported in Section V and the paper concludes with observations and suggestions for further research in Section VI.

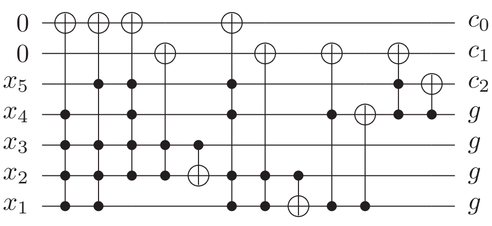


Fig. 1. A Reversible Circuit Realizing RD53.

TABLE I
MCT GATE QUANTUM COST.

m	$(n-m) \geq$	**cost**	m	$(n-m) \geq$	**cost**
1		1	8	5	62
2		1	8	1	100
3		5	8	0	253
4		13	9	6	74
5	2	26	9	1	128
5	0	29	9	0	509
6	3	38	10	7	86
6	1	52	10	1	152
6	0	61	10	0	1021
7	4	50	> 10	$m-3$	$12m-34$
7	1	80	> 10	1	$24m-88$
7	0	125	> 10	0	2^m-3

II. Reversible Logic Circuits

In this section, we provide the background on reversible circuits necessary to make this paper self-contained. Readers interested in a more extensive introduction to the subject should consult the literature, e.g. [22].

Definition 1. *A multiple-output Boolean function $B^n \rightarrow B^n$ is* ***reversible*** *if it maps each input pattern to a unique output pattern. Otherwise, the function is termed* ***irreversible****.*

A reversible function can be realized by a circuit $G = G_1 G_2 \dots G_k$ comprised of a cascade of reversible gates G_i with no fan-out or feedback [22]. A reversible gate, itself realizes a reversible function. Many reversible gates have been proposed [22]. In this paper, we concentrate on multiple control Toffoli and multiple control Fredkin gates which are defined as follows:

Definition 2. *A* **multiple control Toffoli gate** *(MCT) with* **target line** *x_j and* **control lines** *$\{x_{i_1}, x_{i_2}, \dots, x_{i_k}\}$, maps $(x_1 x_2 \dots x_j \dots x_n)$ to $(x_1 x_2 \dots (x_{i_1} x_{i_2} \dots x_{i_k}) \oplus x_j \dots x_n)$. Note that all control lines must be 1 for the target to be inverted and an MCT gate is thus a controlled inversion of the target line. An MCT gate with no control always inverts the target line and is the well-known* **NOT gate***. An MCT gate with a single control line is called a* **controlled-NOT (CNOT) gate***. The case of two control lines is the original gate defined by Toffoli.*

Definition 3. *A* **multiple control Fredkin gate** *(MCF) with* **target lines** *x_p and x_q, and* **control lines** *$\{x_{i_1}, x_{i_2}, \dots, x_{i_k}\}$, maps $(x_1 x_2 \dots x_p \dots x_q \dots x_n)$ to $(x_1 x_2 \dots x_q \dots x_p \dots x_n)$ if all the control lines have value 1. An MCF gate is thus a controlled swap of the target lines. An MCF gate with no control always swaps the target lines. The case of a single control line is the original gate defined by Fredkin.*

An MCT gate is denoted $MCT(C;t)$ where C is the possibly empty set of control lines and t is the target line. An MCF gate is denoted $MCF(C;t_p,t_q)$ where C is the possibly empty set of control lines and t_p and t_q are the target lines. Note that the control lines and unconnected lines pass through both types of gate unchanged. For drawing circuits, we follow the established convention of using the symbol $\oplus$ to denote the target of an MCT gate, $\times$ to denote the targets of an MCF gate, and solid black circles to indicate control connections.

Example 1. *The circuit in Figure 1 realizes the irreversible benchmark function RD53 (taken from RevLib [23]) as a reversible circuit. Note that two constant input and four garbage outputs are required.*

The number of gates in a cascade is a very poor measure of its complexity since the costs of MCT and MCF gates depend on the number of control lines and the target technology. Thus, we use two distinct cost models in this paper: transistor cost and quantum cost. While the transistor cost model estimates the cost of the circuit in terms of the number of CMOS transistors, the quantum cost model estimates the cost of the circuit in terms of the number of elementary quantum gates. More formally:

Definition 4. ***Transistor cost model:*** *The transistor cost of an MCT or MCF gate is $8 \times m$ where m is the number of control lines in the gate [24]. The transistor cost of a circuit composed of MCT and MCF gates is the sum of the transistor costs of the individual gates.*

Definition 5. ***Quantum cost model:*** *The quantum cost of an MCT gate is given in Table I*[1] *where m is the number of control and target lines for the gate and n is the number of circuit lines. The quantum cost of a circuit is the sum of the quantum costs of the individual gates. The quantum cost of an MCF gate with m control and target lines is calculated as the cost for an MCT gate with that number of lines plus 2.*

Even if these models provide better cost estimates than simple gate count they are still only approximations. In particular, they do not take into account transistor or elementary quantum operation reductions that can be made by combining the realizations of MCT and MCF gates that are adjacent, or can be moved to be adjacent, in the circuit. Achieving such reductions in a systematic manner is a complex optimization problem that we leave for future research. Instead we use the well-established cost metrics defined above.

Quite often, the objective in synthesizing a reversible circuit is not only to realize a reversible but also an irreversible Boolean function. This requires the irreversible function to be embedded into a reversible one which requires the addition of constant inputs and garbage outputs defined as follows:

Definition 6. *A* ***constant input*** *to a reversible circuit is one that is set to a fixed value to achieve the desired functionality.*

Definition 7. *A* ***garbage output*** *from a reversible circuit is one which is a don't-care for all possible input conditions.*

[1]Using the calculations introduced in [19] and further optimized in [17] and [25].

In [25], it was shown that at least $g = \lceil log_2(\mu) \rceil$ garbage outputs are required to embed a completely-specified irreversible function into a reversible function, where μ is the maximum number of times a single output pattern is repeated in the truth table of the irreversible function. Thus, an n-input, m-output irreversible function has a total of $m + g$ outputs, $m+g \geq n$. This requires the addition of $c \geq 0$ inputs such that $n+c = m+g$. The new lines must be assigned constant input values. The interest here is what circuit cost savings can be achieved if more than the minimal number of lines are added to a circuit.

III. General Idea

Most reversible synthesis approaches produce circuits using the minimal number of signal lines which is equal to $(n+c) = (m+g)$, i.e. the number of inputs and outputs of the reversible function (reversible embedding) that is realized. Optimization approaches, such as the two noted in Section 1, have also concentrated on using the minimal number of lines. In this section we show how extending the circuit by additional signal lines can improve the cost of a reversible circuit. Hence, the additionally added line is denoted as a *helper line* in the following.

Definition 8. *Let G be a reversible circuit. A* helper line[2] *is an additionally added circuit line*

- *whose input is set to a constant value 0 and*
- *whose output is used as a garbage output.*

Having a helper line available, values can be "buffered" on this line so that they can be later reused by other gates. In doing so, control lines can be saved as shown by the following definition.

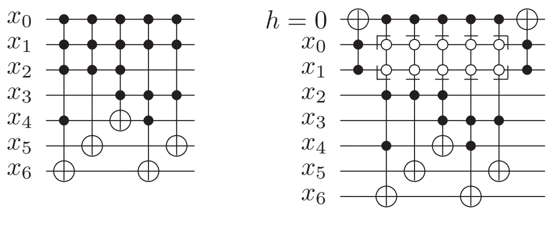
Definition 9. *Let G be a reversible circuit and h be a helper line. Then, a gate $MCT(C, t)$ of G can be replaced by the sequence $MCT(F, h)$, $MCT(h \cup \hat{C}, t)$, $MCT(F, h)$ where $C = F \cup \hat{C}, F \cap \hat{C} = \emptyset$, and $F \neq \emptyset$. In the following this replacement is referred as* factoring *the initial gate, and F is a* factor *of $MCT(C, t)$.*

The terminology "factoring" and "factor" are natural since partitioning the control set C into F and $\hat{C}$ is essentially factoring the AND function for the control lines. This factoring depends on the fact that $0 \oplus x_1 x_2 \dots x_k = x_1 x_2 \dots x_k$, i.e. that the result of a factor can be "buffered" by a constant line assigned to 0.

By applying Definition 9 to gates in a circuit, control lines can be removed. Since the number of control lines directly influences the circuit cost, this may lead to less costly circuits. However, this is only the case, if the total cost of the added gates is less than the cost saved by factoring the control lines. By substituting in a single gate only, this can not happen for the transistor count cost model but it can for the quantum cost model. If more than one gate can be substituted, higher cost savings are achieved (reductions for the transistor cost model are also observed).

These ideas are clarified in the following example.

[2]Note that the helper line works as an ancillary line. However, since the term ancillary line is already used in other contexts, we use the term helper line for clarity.



(a) Original Circuit. (b) Factored Circuit.

Fig. 2. Example of Control Factoring.

Example 2. *Consider the cascade of Toffoli gates depicted in Fig. 2 (a). The gates in this cascade have a common control factor $F = \{x_0, x_1\}$. Hence, the cost of this circuit can be reduced as shown in Fig. 2 (b) by adding an additional line h (at the top of the circuit) as well as the Toffoli gates $TOF(F, h)$ before and after the cascade. This leads to additional quantum cost of $2 \times 5 = 10$. However, the factored gates reuse the result of F (dashed rectangle in Fig. 2 (b)) leading to a reduction of one control line per gate. The removed control lines are shown as white circles. In total this reduces the quantum cost from* 104 *to* 59 *and the transistor count from 144 to 136, respectively.*

Note that the added line is set to constant input 0. Furthermore, the right most Toffoli gate operating on the added line is only needed if the line is to be used for a subsequent factor.

IV. Algorithm

Based on the ideas presented in the last section, we now propose an algorithm that adds one helper line and then employs a straightforward search procedure to use that line for optimizing the circuit. More precisely, we show how to extract factors from Toffoli and Fredkin gates in the circuit (the circuit may contain other types of gates). The algorithm can be applied repeatedly to add more than one helper line. It can also be iterated to add lines until adding a further line results in no cost reduction. The transistor cost model or the quantum cost model can be used and in fact the algorithm is readily adapted to any other gate-based cost model.

Algorithm 1. ***Reversible Circuit Factoring*** *Consider a reversible circuit consisting of the cascade of gates $G_1 G_2 \dots G_k$. Let C_i denote the set of control lines for G_i and let T_i denote the set of target lines for G_i.*

1) *Add a single helper line h.*
2) *Find the highest cost reducing factor across the circuit as follows:*
   For $1 \leq i \leq k$
   If G_i is an MCT or MCF gate and the helper line h is available, i.e. it is not being used by a previously applied factor at this point in the circuit: For every partitioning of C_i into $\{F, \hat{C}\}$ with F not empty
   a) *Find the lowest $j \geq i$ such that $j = k$ or $(F \cap (T_{j+1} \cup h)) \neq \emptyset$, i.e. find the next gate G_j that manipulates one of the lines in F so that the value of the helper line cannot be reused any longer. If the outputs of the circuit are reached use G_k instead.*

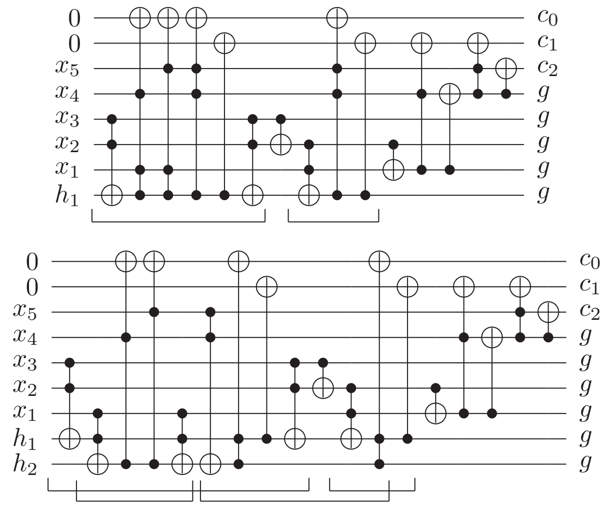


Fig. 3. Reversible Circuits for RD53 with 1 Helper Line and 2 Helper Lines.

b) *Determine the cost reduction that would result from applying this factor to all applicable gates between G_i and G_j, including the cost of introducing two instances of the factor gate $MCT(F, h)$.*

c) *Keep a record of the factor and the gate range that leads to the greatest cost reduction.*

3) *If no cost reducing factor is found in 2. terminate.*
4) *Otherwise, apply the best factor found and repeat from step (2) on the revised circuit.*

Note that, as already mentioned above the rightmost $MCT(F, h)$ operating on the helper line is only added if the helper line is going to be used for another factor.

Example 3. *Figure 3 shows the result of applying our algorithm to the realization of RD53 from Figure 1 using the quantum cost metric. The applied factors are highlighted by brackets at the bottom of the figures. While the original circuit has quantum cost of 128, that can be reduced with 1 helper line to 83 (top circuit) or with 2 helper lines to 66 (bottom circuit). Adding a third helper line does not reduce the quantum cost of this circuit further.*

The order in which factors are considered typically has an effect. We apply the algorithm to the circuit as given and then to the circuit found by reversing the order of the original circuit. The better of the two final circuits is taken as the result. Thus, the presented algorithm is a heuristic. But as the experiments in the next section show, even this simple approach leads to good results.

V. Experimental Results

This section provides experimental results for the proposed approach. To this end, the method described above has been implemented in C and applied to all benchmarks from the RevLib reversible logic benchmark website (www.revlib.org) [23] as of July, 2009. Our experiments were run on an AMD Athlon 3500+ with 1 GB of memory. The QMDD-based circuit equivalence checking method from [26] was used to verify all results.

Since some of the circuits in RevLib have already been optimized using various approaches (e.g. extensive template post-synthesis optimization, output permutation optimization, and other techniques), to provide a more even staring point we pre-optimized circuits using the approach described in [11] together with a basic set of 14 templates[3]. Our new optimization method was then applied. This approach shows that even for circuits optimized by other means, further significant reductions can be achieved if helper lines and the algorithm introduced above are used[4].

Table II summarizes the obtained results for one and two helper lines, respectively. The first three columns give the name of the circuit (including the RevLib file ID), the number of circuit lines, as well as the number of gates of the initial (already optimized) circuit, respectively. In the following columns, the obtained results for quantum cost and transistor count models are presented. The proposed approach has been applied with one and with two helper lines to both the circuit as given as well as in the reversed order. The better of the two results is chosen. The percentage improvement is shown for each case relative to the initial circuit cost, which is the cost after template application where applicable. Finally, the last column gives the highest CPU time (in seconds) of a single run for each benchmark. Space does not allow us to report the results for all circuits. Thus, small circuits that have less than 5 lines and less than 10 gates are omitted. Furthermore, for some circuits adding a helper line gave no improvement. Those benchmarks are listed at the bottom of Table II.

Considering quantum cost, for most of the circuits significant cost reductions can be observed, even if only a single line is added. Over all circuits (including the ones that gave no improvement), adding a single line reduces the quantum cost by 22.51% on average and in the best case (*cycle17_3_112*) by just over 69%. This can be further improved if another line is added leading to an average additional reduction of 5.10%. If transistor cost is considered, the reductions are somewhat smaller but still significant. When adding a single line the transistor cost is reduced by 5.83% on average and in the best case (*cycle17_3_112*) by 37%. Adding a second line reduces the transistor count by a further 1.65%. Since two additional lines is negligible in CMOS technologies, this is a notable reduction as well. In addition, these optimizations can be achieved in very short run-time. Even for circuits including thousands of gates our approach terminates after some minutes – in most of the cases after some seconds.

Finally, we evaluated the improvement achieved when more than two helper lines are added. More precisely, we have applied our method with from one to five helper lines to all the circuits on the RevLib website (including the small ones that have been omitted in Table II). Again, all these circuits were pre-optimized using templates as described above. A total of 95 of the 177 circuits show an improvement in quantum cost when a single helper line is added. Of the other 82 circuits, 64 have a very small number of lines (less than or equal to 5) and are already highly optimized due to their relatively small size. Fig. 4 shows the improvement in quantum cost

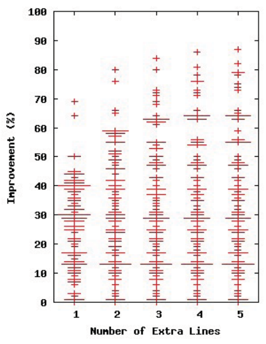
[3]This took over 10 hours of computer time. Furthermore, we were not able to apply the templates to the *urf* series of circuits (which are quite large) because of run-time restrictions.

[4]Note, that similar results are also achieved if the proposed approach is directly applied to non-optimized circuits.

TABLE II
EXPERIMENTAL RESULTS FOR REVLIB CIRCUITS.

			Quantum Cost Model					Transistor Cost Model					
				Add 1 Line		Add 2 Lines			Add 1 Line		Add 2 Lines		
			Initial	Optimized	%	Optimized	%	Initial	Optimized	%	Optimized	%	Max
Circuit[a]	**Lines**	**Gates**	**Cost**	**Cost**	**Impr.**	**Cost**	**Impr.**	**Count**	**Cost**	**Impr.**	**Cost**	**Impr.**	**CPU (s)**
cycle17_3_112	20	48	6063	1877	69.04	1227	79.76	3272	2040	37.65	1728	47.19	2.10
cycle10_2_110	12	19	1202	436	63.73	293	75.62	800	568	29.00	512	36.00	0.18
plus63mod4096_163	12	471	25843	12972	49.80	8865	65.70	16200	14160	12.59	13000	19.75	2.87
plus63mod8192_164	13	549	38462	19347	49.70	15798	58.93	21840	19064	12.71	16912	22.56	3.83
urf2_153	8	638	17027	9308	45.33	7052	58.58	18240	16280	10.75	15688	13.99	3.29
urf5_159	9	499	24523	13494	44.97	8576	65.03	18640	15520	16.74	13640	26.82	3.16
urf2_154	8	620	16152	8979	44.41	6849	57.60	17432	15712	9.87	15168	12.99	3.15
hwb8_115	8	610	14695	8178	44.35	6212	57.73	15336	13936	9.13	13448	12.31	2.84
hwb8_118	8	633	16526	9226	44.17	7094	57.07	17680	15936	9.86	15432	12.71	3.20
urf3_156	10	2732	128172	74431	41.93	51980	59.45	103680	91320	11.92	83104	19.85	18.87
urf1_150	9	1517	48952	28619	41.54	20532	58.06	48616	43024	11.50	40728	16.23	8.81
urf3_157	10	2674	121716	71362	41.37	50101	58.84	100544	88568	11.91	81016	19.42	18.40
hwb8_114	8	748	11941	7078	40.73	5774	51.65	14488	13640	5.85	13392	7.56	2.87
urf1_151	9	1487	45855	27209	40.66	19817	56.78	47024	41760	11.19	39648	15.69	8.51
hwb8_113	8	808	13460	8075	40.01	6656	50.55	16896	15912	5.82	15648	7.39	3.26
hwb9_120	9	1538	44708	26951	39.72	19913	55.46	46448	41496	10.66	39592	14.76	8.46
hwb9_122	9	1535	44659	26928	39.70	19904	55.43	46384	41456	10.62	39552	14.73	8.43
hwb9_123	9	1952	22482	13592	39.54	10935	51.36	28696	27760	3.26	27360	4.66	6.25
hwb8_117	8	748	7014	4365	37.77	3770	46.25	10528	10200	3.12	10120	3.88	2.32
hwb8_116	8	753	6976	4353	37.60	3771	45.94	10536	10216	3.04	10144	3.72	2.31
hwb9_119	9	2013	35967	23119	35.72	18498	48.57	44344	41400	6.64	40368	8.97	8.63
hwb9_121	9	2004	35973	23130	35.70	18499	48.58	44304	41360	6.64	40328	8.97	8.65
hwb7_60	7	166	1754	1147	34.61	1010	42.42	3168	2960	6.57	2912	8.08	0.81
mod8-10_177	5	12	84	55	34.52	44	47.62	144	144		144		0.03
hwb7_59	7	387	3939	2598	34.04	2291	41.84	6800	6472	4.82	6400	5.88	1.39
4gt4-v0_73	5	17	57	38	33.33	38	33.33	144	144		144		0.06
hwb7_62	7	328	2608	1773	32.02	1607	38.38	4632	4512	2.59	4512	2.59	1.01
4gt12-v0_86	5	15	47	32	31.91	32	31.91	136	104	23.53	104	23.53	0.05
alu-v2_30	5	19	111	76	31.53	62	44.14	240	208	13.33	176	26.67	0.08
plus127mod8192_162	13	1077	61425	42713	30.46	29736	51.59	39984	36056	9.82	31872	20.29	7.11
ham7_104	7	23	83	58	30.12	58	30.12	272	272		272		0.06
hwb7_61	7	305	3261	2289	29.81	2082	36.15	5592	5432	2.86	5408	3.29	1.12
hwb6_56	6	153	1329	937	29.50	871	34.46	2456	2392	2.61	2392	2.61	0.52
alu-v2_31	5	13	45	32	28.89	32	28.89	144	128	11.11	128	11.11	0.06
4gt4-v0_78	5	13	53	38	28.30	38	28.30	144	112	22.22	112	22.22	0.06
ham15_108	15	77	403	290	28.04	261	35.24	992	968	2.42	968	2.42	0.25
rd53_136	7	15	76	56	26.32	49	35.53	208	208		208		0.04
hwb6_57	6	65	433	322	25.64	299	30.95	976	928	4.92	928	4.92	0.29
4gt12-v0_87	5	11	43	32	25.58	32	25.58	104	104		104		0.04
4gt11_82	5	12	16	12	25.00	12	25.00	104	72	30.77	72	30.77	0.04
rd53_135	7	19	68	51	25.00	51	25.00	224	216	3.57	216	3.57	0.05
rd53_137	7	16	65	49	24.62	49	24.62	176	176		176		0.05
rd53_131	7	23	106	81	23.58	81	23.58	224	224		224		0.06
ham15_107	15	177	1174	916	21.98	823	29.90	2456	2360	3.91	2336	4.89	0.62
urf6_160	15	10740	53700	42451	20.95	42138	21.53	171840	157584	8.30	157376	8.42	144.85
sym6_145	7	59	348	279	19.83	265	23.85	744	728	2.15	728	2.15	0.18
rd53_130	7	28	230	190	17.39	174	24.35	344	344		344		0.08
mod5adder_127	6	21	121	100	17.36	100	17.36	216	216		216		0.05
one-two-three-v0_97	5	13	65	54	16.92	54	16.92	200	200		200		0.05
urf2_161	8	3250	20465	17127	16.31	16542	19.17	54416	53664	1.38	53664	1.38	11.64
4mod5-v0_18	5	11	19	16	15.79	15	21.05	88	64	27.27	56	36.36	0.04
ham15_109	15	109	206	176	14.56	169	17.96	1008	1008		1008		0.28
hwb5_53	5	62	286	247	13.64	247	13.64	824	824		824		0.18
sym9_148	10	295	1250	1080	13.60	1048	16.16	3448	3432	0.46	3416	0.93	0.82
urf3_155	10	26468	132340	114417	13.54	113886	13.94	423488	413760	2.30	413656	2.32	136.25
4gt13_90	5	15	31	27	12.90	27	12.90	128	96	25.00	96	25.00	0.05
rd53_132	7	27	118	103	12.71	103	12.71	184	184		184		0.06
cnt3-5_180	16	20	120	105	12.50	105	12.50	320	320		320		0.09
hwb5_54	5	24	72	63	12.50	63	12.50	240	240		240		0.09
urf1_149	9	11554	57770	50603	12.41	50410	12.74	184864	181712	1.71	181712	1.71	44.99
rd53_133	7	20	73	64	12.33	64	12.33	240	232	3.33	232	3.33	0.06
hwb5_55	5	26	98	87	11.22	87	11.22	296	296		296		0.07
urf5_158	9	10276	51380	45931	10.61	45650	11.15	164416	162440	1.20	162408	1.22	38.61
urf2_152	8	5030	25150	22557	10.31	22516	10.47	80480	79856	0.78	79856	0.78	17.53
urf4_187	11	32004	160020	147137	8.05	146890	8.21	512064	501392	2.08	501280	2.11	1189.77
4gt5_76	5	14	26	24	7.69	24	7.69	112	104	7.14	104	7.14	0.05
sys6-v0_111	10	23	71	69	2.82	68	4.23	280	264	5.71	256	8.57	0.07
rd53_138	8	15	43	42	2.33	42	2.33	176	168	4.55	168	4.55	0.05
rd73_140	10	23	75	74	1.33	74	1.33	288	280	2.78	280	2.78	0.06
sym9_146	12	28	108	107	0.93	107	0.93	384	376	2.08	376	2.08	0.09
rd84_142	15	31	111	110	0.90	110	0.90	408	400	1.96	400	1.96	0.10

[a]To provide an even basis for the evaluation, all circuits already went through template optimization (using the approach described in [11] together with a basic set of 14 templates) before our approach have been applied. For brevity, small circuits (i.e. circuits that have less than 5 lines and less than 10 gates) are omitted. The circuits *0410184_169, 0410184_170, 4gt13_91, cnt3-5_179, decod24-enable_126, ham7_105, ham7_106, hwb6_58, mod5adder_128, mod5adder_129, rd73_141, rd84_143, sym9_147 and sys6-v0_144* gave no improvement.



Helper Lines	% Average Cost Reduction	Increment
1	26.58%	
2	32.38%	5.80
3	34.15%	1.77
4	34.73%	0.570
5	34.97%	0.24

Fig. 4. Improvement for up to five Helper Lines.

of the remaining circuits (both, for the respective benchmarks in the plot diagram as well as on average in the table). As noted above, a significant improvement can be observed if a single helper line is added. This is further increased if more lines are applied. However, the improvements diminishes with increasing number of helper lines. Finally, no further improvement have been observed, if a sixth line is applied. This is the expected behaviour since multiple helper lines are only useful when multiple gates sharing common factors are present.

In summary, by applying the proposed approach significant cost reductions can be achieved if a single line is added to the circuit (even for already optimized realizations). Further (diminishing) improvements result if more than one helper lines are added.

VI. Concluding Remarks

In this paper we showed that adding lines to a reversible circuit can reduce its cost and that the reduction can be quite significant even if only one or two lines are added. The reduction is as expected higher for the exponential quantum cost model than it is for the linear transistor count model.

The factoring method presented adds additional Toffoli gates to a circuit and would thus appear to increase the delay through the circuit. However, for the quantum model each MCT (including Toffoli) gate represents a cascade of quantum gates. By applying our approach we shorten the length of the corresponding cascades and thus reduce the total number of quantum gates. The actual delay change would have to be analyzed for each circuit. The same is true for the transistor model, and in that case the delay would also have to be analyzed, albeit in quite a different manner.

The most critical issue is the fact, that additional lines (and in the quantum case qubits) must be added to enable the possible optimizations. Thus, the designer must trade off if this additional expense is balanced by the subsequent implementation reductions. Since up to 70% of the quantum cost can be saved, this should be the case for many circuits.

For future work, alternative methods for choosing factors should be compared to the search procedure used in this work. Further, the possibility of introducing helper lines during the synthesis process rather than as a post-synthesis optimization should be investigated.

Acknowledgment

This work was supported in part by a Discovery Grant from the Natural Sciences and Engineering Council of Canada and by the German Research Foundation (DFG) (DR 287/20-1).

References

[1] V. V. Zhirnov, R. K. Cavin, J. A. Hutchby, and G. I. Bourianoff, "Limits to binary logic switch scaling – a gedanken model," *Proc. of the IEEE*, vol. 91, no. 11, pp. 1934–1939, 2003.

[2] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Dev.*, vol. 5, p. 183, 1961.

[3] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Dev*, vol. 17, no. 6, pp. 525–532, 1973.

[4] B. Desoete and A. de Vos, "A reversible carry-look-ahead adder using control gates," *INTEGRATION, the VLSI Journal*, vol. 33, no. 1-2, pp. 89–104, 2002.

[5] T. Simonite, "Error-check breakthrough in quantum computing," *New Scientist*, June 8 2006.

[6] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Reversible logic circuit synthesis," in *Proc. Int'l Conf. on CAD*, 2002, pp. 125–132.

[7] A. Agrawal and N. Jha, "Reversible logic synthesis," in *Proc. Design, Automation and Test in Europe*, 2004, pp. 1384–1385.

[8] J. Donald and N. K. Jha, "Reversible logic synthesis with Fredkin and Peres gates," *J. Emergin Technology Computing Systems*, vol. 4, no. 1, pp. 1–19, 2008.

[9] P. Gupta, A. Agrawal, , and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 25, no. 11, pp. 2317–2330, Nov. 2006.

[10] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proc. Design Automation Conf.*, 2003, pp. 318–323.

[11] D. Maslov, G. W. Dueck, and D. M. Miller, "Techniques for the synthesis of reversible Toffoli networks," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 12, no. 4, pp. 42.1–42.28, 2007.

[12] M. Saeedi, M. Sedighi, and M. S. Zamani, "A novel synthesis algorithm for reversible circuits," in *Proc. Int'l Conf. on CAD*, 2007, pp. 65–68.

[13] M. Saeedi, M. S. Zamani, and M. Sedighi, "On the behavior of substitution-based reversible circuit synthesis algorithms: Investigation and improvement," in *Proc. IEEE Computer Society Symp. on VLSI*, 2007, pp. 428–436.

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

[15] R. Wille, D. Große, G. Dueck, and R. Drechsler, "Reversible Logic Synthesis with Output Permutation," in *VLSI Design*, 2009, pp. 189–194.

[16] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for designing CNOT-based quantum circuits," in *Proc. Design Automation Conf.*, 2002, pp. 419–424.

[17] D. Maslov, C. Young, D. M. Miller, and G. W. Dueck, "Quantum circuit simplification using templates," in *Proc. Design, Automation and Test in Europe*, 2005, pp. 1208–1213.

[18] J. Zhong and J.C.Muzio, "Using crosspoint faults in simplifying Toffoli networks," in *International IEEE Northeast Workshop on Circuits and Systems*, 2006, pp. 129–132.

[19] A. Barenco, C. Bennett, R. Cleve, D. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, vol. 52, pp. 3457–3467, 1995.

[20] R. Wille and R. Drechsler, "BDD-based Synthesis of Reversible Logic for Large Functions," in *Proc. Design Automation Conf.*, 2009, pp. 270–275.

[21] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 22, no. 6, pp. 710–722, 2003.

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

[23] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *Int'l Symp. on Multi-Valued Logic*, 2008, pp. 220–225, RevLib is available at www.revlib.org.

[24] M. K. Thomson and R. Glück, "Optimized reversible binary-coded decimal adders," *J. of Systems Architecture*, vol. 54, pp. 697–706, 2008.

[25] D. Maslov and G. W. Dueck, "Reversible cascades with minimal garbage," *IEEE Trans. on CAD*, vol. 23, no. 11, pp. 1497–1509, 2004.

[26] R. Wille, D. Große, D. M. Miller, and R. Drechsler, "Equivalence checking of reversible circuits," in *Proc. Int'l Symp. on Multiple-valued Logic*, 2009.