A Post-Synthesis Optimization Technique for Reversible Circuits Exploiting Negative Control Lines

Kamalika Datta, Indranil Sengupta, and Hafizur Rahaman

Abstract—Recent works in the synthesis of reversible logic circuits have been motivated by ever increasing emphasis on low-power design alternatives, and recent developments in quantum computing. Although most of the synthesis approaches use multiple-control Toffoli (MCT) gates with positive control lines, a few recent works have also considered MCT gates with negative control lines resulting in better circuit realizations. Some of the works have also tried to carry out post-synthesis optimization of given MCT gate netlists with positive control lines, using template matching and similar netlist transformation techniques. However, only one work is reported that attempts to optimize netlists containing negative control MCT gates. This paper proposes an efficient optimization technique for MCT gate netlists with both positive and negative control lines, which is based on repeated applications of a small set of pairwise gate merging and replacement rules. Experiments carried out on reversible circuit benchmarks show that it is possible to achieve significant reductions in number of gates and quantum costs.

Index Terms—Reversible logic, post-synthesis optimization, ESOP, negative control

1 Introduction

With the sustained developments in semiconductor technology over the last few decades and Moore's law continuing to hold, power consumption in integrated circuits has become a major concern. At the same time, researchers are looking for alternative technologies that would enable them to address the power consumption issue. Reversible logic circuits have been projected as a potential futuristic technology in this regard.

The motivation for considering reversible circuits as a viable alternative comes from several directions. Firstly, there is a principle proposed by Landauer [9] that quantifies that whenever some information bit is lost during computation, at least $KT \log 2$ Joule of energy is dissipated as heat, where K is the Boltzmann constant, and T the absolute temperature of the environment. Very recently, a group of physicists [3] experimentally validated this principle by actually measuring the tiny amount of heat dissipated when one bit of information is erased. Also Bennett argued [2] that the theoretical limit of zero power dissipation is possible only when computations are reversible in nature. In addition, recent developments in the area of quantum computation [15] have motivated research in reversible circuits, since all the basic quantum operations are inherently reversible in nature.

There have been a lot of research in the synthesis of reversible logic circuits in the past few years [7], [8], [10], [17], [22], [23]. The problem of reversible circuit synthesis is significantly different from that of classical logic circuits, and existing methodologies cannot be used directly. In particular, reversible circuit implementations cannot use feedback or fanout connections. Also generation of exact (minimum gate) solutions for larger circuits is very difficult, and so only methods that generate sub-optimal netlists exist for such circuits. Such netlists have ample scopes for optimization. Post-synthesis optimization of reversible circuit netlists have therefore drawn the attention of several researchers. Optimization techniques based on template matching [12] and window optimization [19] have been proposed that are based on multiple-control Toffoli (MCT) gates with positive control lines. However, MCT gates with negative control lines have been proposed recently [24], which has been shown to provide significant reductions in the cost of circuit realizations. To the best of knowledge of the authors, only one post-synthesis optimization technique [4] that uses negative control MCT gates exists in the literature. Since it is expected that the use of negative control gates can lead to a reduction in cost, the present paper proposes a simple rule-based netlist optimization technique is this regard.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to reversible circuits, the cost metrics used for evaluation, and a brief review of the existing netlist optimization techniques. The proposed approach is discussed in Section 3, followed by a discussion of the experimental results in Section 4. Finally, Section 5 concludes the paper and identifies a few directions for future work.

2 Background

In this section we discuss about some preliminaries of reversible logic followed by a brief review of some of the existing post-synthesis optimization techniques.

2.1 Reversible Logic and Reversible Gates

A Boolean function $f : \mathbf{B}^n \rightarrow \mathbf{B}^n$ is said to be reversible if it is bijective, that is, there is a unique correspondence between input and output. Among various traditional logic gates such as AND, OR, NOR, NAND, EXOR, NOT, only NOT gate is reversible in nature. There exists various other well-known reversible gates in the literature, like CNOT [5], Toffoli [21], Fredkin [6], etc.

There exists some constraints on reversible circuit implementations. For instance, it must consist of reversible gates, without any fanout or feedback. Fig. 1 shows some of the basic reversible gates. Toffoli gate is a three-input gate, where the logic value on the target line gets inverted if both the control inputs are 1. CNOT and NOT are special cases of Toffoli gates, with 2- and 1-inputs respectively.

Here in this paper, we have used multiple-control Toffoli gates, which can be defined over a set of inputs $X = \{x_1, x_2, \ldots, x_n\}$, a (possibly empty) set of control lines $C = \{x_{i_1}, x_{i_2}, \ldots x_{i_k}\} \in X$, and a target line $x_j \in (X - C)$. The target line gets inverted if all the control lines are set to 1. Recently in [11] MCT gates with negative control is introduced, where the target line gets inverted only if the positive (negative) controls have the value 1 (0). Fig. 2 shows a MCT gate netlist consisting of gates with combinations of positive and negative controls. To evaluate a synthesis process, several metrics are used, namely, the number of gates or *Gate Count (GC)*, *Quantum Cost (QC)* [1] which is the equivalent cost in quantum technology, and *Transistor Cost (TC)* [20] which is the cost of equivalent transistor implementation. In some recent papers [13], [16], an improved quantum cost metric has been proposed based on better quantum gate realizations of MCT gates.

In the context of the present work, however, since the main goal in experimentation is to demonstrate the improvements gained in terms of cost, we have used the conventional quantum cost metric of [1] that is also used in *RevKit* [18]. For a MCT gate netlist $G = \{g_1, g_2, \ldots, g_p\}$ comprising of n lines, QC is estimated as

$$QC = \sum_{i=1}^{p} quantum\_cost(n, c_i), \quad (1)$$

- *K. Datta and H. Rahaman are with the Department of Information Technology, Indian Institute of Engineering Science and Technology, Shibpur, Howrah 711103, India. E-mail: kdatta.iitkgp@gmail.com, rahaman_h@it.becs.ac.in.*
- *I. Sengupta is with the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur 721302, India. E-mail: isg@iitkgp.ac.in.*

Manuscript received 5 May 2013; revised 13 Dec. 2013; accepted 5 Mar. 2014. Date of publication 2 Apr. 2014; date of current version 13 Mar. 2015.
Recommended for acceptance by D. Kagaris.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2014.2315641

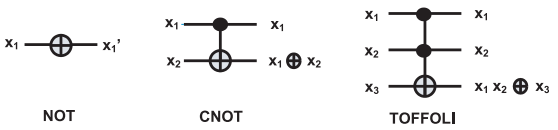


Fig. 1. Basic reversible gates: NOT, C-NOT, Toffoli.

where c_i denotes the number of control connections in gate g_i. The value of *quantum_cost* (n,c) is calculated following the approach used in *RevKit* [18].

For gates with negative control connections, the calculation is similar with just one exception. For a gate with all negative control connections, the quantum cost is *one more* [14] than that estimated using [18].

For example, for the netlist shown in Fig. 2, the quantum cost is calculated as

$$QC = 5 + 5 + 2^* + 5 + 6^* = 23,$$

where the numbers marked by asterisks correspond to gates with all negative control connections.

2.2 Post-Synthesis Optimization Techniques

An optimization approach for MCT gate netlists with positive control lines is presented in [12]. In the work, a template based approach is used in conjunction with a synthesis technique that uses so-called mEXOR gates. A template is defined as a network of gates that realizes the identity function. Essentially, if a sequence of gates in the netlist matches with more than half the number of gates in a template, then the original sequence of gates is replaced by the remaining gates in the template. All templates with up to six gates have been used, and results shown for relatively smaller benchmark circuits.

Another optimization approach that also uses positive control gates is reported in [19]. The method uses window optimization, where instead of looking at the entire netlist at a time, a smaller sub-netlist called a *window* is considered. Two strategies for extracting the windows from a given netlist are used in the work. The first strategy, called *Shift Window Optimization (SWO)*, shifts a window of fixed size across the netlist from left to right. Clearly, windows can overlap in this approach. The second strategy, called *Line Window Optimization (LWO)*, considers sub-netlists with respect to the number of lines, with the number gradually increased in an iterative fashion. Once a window is extracted, various alternate methods are used to optimize it, like re-synthesis, exact synthesis, and some existing optimization technique. Optimization results with up to 12 lines have been reported.

In [4], for the first time a post-synthesis optimization technique for MCT gate netlists that also considers negative control lines has been presented. The method is inspired by template matching, where a set of seven rules have been identified that explicitly makes use of the properties of negative control lines. The template matching rules that have been proposed make the method particularly suitable for ESOP-generated netlists. Experimental results on several reversible circuit benchmarks from [23] show that it is possible to obtain significant reductions in the complexity of the netlists, both in terms of number of gates and quantum cost.

In the present work, a post-synthesis optimization technique has been presented that uses gates with both positive and negative controls. Instead of using a fixed set of templates, this work uses the concept of pairwise gate merging and replacement rules. Although very simple, these rules are very powerful and helps in outperforming the only comparable work [4] both in terms of number of gates and quantum cost.

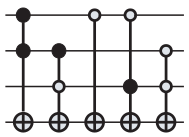


Fig. 2. MCT gates with both positive and negative controls.

3 Proposed Netlist Optimization Approach

We now discuss the details of the proposed approach for netlist optimization. The algorithms as developed have been implemented as a tool that takes any given MCT gate netlist as input, and produces an optimized netlist as output. Basically a small set of rules based on pairwise gate merging and gate replacement are applied iteratively to identified *segments* in the netlist for performing the optimization. This results in a significant reduction in the number of gates and also quantum cost as illustrated later for many benchmark circuits.

3.1 Some Definitions and Optimization Rules

We now present some definitions and rules used for optimization, using which the algorithms will be presented in the next subsection. First we introduce the notion of a *segment*, which is a sub-netlist that can be independently optimized.

Definition 1 (Segment). *In a given MCT gate netlist $G = \{g_1, g_2, \ldots, g_p\}$, a segment is defined as a set of consecutive gates $G_{seg} = \{g_{k_1}, g_{k_1+1}, \ldots, g_{k_2}\}$, where $G_{seg} \subseteq G$, $k_1 \geq 1$ and $k_2 \leq p$, such that any pair of gates in G_{seg} can be swapped without affecting the functionality of the circuit.*

We now introduce the notion of *adjacent gates* and *distance-2 gates*, which can be merged and/or replaced by other gates so as to reduce the quantum cost.

Definition 2 (Adjacent Gates). *Two MCT gates g_1 and g_2 are said to be* adjacent *if they have the targets on the same line, and differ in only one line with respect to the control connections (positive, negative or don't care).*

Definition 3 (Distance-2 Gates). *Two MCT gates g_1 and g_2 are said to be* distance-2 gates *if they have the targets on the same line, and differ in two lines with respect to the control connections (positive, negative or don't care).*

Two adjacent gates can be merged into a single gate, resulting in a reduction of quantum cost. Similarly, two distance-2 gates can be replaced by two simpler gates, which may also reduce the quantum cost. These are summarized in terms of the following two simple rules that are powerful enough to express more complex rules in an elegant way.

Rule 1 (Merging Rule for Adjacent Gates). Consider two adjacent MCT gates g_1 and g_2 with n lines numbered from 1 to n, with the targets for both gates located on line t, $1 \leq t \leq n$. Also assume that the gates differ in line i with respect to the control connections, and identical in all the remaining $(n-2)$ lines. The gates g_1 and g_2 can be merged into a single gate g_m with the target and control connections identical to g_1 (or g_2) in all lines other than i, and

a) if g_1 and g_2 have negative control and positive control connections respectively on line i, g_m will have a don't care (that is, no connection) on line i;
b) if g_1 and g_2 have negative control and don't care connections respectively on line i, g_m will have a positive control connection on line i;
c) if g_1 and g_2 have positive control and don't care connections respectively on line i, g_m will have a negative control connection on line i.

TABLE 1
Replacement Scenarios for Distance-2 Gates

Case	Line	g_1	g_2	g_{1r}	g_{2r}	Impact on QC
1	i	0	1	-	1	Decreases
	j	1	0	1	-	
2	i	0	-	1	-	Decreases
	j	1	0	1	-	
3	i	1	-	-	0	Decreases
	j	0	1	-	0	
4	i	-	0	1	0	Increases
	j	-	0	-	1	
5	i	-	1	0	1	Increases
	j	-	1	-	0	
6	i	-	0	1	0	Increases
	j	-	1	-	0	

'1': positive control, '0': negative control, '-': don't care

Rule 2 (Replacement Rule for Distance-2 Gates). Consider two distance-2 MCT gates g_1 and g_2 with n lines numbered from 1 to n, with the targets for both gates located on line t, $1 \leq t \leq n$. Also assume that the gates differ in lines i and j with respect to the control connections, and identical in all the remaining $(n-3)$ lines. The gates g_1 and g_2 can be replaced by gates g_{1r} and g_{2r} with the target and control connections identical to g_1 (or g_2) in all lines other than i and j. There are six possible cases summarized in Table 1.

It can be seen from the table that for a pair of distance-2 gates, the first three cases of replacement result in a reduction of quantum cost, while the last three result in an increase in quantum cost. This condition must be checked before applying the rule for replacing pairs of distance-2 gates.

3.2 Basic Steps of Optimization

For a MCT gate netlist consisting of positive control gates, some of the NOT gates can be eliminated by using negative control gates. We can scan the netlist from left to right and eliminate pairs of NOT gates on line i (say) where possible by inverting the control connections on line i in all the intermediate gates. Fig. 3 shows an example netlist, and the steps of eliminating some of the NOT gates therein.

From a given gate netlist $G = \{g_1, g_2, \ldots, g_p\}$ and any starting index k_1, a segment can be extracted using a simple procedure as depicted in Algorithm 1. As per the definition of a segment, any pair of gates therein can be swapped without affecting the functionality. In order to satisfy this requirement, the gate netlist G is scanned starting from gate g_{k_1} including all gates up to some gate g_{k_2} such that

a) no control connections (• or ○) appear on any of the lines in which some gate between g_{k_1} and g_{k_2} has its target connection (⊕), or
b) the end of the netlist is reached (that is, $k_2 = p$).

The algorithm returns the index k_2 of the last gate in the segment.

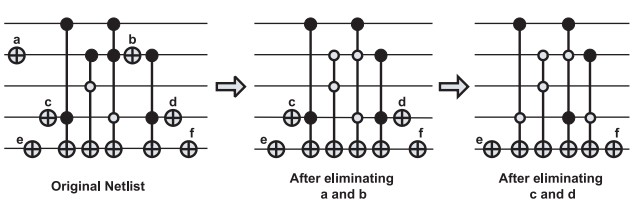


Fig. 3. NOT gate elimination example.

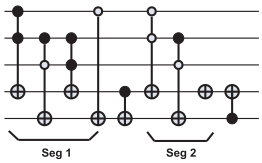


Fig. 4. Identification of segments in a netlist.

Algorithm 1: Extract segment from a given netlist

Input: Given netlist of MCT gates $G = \{g_1, g_2, \ldots, g_p\}$;
Starting index of segment, k_1
Output: Index of the last gate of segment, k_2
begin
  output_list = $\emptyset$;
  output_list = *output_list* $\cup$ *target_in* (G, k_1);
    // Add target lines to output_list
  for i = $(k_1 + 1)$ **to** p **do**
    begin
      for (all lines l in g_i having a control connection) **do**
        if ($l \in$ *output_list*)
          return $(i - 1)$;
            // Control connection on output line
      output_list = *output_list* $\cup$ *target_in* (G, i);
    end
  return p; // Reached end of netlist
end

In the above algorithm, the function *target_in* (G, i) returns the target line number of gate g_i in the netlist G. An example netlist and two segments therein as detected by the algorithm are shown in Fig. 4.

It may also be noted that within a *segment*, any pair of *adjacent (distance-2)* gates can be merged (replaced) using the above rules. An example segment consisting of five gates, and example rule applications are shown in Fig. 5.

Rules 1 and 2 introduced in the previous subsection can be used to merge/replace gate pairs within a segment. The outline of the procedure to do this is given in *A*lgorithm 2.

The function *isAdjacent*(G, i, j) checks if gates g_i and g_j in the netlist G are adjacent, and returns *true* if they are so; otherwise, it returns *false*. The function *merge_gates*(G, i, j) merges the two adjacent gates g_i and g_j in the netlist G, and returns the new gate after merging. The function *atDistance2*(G, i, j) checks if gates g_i and g_j in G are at distance-2, and returns the case number (vide Table 1) if they are so; it returns 0 otherwise. The function *replace_gates*$(G, i, j,$ *case*$)$ returns the new pair of gates after replacing gates g_i and g_j using *case*.

The algorithm to merge/replace gate pairs has a time complexity of $O(n_g^2)$ for every iteration of the *do-while* loop, where $n_g = (k_2 - k_1 + 1)$ is the number of gates within the segment. Through experimentation on a large number of reversible benchmark circuits, it has been found that the average value of n_g is typically less than 500, and the *do-while* loop does not iterate more than ten times.

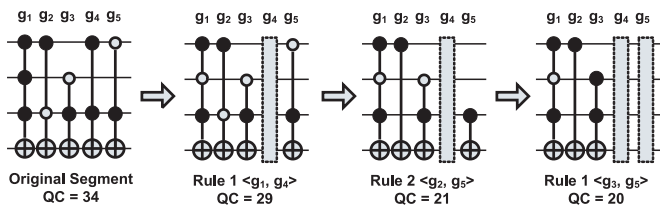


Fig. 5. Example application of rules.

This explains the very small run times of the post optimization process as reported in the experimental results section.

Algorithm 2: Merge/replace gate pairs within a segment

Input: Given netlist of MCT gates $G = \{g_1, g_2, \ldots, g_p\}$; First ($k_1$) and last ($k_2$) gate indices in the segment
Output: Modified netlist G'
begin
  do
    changes = *false*;
    for i = k_1 **to** $(k_2 - 1)$ **do**
      for j = $i + 1$ **to** k_2 **do**
        begin
          if (*isAdjacent* (G, i, j))
            begin
              g_i = *merge_gates* (G, i, j);
              g_j = NULL;
              changes = *true*;
            end
          case = *atDistance2* (G, i, j);
          if $(1 \leq case \leq 3)$
            begin
              $\{g_{ir}, g_{jr}\}$ = *replace_gates* $(G, i, j,$ *case*$)$;
              changes = *true*;
            end
        end
  while (*changes*);
  $G' = G$;
  return G';
end

After reducing the netlist by merging adjacent gates and replacing distance-2 gates within segments using Algorithm 2, a number of *NULL* gates may appear in the netlist. For example, in Fig. 5, gates g_4 and g_5 are NULL gates. The *NULL* gates are eliminated by compacting the netlist after processing each segment in the netlist.

Algorithm 3: Post-synthesis optimization of MCT gate netlist

Input: Given netlist of MCT gates $G = \{g_1, g_2, \ldots, g_p\}$
Output: Optimized netlist G_{opt}
begin
  changes = *false*;
  G_{new} = *reduce_NOT* (G);
  *n*gates = *size_of* (G_{new});
  do
    start = *random* (*ngates*);
    end = *find_segment* $(G_{new}$, *start*$)$;
    if (*start* $<$ *end*)
      begin
        G_{new} = *merge_replace* $(G_{new}$, *start*, *end*$)$;
        G_{new} = *compact_netlist* (G_{new});
        ngates = *size_of* (G_{new});
        changes = *true*;
      end
  while (*changes*);
  G_{opt} = G_{new};
  return G_{opt};
end

3.3 The Overall Approach

In the implementation, a MCT gate netlist is taken as input, and the following transformations are carried out.

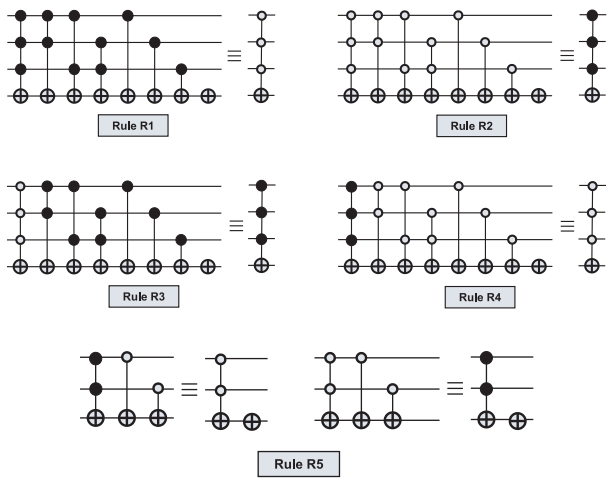


Fig. 6. Illustration of template rules in [4].

a) Pairs of NOT gates on the same line i (say) are eliminated by complementing the polarities of the control connections of all the gates in between. This is a preprocessing step, and can be carried out provided no targets exist in any of the gates in between on line i.
b) Segments of the netlist are determined such that there are no control connections on the lines in which some targets are located. Gates can be freely moved within a segment without affecting the functionality of the circuit.
c) The segment boundaries are stretched to include some gates that are outside the range of the segment.
d) Within a segment, gate pairs that can be merged/replaced by other gates thereby reducing the cost are identified.

Steps (b), (c) and (d) are carried out in an iterative fashion until no further changes in the netlist are possible.

The basic procedures explained in Algorithms 1 and 2 can be combined to frame the overall algorithm for netlist optimization, which is summarized in Algorithm 3.

The functions used in the algorithm are explained briefly in the following.

1) The function *size_of*(G) returns the number of MCT gates in a given netlist G.
2) The function *random*(n) returns a pseudo-random integer between 0 and $(n - 1)$.
3) The function *reduce_NOT*(G) eliminates pairs of NOT gates in a given netlist G where possible.
4) The function *find_segment*(G, i) identifies a segment in a given netlist G starting from index i, and returns the index of the last gate in the segment (vide Algorithm 1).
5) The function *merge_replace*(G, i, j) merges / replaces pairs of adjacent and distance-2 gates between indices i and j in G, and returns the modified netlist (vide Algorithm 2).
6) The *NULL* gates present in a netlist are eliminated using the function *compact_netlist*(G).

3.4 Analysis of the Rules

In a recent work [4], a post-optimization technique using a set of template-based rules using both positive and negative control MCT gates has been presented. Some of these rule applications are illustrated diagrammatically in Fig. 6.

In rule R1 (R2), a set of positive (negative) control MCT gates with all possible control dot patterns are replaced by a single gate with all negative (positive) control lines. Rule R3 (R4) is similar, with only one gate in the original netlist appearing in the opposite polarity. Rule R5 is a corollary of rules R1-R4 where a partial

TABLE 2
Optimization Results on Benchmark Circuits

Benchmark	Lines	Original Netlist [23]		Optimization using [4]		Proposed Optimization			% Impr	
		GC	QC	GC	QC	GC	QC	Time	GC	QC
decod_217	21	80	1746	79	1745	21	613	0.03	73.8	64.9
c7552_205	21	80	1746	79	1745	23	623	0.04	71.3	64.3
sf_274	5	19	157	7	145	6	38	0.00	68.4	75.8
rd32_273	5	20	124	8	112	7	67	0.00	65.0	46.0
sf_276	5	16	154	8	146	6	27	0.00	62.5	82.5
rd53_131	7	28	119	12	104	12	104	0.00	57.1	12.6
sf_275	5	11	53	6	44	5	41	0.00	54.5	22.6
9symm1_195	10	129	14193	58	12747	62	13026	0.26	51.9	8.2
max46_240	10	107	5444	51	4498	52	4538	0.14	51.4	16.6
sym9_193	10	129	14193	58	12747	63	13090	0.28	51.2	7.8
alu4_201	22	1063	55413	523	46413	529	46764	26.90	50.2	15.6
tial_265	22	1041	56224	516	47145	522	47556	22.74	49.9	15.4
life_238	10	107	6767	57	5740	57	5744	0.18	46.7	15.1
f51m_233	22	663	37417	358	33333	355	32882	9.96	46.5	12.1
alu2_199	16	157	5663	87	4776	87	4611	0.44	44.6	18.6
ham7_299	21	61	141	46	135	34	121	0.02	44.3	14.2
table3_264	28	1012	80039	744	79326	577	61412	35.11	43.0	23.3
mux_246	22	35	1078	20	804	20	804	0.04	42.9	25.4
ham15_298	45	153	309	100	290	91	266	0.12	40.5	13.9
misex3_242	28	1752	119177	1199	115637	1043	99119	28.77	40.5	16.8
misex3c_243	28	1721	115190	1188	111258	1049	96064	30.16	39.0	16.6
frg1_234	31	212	15266	130	14737	130	14702	2.26	38.7	3.7
in2_236	29	405	23814	283	23146	250	20600	4.91	38.3	13.5
cordic_218	25	2533	349522	1567	348566	1567	348532	43.00	38.1	0.3
clip_206	14	174	6731	111	6535	109	6119	0.65	37.4	9.1
sym6_145	7	36	777	31	647	23	517	0.04	36.1	33.5
in0_235	26	338	20042	245	18999	218	16985	1.98	35.5	15.3
apex4_202	28	5376	238146	5039	237748	3469	158095	305.14	35.5	33.6
sqr6_259	18	81	1053	66	1034	53	876	0.20	34.6	16.8
dist_223	13	185	7604	126	7288	122	6631	0.78	34.1	12.8
dc1_220	11	39	425	33	419	26	249	0.02	33.3	41.4
sqn_258	10	76	2128	50	2041	51	1887	0.07	32.9	11.3
inc_237	16	93	2145	72	2104	63	1745	0.26	32.3	18.6
sym9_148	10	210	4452	154	3668	143	3433	1.78	31.9	22.9
rd84_313	34	104	304	95	301	71	275	0.03	31.7	9.5
pm1_249	14	35	384	31	354	24	275	0.07	31.4	28.4
5xp1_194	17	85	1418	65	1327	59	1155	0.22	30.6	18.5
dc2_222	15	75	1898	55	1789	53	1688	0.15	29.3	11.1
cm150a_210	22	53	1096	38	822	38	822	0.02	28.3	25.0
bw_291	87	307	943	287	923	222	867	0.38	27.7	8.1
cu_219	25	40	1148	31	1054	29	954	0.02	27.5	16.9
rd73_312	25	73	217	65	214	53	200	0.02	27.4	7.8
mod5adder_306	32	96	292	84	281	70	270	0.02	27.1	7.5
add6_196	19	229	6471	179	6005	167	5534	1.35	27.1	14.5
cycle10_293	12	78	202	74	199	57	186	0.03	26.9	7.9
alu3_200	18	94	2641	79	2512	69	2162	0.25	26.6	18.1
hwb7_302	73	281	909	270	899	208	843	0.20	26.0	7.3
hwb6_301	46	159	507	151	500	119	471	0.06	25.2	7.1
hwb5_300	28	88	276	80	270	67	259	0.02	23.9	6.2
apla_203	22	80	3444	74	3438	64	3024	0.17	20.0	12.2

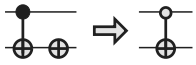
Average improvement in GC: 39.8%
Average improvement in QC: 20.5%

match is obtained, and the missing gates are included in the modified netlist. In addition, the NOT elimination rule as discussed earlier is also used.

It may be noted that the rules R1-R4 can be simulated by repeated applications of the gate merging and replacement rules (rule 1 and 2) as proposed. The main advantage here is that instead of matching much larger templates in the netlist, we only try to apply some rules on *pairs* of gates.

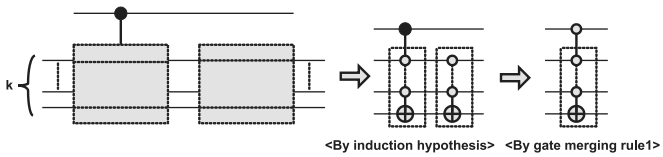
Theorem 1. *Rule R1 of [4] can be simulated using gate merging rule 1.*

Proof. We approach the proof by induction. As the induction hypothesis, we show that the result is true for $n = 2$:



which follows from the gate merging rule 1.

Now assume that the result is true for $n = k$. For $k + 1$ inputs, the given template can be expressed as



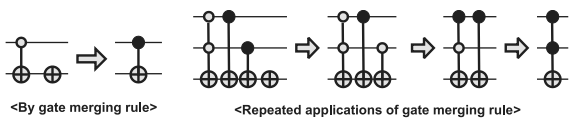
That is, the result is also true for $n = k + 1$. Hence the proof. □

Theorem 2. *Rule R2 of [4] can be simulated using gate merging rule 1.*

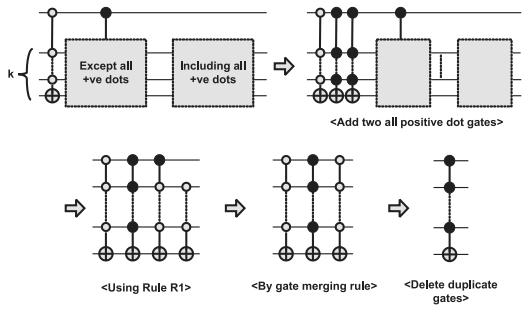
Proof. The proof follows similar to that for Rule R1. □

Theorem 3. *Rule R3 of [4] can be simulated using gate merging rule 1.*

Proof. Here again we prove the result by induction. As the induction hypothesis, it is easy to see that the result holds for $n = 2$ and $n = 3$.



Assume now that the result is true for $n = k$. For $k+1$ inputs, the given template can be expressed as:



The result is thus true for $n = k+1$, and the proof follows. □

Theorem 4. *Rule R4 of [4] can be simulated using gate merging rule 1.*

Proof. The proof follows similar to that for Rule R3. □

It may be noted that Rule R5 does not always follow from repeated applications of the gate merging and replacement rules. However, unpredictable sequences of merging and replacements have the ability of optimization using more complex templates in general. This has also been demonstrated through experiments, where it has been found that the proposed method produces better optimizations that [4].

4 Experimental Results

The proposed post-synthesis optimization tool has been implemented in C on a dual-core Pentium desktop with 4 GB memory and 2.6 GHz clock, running Ubuntu v12.04. Results have been obtained on a large number of reversible benchmark netlists available on the RevLib website [23]. Though the technique can be applied to any MCT gate netlist, since the applicability of the rules mainly depend on the successful detection of segments, it works well for netlists with relatively large segments. In other words, the method is better suited for netlists generated from ESOP expressions.

Table 2 summarizes the results of experimentation. The first column Benchmark represents the names of the benchmark circuits, while the second column Lines represents the number of lines (qubits) in the implementation. The next two columns show the gate counts and quantum costs of the original netlists from [23] with window optimization where possible, while the following two columns show the corresponding values using the method in [4]. The next three columns show the GC and QC values values using the proposed method, and also the time taken in seconds. The last two columns show the percentage improvements of the proposed method with respect to the original netlist. It can be seen that for most of the benchmarks, the run times are under 1 seconds with maximum being about 5 minutes.

The results show that it is possible to obtain improvements of up to 73.8 percent in GC and 64.9 percent in QC. For instance, in the benchmarks `alu4_201` and `tial_265` where the original netlists contained more than 1,000 gates, gate reductions of around 50 percent is observed. The average improvements in GC and QC for the results shown in table 2 are observed to be 39.8 and 20.5 percent respectively. Since the run times are also very low, this represents a significant achievement.

TABLE 3
Segment Sizes and Improvements for Selected Benchmarks

Benchmark	Segment Size		Improvement			
	Max	Avg	GC	QC	%GC	%QC
9symm1_195	56	27.8	67	1167	51.9	8.2
add6_196	140	59.6	62	937	27.1	14.5
alu2_199	82	37.5	70	1052	44.6	18.6
alu4_201	519	233.2	534	8649	50.2	15.6
apex4_202	3725	1690.1	1907	80051	35.5	33.6
f51m_233	332	156.6	308	4535	46.5	12.1
frg1_234	103	43.8	82	564	38.7	3.7
in0_235	164	64.2	120	3057	35.5	15.3
misex3_242	1042	525.8	709	20058	40.5	16.8
misex3c_243	1027	522.9	672	19126	39.0	16.6
sym9_148	153	74.5	67	1019	31.9	22.9
table3_264	691	281.8	435	18627	43.0	23.3
tial_265	526	231.3	519	8668	49.9	15.4

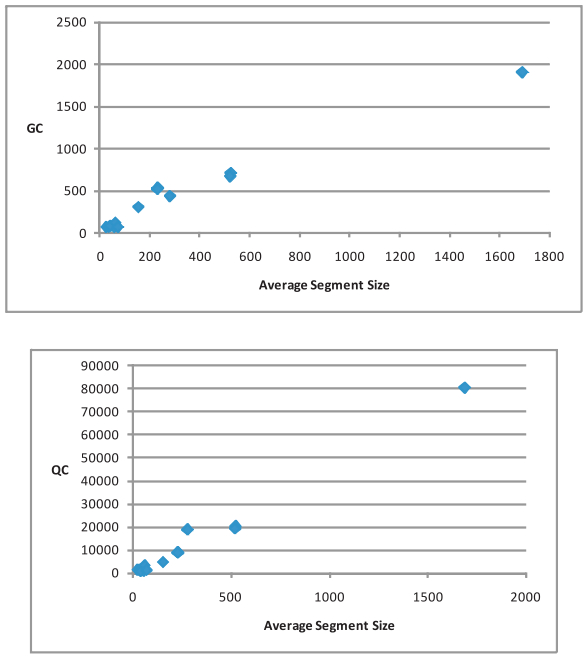


Fig. 7. GC and QC improvements versus segment size.

It may also be noted that the proposed method constantly outperforms the method of [4] in terms of GC as well as QC, in spite of the fact that [4] uses more elaborate and complex template matching rules. The improvements in QC for some of the larger benchmarks are: 33.6 percent for `apex4_202`, 10 percent for `in0_235`, 15.3 percent for `misex3_242`, 16.8 percent for `misex3c_243`, 23.3 percent for `table3_264`, 64.9 percent for `decod_217`, etc. This may be attributed to the fact that the simple gate merging and replacement rules as proposed can simulate many of the template rules of [4], and also many new rules not used in [4]. This fact more than compensates the observation that Rule R5 of [4] cannot be simulated by the proposed rules in the general case. The mobility of the gate within a segment adds to the power of the rules.

The gate merging and replacement rules used in the present work are applied on *segments* of MCT gate netlists. Table 3 shows relevant statistics on some of the benchmarks. The table depicts the maximum and average segment sizes, and also the absolute and percentage improvements over [23] in terms of GC and QC. It can be seen that a good correlation exists between the segment size and the achievable (absolute) improvement, which can also be verified from the scatter plot shown in Fig. 7.

5 Conclusion

In this paper, we have presented a post-synthesis optimization technique for reversible MCT gate netlists which explicitly uses negative control lines for the purpose of optimization. A set of gate merging and replacement rules have been formulated, repeated applications of which help in reducing the number of gates as well as the quantum cost of the netlist. In order to reduce the time complexity of the whole process, segments in the netlists are identified and the rules are applied independently to the various segments. This divide-and-conquer strategy helps in reducing the run time by a great extent. Experimental results confirm that the proposed method leads to significant reduction in cost, and is also better than the only other method available in the literature [4] for netlist optimization using negative control gates. As a future work, better heuristics for extending the sizes of the segments and also better gate reordering and template matching techniques can be incorporated into the tool as developed to provide further reduction in costs.

References

[1] A. Barenco, H. H. Bennett, R. Cleve, D. P. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Rev. A (At., Mol., Opt. Phys.)*, vol. 52, no. 5, pp. 3457–3467, 1995.

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

[3] A. Bèrut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider, and E. Lutz, "Experimental verification of Landauer's principle linking information and thermodynamics," *Nature*, vol. 483, no. 3, pp. 187–189, 2012.

[4] K. Datta, G. Rathi, R. Wille, I. Sengupta, H. Rahaman, and R. Drechsler, "Exploiting negative control lines in the optimization of reversible circuits," in *Proc. Int. Conf. Reversible Comput.*, Jul. 2013, pp. 209–220.

[5] R. Feynman, "Quantum mechanical computers," *Optic News*, vol. 11, pp. 11–20, 1985.

[6] E. Fredkin and T. Toffoli, "Conservative logic," *Int. J. Theoretical Phys.*, vol. 21, pp. 219–253, 1982.

[7] D. Grosse, R. Wille, G. W. Dueck, and R. Drechsler, "Exact multiple control Toffoli network synthesis with SAT techniques," *IEEE Trans. CAD Integr. Circuits Syst.*, vol. 28, no. 5, pp. 703–715, May 2009.

[8] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. CAD Integr. Circuits Syst.*, vol. 25, no. 11, pp. 2317–2329, Nov. 2006.

[9] R. Landauer, "Irreversibility and heat generation in computing process," *IBM J. Res. Devlop.*, vol. 5, no. 3, pp. 183–191, Jul. 1961.

[10] D. Maslov and G. W. Dueck, "Improved quantum cost for n-bit Toffoli gates," *Electron. Lett.*, vol. 39, no. 25, pp. 1790–1791, 2004.

[11] D. Maslov and G. W. Dueck, "Level compaction in quantum circuits," in *Proc. IEEE Congr. Evol. Comput.*, 2006, pp. 2405–2409.

[12] D. Maslov, G. W. Dueck, and D. M. Miller, "Toffoli network synthesis with templates," *IEEE Trans. CAD Integr. Circuits Syst.*, vol. 24, no. 6, pp. 807–817, Jun. 2005.

[13] D. Maslov, C. Young, G. W. Dueck, and D. M. Miller, "NCV realization of MCT gates with mixed controls," in *Proc. Pacific Rim Conf. Commun., Comput. Signal Process.*, 2011, pp. 567–571.

[14] D. M. Miller and Z. Sasanian, "Recent developments on mapping reversible circuits to quantum gate libraries," in *Proc. Int. Symp. Electron. Syst. Des.*, Dec. 2012, pp. 17–22.

[15] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. New York, NY, USA: Cambridge Univ. Press, Oct. 2000.

[16] Z. Sasanian and D. Miller, "Mapping a multiple-control Toffoli gate cascade to an elementary quantum gate circuit," *Multiple-Valued Logic Soft Comput.*, vol. 18, no. 1, pp. 83–98, 2012.

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

[18] M. Soeken, S. Frehse, R. Wille, and R. Drechsler, "RevKit: An open source toolkit for the design of reversible circuits," in *Proc. 3rd Int. Conf. Reversible Comput.*, 2012, vol. 7165, pp. 64–76.

[19] M. Soeken, R. Wille, G. W. Dueck, and R. Drechsler, "Window optimization of reversible and quantum circuits," in *Proc. Symp. Des. Diagnostics Electron. Circuits Syst.*, 2010, pp. 341–345.

[20] M. Thomson and R. Gluck, "Optimized reversible binary-coded decimal adders," *J. Syst. Arch.*, vol. 54, pp. 697–706, 2008.

[21] T. Toffoli, "Reversible computing,," in *Proc. 7th Colloq. Automata, Lang. Program.*, 1980, pp. 632–644.

[22] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Proc. Des. Autom. Conf.*, 2009, pp. 270–275.

[23] R. Wille, D. Grosse, L. Teuber, G. W. Dueck, and R. Drechsler, "Revlib: An online resource for reversible functions and reversible circuits," in *Proc. Int. Symp. Multi-Valued Logic*, May. 2008, pp. 220–225.

[24] R. Wille, M. Soeken, N. Przigoda, and R. Drechsler, "Exact synthesis of Toffoli gate circuits with negative control lines," in *Proc. Intl. Symp. Multi-Valued Logic*, 2012, pp. 69–74.