Translation Techniques for Reversible Circuit Synthesis with Positive and Negative Controls

D. Michael Miller and Gerhard W. Dueck

1 Introduction

A reversible Boolean function is a multiple-output function that maps each input assignment to a unique output assignment. Such a function must have the same number of inputs and outputs and the function always has an inverse. The reversible circuit synthesis problem is to realize such a function as a cascade of reversible gates. In this chapter we present function translations that can improve the synthesized circuit making effective use of both positive and negative controls for the reversible gates.

The first function translation considered negates selected function inputs and the corresponding function outputs. If we synthesize a circuit for the translated function, that circuit is easily translated to become a circuit for the original function by changing the polarity of certain controls in the circuit. For an n-input, n-output function, there are 2^n choices for which input-output pairs to negate, hence a broad range of potential circuits for the original function. The case where all input-output pairs are negated translates the original function to its dual.

The second function translation considered permutes input-output pairs. For an n-input, n-output function, there are $n!$ permutations. Note that as the same permutation is applied to the inputs and outputs of the function, if we find a circuit for the translated function, it can be mapped to a circuit for the original function by a simple relabeling of the inputs and corresponding outputs. Swap gates are not

D. M. Miller (✉)
University of Victoria, Victoria, BC, Canada
e-mail: mmiller@uvic.ca

G. W. Dueck
University of New Brunswick, Fredericton, NB, Canada
e-mail: gdueck@unb.ca

R. Drechsler, D. Große (eds.), *Recent Findings in Boolean Techniques*,
https://doi.org/10.1007/978-3-030-68071-8_7

needed as is the case in some earlier work where permuting the inputs and outputs are considered as separate operations.

The two function translations can be combined and it is also possible to synthesize a circuit for a function from that function or its inverse. This gives a possibility of $2 \times n! \times 2^n$ translations for a given function. We use transformation-based synthesis techniques to demonstrate the effectiveness of the techniques, but we note that the function translations can be applied with any reversible circuit synthesis method.

Transformation-based synthesis for reversible functions was introduced in 2003 [9]. It is a simple technique that in its most basic form generates a reversible circuit by mapping a given function to the identity by considering the rows of a truth table in order from row 0 to row $2^n - 1$. Variants to that basic approach have been developed and are outlined later in this chapter. A bounded search transformation-based synthesis method is also considered. Results presented show it can be effective but at high computational cost.

A second facet of this chapter is the consideration of ways to simplify reversible circuits with positive and negative controls. To that end, we employ simplification rules presented by Rahman and Rice in [15]. In addition, we consider the use of a generalized form of Peres and inverse Peres gates [14] that allows for both negative and positive controls. Once again it is worth noting that the simplification techniques discussed are applicable to the circuits and are not specifically for transformation-based synthesis. They can be used in conjunction with any other reversible circuit synthesis approach.

Often the goal is to map a reversible circuit to a quantum gate implementation. Here we consider mapping to the NCV gate library [13]. In particular, we present an approach to dealing with negative control CNOT gates which are often not permitted in quantum circuit technologies.

This chapter concludes with an assessment of the positives and limitations of this work with suggestions for ongoing research. The use of negative controls in reversible and quantum circuits, *white dots* as they are often called due to the commonly used graphic, have been considered by a number of researchers [2, 10, 11, 15, 16, 21]. We refer the interested reader to those works and acknowledge them as providing motivation for this work.

2 Background

We here present the necessary background on reversible functions, gates, and circuits as well as necessary detail on NCV quantum circuits. Readers seeking more detailed information should consult [13].

Table 1 A 3 × 3 reversible function

x_2	x_1	x_0	x_2^+	x_1^+	x_0^+
0	0	0	1	1	1
0	0	1	0	0	1
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	0	1

2.1 Reversible Functions, Gates, and Circuits

Definition 1 An n-input, n-output, totally-specified Boolean function $f(X)$, $X = \{x_0, x_1, \ldots, x_{n-1}\}$ is *reversible* if it maps each input assignment to a unique output assignment.

A reversible function can be written as a standard truth table as in Table 1 where + denotes output. The function can also be viewed as a bijective mapping of the set of integers $0, 1, \ldots, 2^n - 1$ onto itself. Hence a reversible function can be defined as an ordered set of integers corresponding to the right side of the table, e.g., $\{7, 1, 4, 3, 0, 2, 6, 5\}$, for the function in Table 1 where the decimal number corresponds to the binary sequence in the obvious way. A reversible function is a permutation and can be expressed as a set of disjoint cycles as done in [19], but we do not follow that approach here.

Definition 2 An m-input, m-output gate is a *reversible gate* if it realizes a reversible function.

In this work, we use the family of mixed-polarity multiple-control Toffoli gates described in Definition 3.

Definition 3 An $m \times m$ *mixed-polarity multiple-control Toffoli (MPMCT) gate* has a single target line and $m - 1$ control lines. Each control is either positive, i.e., activated by a 1, or negative, i.e., activated by a 0. The value on the target line is inverted if all positive controls have value 1 and all negative controls have value 0. The controls are always passed through the gate unaltered.

We write an $m \times m$ MPMCT gate as $T(controls, target)$ where negative controls are indicated by an overline. For example, $T(x_1, \overline{x_2}, x_0)$ denotes an MPMCT gate which inverts the value of x_0 if $x_1 = 1$ and $x_2 = 0$. For drawing gates, $\oplus$ denotes a target, $\bullet$ denotes a positive control, and $\circ$ denotes a negative control.

An MPMCT gate with no controls always inverts the target and is thus the well-known NOT gate. An MPMCT gate with a single control is referred to as a *controlled NOT* (CNOT) and is also known as a Feynman gate [4] if the control is positive. An MPMCT gate with two positive controls is the gate originally proposed by Toffoli [25].

Peres gates and their inverse [14] are often used in reversible circuit synthesis. Conventionally, such a gate has two positive controls. Generalizations of Peres gates have been considered in [11, 24]. In this work we employ mixed-polarity Peres and inverse Peres gates as described in the following definition.

Definition 4 A *mixed-polarity Peres* (MPP) gate is a single gate equivalent to a 2-input mixed-control Toffoli gate followed immediately by a positive-control CNOT gate whose target and control are the controls of the Toffoli gate. A *mixed-polarity inverse Peres* (MPIP) gate is similar except the CNOT immediately precedes the Toffoli. Note these definitions are extensions to the original Peres and inverse Peres gates in that the Toffoli gate can have negative as well as positive controls.

An MPP gate will be denoted $P(c, t_1, t_2)$ where t_1 is the target of the CNOT gate with control c and t_2 is the target of the Toffoli gate with controls c and t_1. t_1 can have an overline to indicate it is a negative when used as a control. An MPIP gate is denoted in the same way with IP instead of P.

Definition 5 An $n \times n$ *reversible circuit* is a cascade of reversible gates with no fanout or feedback.

For example, the circuit in Fig. 1 realizes the function in Table 1. Note the third gate $T(x_2, x_1)$ and the fourth gate $T(x_1, x_2, x_0)$ can be replaced by the inverse Peres gate $IP(x_2, x_1, x_0)$.

2.2 Quantum Gates and Circuits

Reversible circuits can be realized in a variety of technologies. Here we consider the NCV quantum library consisting of four elementary gates: NOT, CNOT, controlled-V, and controlled-$V^\dagger$. We here provide the basic background required to understand the use of this library for the work in this chapter.

Definition 6 The basic information unit in a quantum circuit is the *qubit* whose value is given by $\alpha\,|0\rangle + \beta\,|1\rangle$ where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$ and $|0\rangle$ and $|1\rangle$ are basis states normally associated to the Boolean values 0 and 1.

The discussion here is greatly simplified by the fact we are implementing Boolean reversible functions and because the quantum circuits we consider are

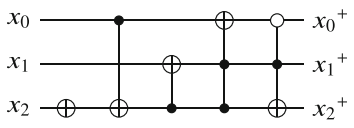

Fig. 1 Reversible circuit for function in Table 1

semi-classical [29] meaning control values are always 0 or 1 thereby avoiding entanglement between qubits.

The operation of a NOT can be expressed as a matrix

$$N = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and applying a NOT to a qubit is given by $N \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$. One can see that if the qubit is in a basis state this operation, as expected, flips it to the other basis state.

The matrices defining the V and $V^\dagger$ operations are

$$V = \frac{1}{2} \begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix} \quad V^\dagger = \frac{1}{2} \begin{bmatrix} 1-i & 1+i \\ 1+i & 1-i \end{bmatrix}$$

It is readily verified that $N = VV = V^\dagger V^\dagger$. For that reason, V and $V^\dagger$ are called the square roots of NOT. It is also readily verified that $VV^\dagger = V^\dagger V = I$, i.e., V and $V^\dagger$ are the inverses of each other.

V and $V^\dagger$ gates always have a single positive control. When used as a quantum elementary gate, CNOT can also only have a positive control which is different from our use of CNOT in a reversible circuit where we allow a positive or a negative control.

Definition 7 An *ancillary line* is a quantum circuit line used in the realization of an MPMCT gate that is not a control or target for that gate. The value of the ancillary is restored to its value, so operation of the gate effectively has no effect on an ancillary,

Quantum circuit cost is discussed in Sect. 6.

3 Function Translations

A number of reversible circuit synthesis methods have been proposed in [17]. Most employ heuristics and are not guaranteed to find an optimal solution, so it is useful to explore alternative formulations of the synthesis problem and translation of the function to be synthesized in particular.

3.1 Function Inverse

Since a reversible function maps each input assignment to a unique output assignment, such a function has an inverse. The following result is well known [9]:

Theorem 1 *Given a reversible circuit* $g_0, g_1, \ldots, g_{k-1}$ *realizing the reversible function* $f(X)$*, the circuit* $g_{k-1}^{-1}, g_{k-2}^{-1}, \ldots, g_0^{-1}$ *realizes the inverse function* $f^{-1}(X)$.

Proof A reversible gate can be represented by a permutation matrix and a reversible circuit is the product of the matrices for the gates in the circuit. The result follows from the fact that the inverse of a product of matrices is the product of the inverses of the matrices in reverse order. □

This theorem is in fact simpler for circuits composed of MPMCT gates since those gates are all self-inverse, so one need only reverse the order of the gates. Peres gates must be replaced by inverse Peres gates and vice versa.

Given this result, one can synthesize circuits for $f(X)$ and for $f^{-1}(X)$ and use the better of the two as a realization for $f(X)$ where Theorem 1 is applied if the circuit found for $f^{-1}(X)$ is used.

3.2 Input-Output Negation

The concept of the dual of a Boolean function is readily extended to reversible functions as per the following definition:

Definition 8 The *dual* of a reversible function $f(X)$ is given by $f^D(X) = \overline{f}(\overline{X})$, where $\overline{f}$ denotes negation of each of the outputs of f and $\overline{X}$ denotes negation of each of the variables in X.

We now show how employing $f^D(X)$ gives a further option for exploring circuits to realize $f(X)$ when both positive and negative gate controls are used.

Theorem 2 *Given an MPMCT circuit G realizing* $f^D(X)$*, a circuit realizing* $f(X)$ *is found by changing all 0 controls to 1 controls and all 1 controls to 0 controls for each gate in G.*

Proof Given an MPMCT circuit for $f^D(X)$, add an inverter to each input and to each output. The result is a circuit realizing $f(X)$ since $f(X) = \overline{f^D}(\overline{X})$. Now move the input inverter from each input across the circuit. As it crosses a control, it inverts that control, and as it crosses a target or passes over a gate not involving the circuit line, it does nothing. When it has passed all gates, it cancels with the corresponding output inverter. So in fact, given an MPMCT circuit G realizing $f^D(X)$, a circuit realizing $f(X)$ is found by simply changing all 0 controls to 1 controls and all 1 controls to 0 controls for each gate in G. □

Definition 8 and Theorem 2 are a special case of a more general translation as follows:

Definition 9 Consider an n-tuple $\alpha = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ where each α_i is either the unary function NOT or the unary identity function. The *α-translation* of a

reversible function $f(X)$ is given by $\hat{f}(X) = \tilde{f}(\tilde{X})$ where $\tilde{f}$ denotes application of each α_i to output f_i and $\tilde{X}$ denotes application of each α_i to input x_i.

Theorem 3 *Given an MPMCT circuit G realizing $\hat{f}(X)$, a circuit realizing $f(X)$ is found by changing all 0 controls to 1 controls and all 1 controls to 0 controls for each gate on the lines in G corresponding to those α_i that are NOTs.*

Proof The proof is essentially the proof for Theorem 2 restricted to the lines for which α_i is NOT. □

Given a function f with n variables specified as a truth table, the steps to synthesize a circuit employing Theorem 3 are as follows:

1. Choose an $\alpha = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$. The dual is the case where all α_i are NOT.
2. Form a new function $\hat{f}(X)$ as given by Definition 9.
3. Use the chosen synthesis method to find a circuit G for $\hat{f}(X)$ with no MPP or MPIP gate substitutions.
4. Invert all controls for all gates in G on lines for which α_i is NOT.
5. Do any possible MPP and MPIP gate substitutions. The result is a circuit for f.

It is important to note that MPP and MPIP gate substitutions are not performed when finding a circuit for $\hat{f}(X)$ since inverting the controls in that circuit does not properly handle the control polarity associated with the CNOT that was used to form the MPP or MPIP gate.

3.3 Input-Output Permutation

Definition 10 An *input-output permutation* is a single permutation σ applied to both the inputs and outputs of a reversible function $f(X)$ yielding a new function $\mathring{f} = \sigma f(\sigma X)$. □

Note that as the same permutation is applied to the inputs and outputs of the function, if we find a circuit for $\mathring{f}(X)$, it can be mapped to a circuit for $f(X)$ by simply reordering the lines in the circuit using σ^{-1}. Swap gates are not needed as is the case in some earlier work where permuting the inputs and outputs are considered as separate operations.

Combining function inverse, input-output negation, and input-output permutation, there are $2 \times 2^n \times n!$ translations of a given reversible function. It is straightforward to translate a circuit for any one of those translations to a circuit for the original function.

4 Transformation-Based Synthesis

As noted earlier, we will use transformation-based synthesis as a means to evaluate the effectiveness of the function translations introduced in the previous section. For ease of description, we present transformation-based synthesis in terms of the truth table representation of a reversible function. Note that transformation-based methods can be implemented using alternate more efficient representations such as decision diagrams [23, 27].

The procedure Map(y, x) described in Algorithm 1 is taken from [22]. It is central to all the transformation-based synthesis algorithms described below. Map identifies a sequence of positive control MCT gates to map the bit pattern y to x where $y > x$. The gates are selected so that they have no effect on any bit pattern $z < x$.

Algorithm 1 begins by setting the control specification c to have as few 1's as possible from y such that $c \geq x$. The latter condition is required to be sure the gates will not affect earlier rows in the truth table. The first **for** loop generates MCT gates with controls c with one gate for each variable outside c that has to be flipped to make y match x. The second **for** loop then uses x as the control and generates one gate for each variable in c that has to be made 0 to match x. In both loops, each gate generated has as its target one of the variables whose value needs to be changed.

Algorithm 1 MCT gate selection to map y to x where $y \geq x$

```
1: procedure MAP(y, x)
2:     glist = empty
3:     if x ≡ y then
4:         return glist
5:     end if
6:     c = y
7:     remove 1 bits from right of c while c ≥ x
8:     p = (x ⊕ y)&(~ c)
9:     for each bit position j = 1 in p do
10:        g=T(c,j)
11:        append g to the end of glist
12:    end for
13:    q = c&(~ x)
14:    c = x
15:    for each bit position j = 1 in q do
16:        g=T(c,j)
17:        append g to the end of glist
18:    end for
19:    return glist
20: end procedure
```

Note: T(c,j) denotes a Toffoli gate with positive controls corresponding to 1 bits in c and target j

Basic Algorithm [9] Given a truth table representing a reversible function f, the basic transformation-based synthesis algorithm [9] proceeds through the truth table

rows in order $0 \leq i < 2^n - 1$. At each row i, if $f(i) \neq i$ MCT gates are selected to map $f(i)$ to i. These gates are chosen such that they do not affect any row j for $j < i$, i.e., those that have already been considered. The gates are added to the circuit being constructed from the output towards the input and the reversible specification is updated by applying the gates to the output side of the specification. When all rows $0 \leq i < 2^n - 1$ have been considered, the resulting truth table is the identity function and the gates chosen represent an implementation of the original reversible function. Note that row $2^n - 1$ does not have to be considered as $f(2^n - 1) = 2^n - 1$ when all previous rows match.

Bidirectional Algorithm [9] The bidirectional transformation-based synthesis is a straightforward extension of the basic algorithm. For each row i, the gates G_0 required to transform the output pattern $f(i)$ to i are determined as in the basic algorithm. In addition, there must be a row j later in the table where $f(j) = i$. MCT gates G_1 that transform j to i are determined. The less expensive of G_0 and G_1 is determined and those gates are added to the circuit and used to update f. Note that if G_1 is chosen, the gates apply from the input toward the output of the circuit and are used to update the input side of the specification. The cost of a set of gates can be simply the MCT gate count or can be based on the quantum cost of implementing the MCT gates.

Multi-directional Algorithm [22] In the multi-directional algorithm for each row i every row $k, i \leq k \leq 2^n - 1$, is considered by mapping both the input and the output patterns to i thereby potentially adding gates to both the input and the output side of the circuit. The algorithm chooses the row k where the mapping has the lowest quantum cost and in the case of a tie the first row k where the mapping results in a function closest to the identity. To see that this algorithm subsumes the previous two note that the basic algorithm is simply the case of only considering row i, while the bidirectional algorithm is the case of only considering two cases: row i and row j where $f(j) = i$.

Search Algorithm It is interesting to consider whether searching can improve upon the above methods. The method given in Algorithm 2 is a simple branch-and-bound search based on the idea behind the multi-directional algorithm.

Search is a recursive procedure with parameters: f the function under consideration, k the row in the truth table of f under consideration, and $glist$ the circuit (list of gates) so far. For the initial call to Search, f should be the function to be realized, $k = 0$, and $glist$ should be empty. The cost of a circuit can be the number of gates or its quantum cost as discussed in Sect. 6. The cost of the best circuit found to date is used to bound the search. $BestCircuit$ and $BestCost$ are globals. Before starting a search, we use the multi-directional algorithm to find the initial $BestCircuit$ and $BestCost$. This is more efficient than setting the initial cost estimate to ∞. Note that the algorithm is presented with some obviously redundant computation for clarity. The actual implementation is more efficient.

Lines 2–4 skip rows in the truth table of f that are already in identity form, i.e., $f_k = k$. Lines 5–8 check if k has reached the end of f which must thus be the

Algorithm 2 Transformation-based search method

```
 1: procedure SEARCH(f, k, glist)
 2:     while k < N − 1 and f_k = k do
 3:         k ← k + 1
 4:     end while
 5:     if k = 2^n − 1 then
 6:         if cost of glist < BestCost then
 7:             record glist as BestCircuit and its cost as BestCost
 8:         end if
 9:     else
10:         for k ≤ j ≤ 2^n − 1 do
11:             G_in ← map(j, k)
12:             G_out ← map(f_j, k)
13:             apply G_in and G_out to map f to g
14:             dist[j] ← Δ(g)
15:         end for
16:         min ← minimum value in dist
17:         if cost(glist) + min ∗ α_0 < BestCost then
18:             for k ≤ j ≤ 2^n − 1 do
19:                 if dist[j] ≤ min ∗ α_1 then
20:                     G_in ← map(j, k)
21:                     G_out ← map(f_j, k)
22:                     apply G_in and G_out to map f to g
23:                     Search(g, k + 1, G_in||glist||reverse(G_out))
24:                 end if
25:             end for
26:         end if
27:     end if
28: end procedure
```

identity and $glist$ is a completed circuit for the original function. If it is less costly than the best circuit found to date, it is recorded as the best circuit.

Lines 10–16 consider each of the rows j from k through $2^n - 1$ where n is the number of function variables. In each case, the input side j and output side f_j are mapped to k and the gates are applied to map f to a resulting function g. Gates in G_{in} are applied to the input side of f and gates in G_{out} are applied to the output side of f. The idea of trying all j, $k \leq j \leq 2^n - 1$, is carried over from the multi-directional method. For each g, the operator Δ computes the Hamming distance from g to the identity function which is the sum of the Hamming distances between r and g_r for each row of g. The minimum distance is recorded in min.

Line 17 selects whether to continue based on the formula $cost(glist) + min * \alpha_0 < BestCost$ which is estimating the cost of finishing the current circuit based on its cost to date and the minimum Hamming distance found. The factor α_0 is discussed below.

If line 17 determines continuation, lines 18–25 go back through the rows from k to $2^n - 1$. For each, if $dist[j] \leq min * \alpha_1$, g is computed (the factor α_1 is discussed below) and a recursive call is made to Search with parameters $g, k + 1$ (the next row

to consider) and the circuit to date which is $glist$ with the gates from G_{in} prepended to the front and the gates from G_{out} reversed and appended to the end. The latter set of gates are reversed because procedure map generates gates from output towards the input when considering mapping an output pattern.

The factor α_0 controls the weight min is giving in estimating the cost of the final circuit. By experiment using NCV quantum cost, we have found 1.33 to be a good value. α_1 in line 19 determines how far $dist[j]$ can be above the minimum for row j to be considered as a basis for further searching. Again by experiment, we have determined that $\alpha_1 = 2$ is effective. More experiments with the search procedure may well lead to a better understanding of the best values for α_0 and α_1 and their interaction.

5 Simplifying a Reversible Circuit

We employ the following simplification rules for MPMCT gates developed by Rahman and Rice [15]. Note that these rules are referred to as templates in [15], but we choose not to call them that to avoid confusion with other formulations of templates in the reversible and quantum circuit literature. Note that rule 1 is the special case of rule 3 with $C = \phi$.

Rahman and Rice Simplification Rules

1. $T(x_c, x_t)T(\overline{x_c}, x_t) = T(x_t) = NOT(x_t)$
2. $T(C, x_t)T(C, x_t) = I$
3. $T(C \cup x_i, x_t)T(C \cup \overline{x_i}, x_t) = T(C, x_t)$
4. (a) $T(C \cup x_i \cup \overline{x_j}, x_t)T(C \cup \overline{x_i} \cup x_j, x_t) = T(x_i, x_j)T(C \cup x_j, x_t)T(x_i, x_j)$
   (b) $T(C \cup \overline{x_i} \cup \overline{x_j}, x_t)T(C \cup x_i \cup x_j, x_t) = T(x_i, x_j)T(C \cup \overline{x_j}, x_t)T(x_i, x_j)$
5. (a) $T(C \cup x_i, x_t)T(C \cup x_j, x_t) = T(x_i, x_j)T(C \cup x_j, x_t)T(x_i, x_j)$
   (b) $T(C \cup \overline{x_i}, x_t)T(C \cup \overline{x_j}, x_t) = T(x_i, x_j)T(C \cup x_j, x_t)T(x_i, x_j)$
   (c) $T(C \cup \overline{x_i}, x_t)T(C \cup x_j, x_t) = T(x_i, x_j)T(C \cup \overline{x_j}, x_t)T(x_i, x_j)$
6. (a) $T(C, x_t)T(C \cup x_i, x_t) = T(C \cup \overline{x_i}, x_t)$
   (b) $T(C, x_t)T(C \cup \overline{x_i}, x_t) = T(C \cup x_i, x_t)$
7. (a) $T(C \cup x_i \cup x_j, x_t)T(C \cup x_k, x_t) = T(x_i \cup x_j, x_k)T(C \cup x_k, x_t)T(x_i \cup x_j, x_k)$
   (b) $T(C \cup x_i \cup x_j, x_t)T(C \cup \overline{x_k}, x_t) = T(x_i \cup x_j, x_k)T(C \cup \overline{x_k}, x_t)T(x_i \cup x_j, x_k)$
   (c) $T(C \cup \overline{x_i} \cup x_j, x_t)T(C \cup x_k, x_t) = T(\overline{x_i} \cup x_j, x_k)T(C \cup x_k, x_t)T(\overline{x_i} \cup x_j, x_t)$
   (d) $T(C \cup \overline{x_i} \cup x_j, x_t)T(C \cup \overline{x_k}, x_t) = T(\overline{x_i} \cup x_j, x_k)T(C \cup \overline{x_k}, x_t)T(\overline{x_i} \cup x_j, x_k)$
   (e) $T(C \cup \overline{x_i} \cup \overline{x_j}, x_t)T(C \cup x_k, x_t) = T(\overline{x_i} \cup \overline{x_j}, x_k)T(C \cup x_k, x_t)T(\overline{x_i} \cup \overline{x_j}, x_k)$
   (f) $T(C \cup \overline{x_i} \cup \overline{x_j}, x_t)T(C \cup \overline{x_k}, x_t) = T(\overline{x_i} \cup \overline{x_j}, x_k)T(C \cup \overline{x_k}, x_t)T(\overline{x_i} \cup \overline{x_j}, x_k)$

For each of the above rules, the substitution holds even if the order of the gates on the left hand side is reversed because they have a common target.

To apply the above simplification rules, we need to be able to determine if two gates can be moved to be adjacent if they are not already. Since a reversible circuit

is a cascade of gates, the key operation is to determine if two adjacent gates can be interchanged since moving gates is in fact a sequence of gate interchanges. Since our circuits have both positive and negative controls, checking whether two adjacent gates can be interchanged is more involved then the commonly used so-called moving rule [9].

Moving Rule for MPMCT Gates
Given two adjacent gates the following checks are applied in order:

1. If the two gates have a common control which is positive for one gate and negative for the other, the gates can be interchanged.
2. If the target and controls for one gate all serve as controls for the second gate, in which case the common controls must have equal polarities or (1) would have applied, the gates can be interchanged with the control for the second gate corresponding to the target of the first gate having negated polarity.
3. If the target of one gate is a control for the second, the gates cannot be interchanged, otherwise they can be interchanged.

6 Mapping a Reversible Circuit to a Quantum Circuit

The first step in mapping a reversible circuit to a quantum circuit is to replace each reversible gate by an implementation of that gate comprised of elementary quantum gates, NCV gates in this work. A NOT gate is both a reversible and an elementary quantum gate, so no substitution is required. The same is true for a CNOT with a positive control.

6.1 Negative Control CNOTs

A CNOT with a negative control is not an elementary quantum gate. Two possible substitutions are shown in Fig. 2. Substitution (a) has been used in earlier work. Here we use substitution (b) as only a single NOT needs to be added and it can in fact be placed on either side of the CNOT giving more flexibility for later simplification.

The situation here is complicated by the use of Theorem 3. If a circuit is being synthesized with the intent that the polarity for all gate controls will be flipped, then we want to avoid CNOTs with positive controls. We thus have the notion of a target CNOT control polarity when doing a circuit simplification.

Our procedure for dealing with a CNOT with incorrect control polarity is straightforward. The following steps are applied for each CNOT g_i with incorrect polarity control x_j.

1. We scan from g_i back towards the input to find a $NOT(x_j)$.

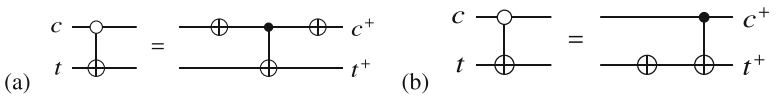

Fig. 2 Mapping a CNOT with a negative control to NCV gates

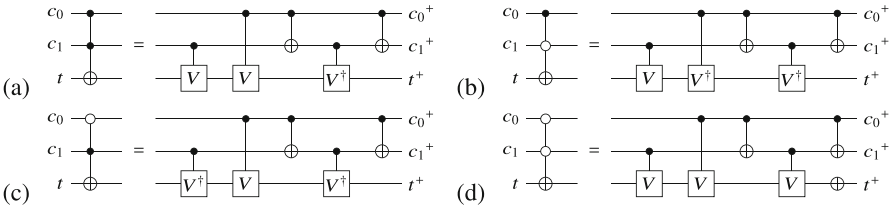

Fig. 3 NCV realization of Toffoli gates dependant on number and placement of negative controls

2. If none is found, we scan from g_i towards the output to find the required $NOT(x_j)$.
3. If a $NOT(x_j)$ is found, in either direction, it is moved across the circuit towards g_i inverting all controls it crosses until it has crossed over g_i.
4. If no $NOT(x_j)$ was found in 1 or 2, a NOT is inserted on the target line of g_i just before g_i and the polarity of the control for g_i is flipped.

6.2 NCV Realization of MPMCT Gates

For an MPMCT gate with two controls, the quantum implementation depends on the number and placement of negative controls as shown in Fig. 3. The difference between (b) and (c) is which of the controls is negative. Note that this only affects the assignment of V and $V^\dagger$ to the first two gates. For two negative controls, a sixth gate, a NOT, is required on the target line (t) as shown in (d).

Quantum realizations of MPMCT gates have been extensively studied beginning with the seminal paper by Barenco et al. [1]. To estimate quantum costs during our synthesis procedures, we use results from [18], which are given in Table 2. Each entry in the table is the number of elementary NCV quantum operations required to realize a gate with the associated number of controls.

The first three rows of the table for gates with 0, 1, and 2 controls are costed as described above. We are using substitution (b) for negative control CNOT gates. For three or more controls, a decomposition method is given in [18] which basically expresses a MPMCT gate as a network of gates with fewer controls. It is important to note that for three or more controls, the cost figures given in Table 2 assume one ancillary line is available. Reference [18] includes less expensive realizations if more ancillaries are available.

Table 2 NCV costs of MPMCT gates assuming one ancillary is available if needed

Controls	Negative controls								
	0	1	2	3	4	5	6	7	8
0	1								
1	1	2							
2	5	5	6						
3	14	14	16	18					
4	20	20	20	22	24				
5	32	32	32	34	36	38			
6	44	44	44	44	46	48	50		
7	64	64	64	64	66	68	70	72	
8	76	76	76	76	76	78	80	82	84

6.3 MPP and MPIP Gate Cost and Substitution

Consider Fig. 3 again. The control lines to a two-control MPMCT gate can be interchanged. Furthermore, since an MPMCT gate is self-inverse, the quantum circuit realization can be reversed. Now, since each of the circuits in Fig. 3 has a CNOT between the two MPMCT gate controls, it is clear that that gate will cancel a CNOT between c_1 and c_2 that follows it. The same is also true if the CNOT comes first. This is the basis for substituting the gate pair with an MPP or MPIP gate. One can see that an MPP or MPIP gate formed from an MPMCT with 0 or 1 negative controls thus has a cost of 4. If the MPMCT gate has two negative controls, the cost is 5.

MPP and MPIP gate substitution is straightforward. One need only scan the circuit to find an appropriate two input MPMCT and CNOT gate pair that can be moved together using the procedure described above. The two gates are replaced by a MPP or MPIP gate depending on which side the CNOT lies.

6.4 Overall Simplification and Mapping Strategy

The process applied to simplify a circuit has the following steps:

1. Apply Rahman and Rice MPMCT simplifications.
2. Apply the CNOT correction procedure to deal with any CNOTs that have the wrong control polarity.
3. Apply Rahman and Rice MPMCT simplifications.
4. If any changes to the circuit were made in 1–3, reverse the circuit and apply 1–3 to the result.
5. After iterating 1–4 until there are no changes, if the circuit is in reverse orientation, reverse it.
6. If MPP and MPIP gates are to be used, make all possible MPP and MPIP gate substitutions.

7 Experimental Results

We have implemented the methods described above in C and run our experiments on a x64-based PC with an Intel i5 650 processor and 3GB RAM. Tables 3 and 5 show the results for the 8! = 40,320 3-variable reversible functions for nine scenarios. The average quantum cost is shown for each scenario as well as the cpu seconds required.

Two versions of the search method are used. Search A has $\alpha_0 = \alpha_1 = 1$ and Search B has $\alpha_0 = 1.33$ and $\alpha_1 = 2$. Raising the α broadens the scope of the search, i.e., more potential circuits are considered, but that of course incurs increased computational cost.

Scenario (a) in Table 3 is presented to serve as a base line. The methods are applied with no function translations and none of the simplifications discussed in Sects. 5 and 6 including no use of MPP or MPIP gates. Results are then shown for (b) adding function inversion, (c) adding circuit simplification, and (d) adding the use of MPP and MPIP gates.

Table 4 shows the incremental improvements of scenarios (b), (c), and (d) compared to the base case (a). It is interesting to note that adding the use of the function inverse, scenario (b), has marginal effect on the search methods whereas adding the use of MPP and MPIP gates, scenario (d), significantly improves both search methods.

The scenarios in Table 5 all use the function inverse, circuit simplification, and MPP and MPIP gates. Scenarios (e), (f), and (g) show the results for adding use of the dual, input-output negation and input-output permutation separately. Scenarios (h) and (i) show the results for using the dual with permutation and input-output negation with permutation. Scenario (i) gives the best results across Tables 3 and 5.

Table 6 shows the improvements offered by each of scenarios (e) to (i) for each of the synthesis methods. For all methods, scenario (i) using input-output negation and input-output permutation gives the most improvement. This is not surprising as that scenario provides the most function translations to explore for each function. It is interesting that the improvement is not as high for the search methods as for the other three methods. That is because the search methods already explore an extensive solution space.

Table 7 is an analysis of scenario (e) in Table 5 and shows the number of functions for which each method gives the best result among the five synthesis methods. Search B exhibits the best performance but there are exceptions. Separate analysis shows that Search A finds a cheaper result than Search B for 5.1% of the functions. There are even 354 functions for which the Basic method finds the cheapest circuit. These anomalies are due to the heuristic nature of the five methods. Similar results are found for analyses of scenarios (f) to (i).

For the Search B method under scenario (e), 45.96% of the best circuits were found by synthesizing a circuit for $f(X)$, 32.22% by synthesizing a circuit for $f^{-1}(X)$, 11.71% by synthesizing a circuit for the dual of $f(X)$, and 10.11% by synthesizing a circuit for the dual of $f^{-1}(X)$. In total, the 40,320 circuits used

Table 3 Average quantum costs for three variable reversible functions: scenarios (a)–(d)

	(a) No inverse or simplification		(b) Inverse no simplification		(c) Inverse simplification		(d) Inverse simplify MPP MPIP gates	
Synthesis method	cost	cpu(s)	cost	cpu(s)	cost	cpu(s)	cost	cpu(s)
Basic	17.87	0.05	15.99	0.08	14.59	0.33	13.50	0.38
Bidirectional	16.55	0.06	15.55	0.09	14.58	0.29	13.28	0.32
Multi-direct.	16.51	0.30	15.35	0.59	14.41	0.77	13.09	0.82
Search A	14.77	4.92	14.51	9.87	13.60	21.55	11.66	12.66
Search B	14.32	34.55	14.25	68.10	13.40	105.42	11.28	43.00

Table 4 Improvements compared to scenario (a)

	Incremental improvement			
Synthesis method	(b)	(c)	(d)	Total
Basic	10.53%	7.83%	6.10%	24.46%
Bidirectional	6.04%	5.86%	7.85%	19.76%
Multi-direct.	7.03%	5.72%	7.97%	20.71%
Search A	1.76%	6.16%	13.13%	21.06%
Search B	0.49%	5.94%	14.80%	21.23%

Table 5 Average quantum costs for three variable reversible functions: scenarios (e)–(i)

Synthesis	(e) Dual		(f) Input-output negation		(g) Permutation		(h) Dual and permutation		(i) Input-output neg. and permutation	
method[a]	cost	cpu(s)	cost	cpu(s)	cost	cpu(s)	cost	cpu(s)	cost	cpu(s)
Basic	13.01	1.14	12.37	4.31	12.04	2.10	11.88	5.76	11.60	26.12
Bidirectional	12.63	1.00	11.97	3.65	11.84	1.76	11.65	4.82	11.31	21.68
Multi-direct.	12.48	1.86	11.85	7.75	11.72	4.78	11.54	10.97	11.21	46.15
Search A	11.49	37.78	11.33	135.16	10.94	74.40	10.90	181.84	10.84	819.35
Search B	11.16	160.04	11.07	648.62	10.76	261.93	10.73	762.61	10.69	3802.06

[a]Function inverse, circuit simplification and MPP and MPIP gates used in all scenarios.

Table 6 Improvements compared to scenario (d)

	Improvements				
Synthesis method	(e)	(f)	(g)	(h)	(i)
Basic	3.63%	8.39%	10.81%	12.00%	14.07%
Bidirectional	4.89%	9.89%	10.84%	12.27%	14.85%
Multi-direct.	4.66%	9.50%	10.47%	11.84%	14.39%
Search A	1.46%	2.84%	6.17%	6.52%	7.08%
Search B	1.06%	1.86%	4.61%	4.88%	5.23%

14,396 MPP and 13,275 MPIP gates. It is interesting to note that of those 27,671 gates, 33.15% had two positive controls, 10.7% had two negative controls, and 56.16% had one positive and one negative control. This demonstrates the usefulness of allowing MPP and MPIP gates rather than just Peres and inverse Peres gates.

Table 7 Best results for scenario (e) in Table 5

Synthesis method	Best result		Unique best result	
	No. of functions	% of total	No. of functions	% of total
Basic	13,968	34.64%	354	0.88%
Bidirectional	17,135	42.50%	122	0.30%
Multi-directional	18,349	45.51%	34	0.08%
Search A	30,082	74.61%	1744	4.33%
Search B	37,678	93.45%	8975	22.26%

Table 8 Search B for scenario (i)—distribution by function translation

Negation	Permutation index					No perm.	Subtotal	% of total
	0	1	2	3	4			
No function inversion								
No neg.	332	376	726	1913	2870	19,514	25,731	63.82%
1	17	18	20	56	65	137	313	0.78%
2	16	21	21	63	67	153	341	0.85%
3	14	15	17	65	67	223	401	0.99%
4	11	14	24	72	67	224	412	1.02%
5	15	21	26	86	88	355	591	1.47%
6	17	20	31	94	96	441	699	1.73%
Dual	19	22	33	131	135	1083	1423	3.53%
Subtotal	441	507	898	2480	3455	22,130	29,911	74.18%
% of total	1.09%	1.26%	2.23%	6.15%	8.57%	54.89%		
With function inversion								
No neg.	234	260	368	921	1248	4416	7447	18.47%
1	16	17	19	50	55	112	269	0.67%
2	14	15	19	43	49	115	255	0.63%
3	14	15	17	50	53	184	333	0.83%
4	9	12	19	63	58	174	335	0.83%
5	11	17	17	64	63	245	417	1.03%
6	15	18	25	79	73	284	494	1.23%
Dual	16	19	28	103	93	600	859	2.13%
Subtotal	329	373	512	1373	1692	6130	10,409	25.82%
% of total	0.82%	0.93%	1.27%	3.41%	4.20%	15.20%		

Table 8 shows the distribution of the best circuits found for the 40,320 three variable functions using method Search B for scenario (i) in Table 5.

To place the above results in some context, the authors of [2] considered the realization of three variable reversible function using mixed-polarity Toffoli gates and positive polarity Reed-Muller techniques. The best results they report have an average quantum cost of 13.36 which is better than our base line but higher than our best results. This illustrates further the advantage of using MPP and MPIP gates and the circuit simplification results discussed in this chapter.

Table 9 Selected *worst case* functions

Synthesis method	cost[a]	cost[b]	cost[c]	code	cpu(s)	cost[a]	cost[b]	cost[c]	code	cpu(s)
	3_17					*4_49*				
Basic	15	15	11	P3-I	0.00	80	68	43	P6-N5	0.06
Bidirectional	15	12	11	P3	0.00	101	87	31	P6-N11-I	0.04
Multi-direct.	15	12	11	P3	0.00	69	70	32	P6-N11	0.08
Search A	15	12	11	N3	0.01	55	52	32	P6-N11	3.55
Search B	15	12	11	P4-I	0.06	36	32	28	P17-N11	476.90
Best known cost	10 [28]					32 [28]				
	hwb4					*hwb5*				
Basic	71	53	32	P22-D	0.05	352	294	199	P119-N3-I	4.24
Bidirectional	64	52	21	P22-I	0.04	323	301	171	P116-N20-I	2.84
Multi-direct.	58	55	21	P22-I	0.07	313	282	172	P115-N3-I	5.35
Search A	49	41	21	P22-I	3.39	280	230	101	P110-N16-I	6909.33
Search B	27	21	20	P20-I	12.97	see note *d*				
Best known cost	19 [8]					71 [8]				

[a]Using inverse translation, no circuit simplification or MPP/MPIP gates
[b]Using inverse translation, circuit simplification and MPP/MPIP gates
[c]Using inverse, input-output negation, input-output permutation, circuit simplification and MPP/MPIP gates
[d]Search B for hwb5 is computationally prohibitive

De Vos and Van Rentergem [3] have presented a reversible circuit synthesis approach using Young-based subgroups. They consider circuits with positive and negative controls. A difference from the work here is that the control function for a gate can be any Boolean function not just the conjunction of controls. They allow CNOT gates with a negative control. They did not use Peres type gates. For 3-variable reversible functions, they report average gate counts of 5.88 for their Algorithm A, 4.21 for their Algorithm B and 3.73 as the optimal average. When we apply our methods allowing negative control CNOT gates, using MPP or MPIP gates and using the dual and choice of a circuit for f or f^{-1}, we find gate averages of 4.73 (Basic), 4.57 (Bidirectional), 4.50 (Multi-directional), 4.49 (Search A), and 4.50 (Search B). Note that this is a very rough comparison as we are using Peres type gates and all our gates use a conjunctive control function.

Table 9 shows the results for four functions which have been described as *worst cases* for several synthesis methods [28]. They are certainly known to be difficult for transformation-based synthesis. Note that the results show the cost of the best circuit when considering the synthesis of f and f^{-1} and using input-output negation and input-output permutation. All simplification techniques discussed in Sects. 5 and 6 are applied including the use of MPP and MPIP gates.

In Table 9, the function translation resulting in the best circuit found for each case is shown in column trans. P-x indicates input-output permutation has been applied where x denotes the permutation index as defined in [12]. N-x indicates

Table 10 Search B applied to hwb5 for four function translations

Translation	cost	Solution cpu(s)	Total cpu(s)
Function	120	2538.43	3600.00
Inverse	85	2444.29	2565.57
Dual	144	1503.20	3600.00
Inverse dual	152	614.17	3600.00

$T(d,e)T(e,d)T(b,e)T(a,b)T(d,b)T(a,d)T(a,e)T(d,e,a)T(a,b)T(c,e,b)P(c,\overline{d},a)$
$T(a,c)T(\underline{a},e)T(b,a)\underline{T}(a,d)T(b,e,a)T(a,\underline{b})T(a,d)T(b,d,a)T(a,d)T(b,c,a)T(a,e,d)$
$T(a,b)T(\overline{b},e,c)T(a,\overline{b},d,e)T(a,c,d)IP(\overline{b},e,c)T(d,b)T(c,\overline{e},a)T(a,e)$

Fig. 4 hwb5 circuit found using Search B: 30 gates, quantum cost 85

input-output negation has been applied, after possible permutation, with the 1's in the binary expansion of x indicating which input-output positions are negated. D is used to indicate all input-output pairs are negated, i.e., the dual is used. Lastly, I indicates the function is inverted following any permutation and negation.

Method Search B is computationally much more expensive than the other methods. For that reason, applying Search B to hwb5 is omitted in Table 9. Instead, Table 10 gives the results for applying Search B to hwb5 for four scenarios: the function, the inverse of the function, the dual of the function, and the dual of the inverse of the function. For each scenario, a limit of 1 h CPU time was imposed.

The best circuit was found using the inverse function. Interestingly that scenario did not hit the 1 h time limit which means the full search was completed. The circuit found using the inverse of hwb_5 is shown in Fig. 4. Note that it uses MPMCT gates with negative controls as well as an MPP and an MPIP gate. This circuit is far better than the results reported for hwb_5 in Table 9 but finding it required very lengthy computation time.

This result is reasonably close to the best circuit found to date, cost 71, which is listed on Maslov's benchmark web site [8]. That circuit was posted by the authors of [26] which presented a variable-length chromosome evolutionary algorithm for reversible circuit synthesis. The cpu usage required to find the circuit is not reported. It is interesting that a relatively simplistic transformation-based synthesis approach can produce so good a circuit especially compared to other techniques (see [8, 28] for examples of circuits for hwb5).

8 Heuristic Selection of Function Translations

The results presented in the previous section show that the use of input-output negation and input-output permutation can significantly reduce circuit cost. However it is clear that searching through all possible translations quickly becomes impractical as n, the number of variables, increases since there are 2^n input-output negations and $n!$ input-output permutations. The ideal would be able to pick a small number

Table 11 Selected translation scenarios for 3-variable functions using the basic method

	Translations	Cost	% Impr. over (a)	cpu(s)
1	Inverse (a)	13.50		0.37
2	Inverse, dual	13.04	3.47%	0.98
3	Inverse, HION	12.80	5.23%	1.42
4	Inverse, full I-O negation	12.37	8.41%	4.24
5	Inverse, HIOP	12.52	7.29%	0.73
6	Inverse, all I-O perm.	12.04	10.86%	2.05
7	Inverse, HION, HIOP	12.11	10.32%	2.91
8	Inverse, all I-O neg., all I-O perm.	11.60	14.09%	25.17

of translations to consider based on properties of the function to be synthesized. As a start toward that goal, we here present two heuristic methods for choosing which translations to consider.

Heuristic Input-Output Negation (HION) We use the following:

1. No input-output negations applied.
2. All input-output pairs negated, i.e., the dual.
3. The α-translation, see Definition 9, where α_i is the identity if $x_i = 0$ and α_i is NOT if $x_i = 1$ in the earliest assignment to $(x_0, x_1, \ldots, x_{n-1})$ where the Hamming distance between that assignment and the corresponding output assignment $(x_0^+, x_1^+, \ldots, x_{n-1}^+)$ is minimal where *earliest* refers to considering input assignments starting from $(0, 0, \ldots, 0)$ in truth table order.

Heuristic Input-Output Permutation (HIOP) We employ two permutations:

1. The inputs and outputs in the order given, i.e., the null permutation.
2. The reverse permutation where the inputs and outputs are in the reverse of the order given, e.g., x_0, x_1, x_2 is permuted to x_2, x_1, x_0.

Table 11 presents results of applying the basic transformation-based synthesis method to the 40,320 3-variable reversible functions for a variety of function translation scenarios. Circuit simplification and MPP and MPIP gates are used in all cases.

The top row of Table 11 uses only the function inverse translation and is intended as a baseline for measuring the improvement offered by the other scenarios. The function inverse translation is employed in all the other scenarios.

Lines 2–4 show the results for using the dual, heuristic input-output negation, and for comparison using all eight possible input-output negations. Lines 5 and 6 compare using heuristic input-output permutation and all six possible input-output permutations. Lastly, rows 7 and 8 compare using both heuristic translation approaches with using all function translations. In all cases, improvement is measured relative to just using the function inversion translation and shows the percentage reduction in the circuit quantum cost.

The results indicate that while our heuristic methods show quite good improvement, they, as one would expect for such simple approaches, fall well short of employing all function translations. Applying the basic synthesis method and heuristic translation approaches to the worst case function 3_17 reduces the quantum cost from 15 down to 11 for the function inverse, reverse permutation, and no input-output negation. However, applying the techniques to the other three worst case functions offers no improvement over just using only the function inverse translation. That is simply the nature of those functions but does suggest that the heuristic techniques described in this section can likely be improved upon.

9 Discussion and Future Work

This chapter has considered a number of aspects of the synthesis and simplification of reversible circuits with positive and negative controls. The main contributions are the function translation techniques of input-output negation and input-output permutation introduced in Sect. 4, the employment of mixed control Peres gates, and various circuit simplifications discussed in Sects. 5 and 6 which combine well with Rahnan and Rice's MPMCT techniques. We plan to investigate whether the circuit rewriting rules proposed in [21] can improve our circuit simplification procedures.

The experimental results presented in this chapter demonstrate that the input-output negation and input-output permutation function translations introduced here have significant potential for improving the synthesis of reversible circuits. However, the approach of searching all possible translations is clearly limited. We have suggested some initial heuristic techniques to choose translations to consider by examining the function to be synthesized. The results show some promise, but it is very likely that incorporating the choice of function translations into the synthesis process will be more effective than making a pre-synthesis choice. This is an area requiring further research.

The transformation-based search method is of potential interest but is truly a work in progress. More work is needed to determine how best to bound the search in order to make it computationally feasible. In addition, the authors of [6, 20] have incorporated Fredkin gates [5], which are controlled swap gates, into transformation-based synthesis. It would be interesting to see how their approaches might be incorporated into our work and how Fredkin gates might be extended to be mixed-polarity multiple control gates.

The work thus far has been limited to considering transformation-based synthesis methods. It would be very interesting to see how effective the function translations input-output negation and input-output permutation are when combined or integrated into other reversible circuit synthesis methods.

This work has shown that mixed polarity Peres and inverse Peres gates can be quite effective in reducing quantum circuit cost. The approach here has been to consider the introduction of such gates into a circuit as part of the circuit simplification process. It would be interesting to consider integration of these gate

types into the synthesis process itself both for transformation-based and other synthesis methods.

The comparison of the methods presented in this chapter to the work of De Vos and Van Rentergem [3], despite the differences in gate types permitted, suggests there is room for further improvement of our methods.

The quantum cost used in this chapter is based on the NCV gate library. Future work should consider cost metrics for fault-tolerant quantum circuits using the Clifford+T quantum gate library [13]. Another area to consider would use Qiskit or ProjectQ [7] to map an MCMPT circuit into an NISQ compatible circuit and then count gates after performing optimizations at the quantum gate level.

References

1. Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, M., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H.: Elementary gates for quantum computation. Phys. Rev. A **52**(5), 3457–3467 (1995)
2. Cheng, C.S., Singh, A.K., Gopal, L.: Efficient three variables reversible logic synthesis using mixed polarity Toffoli gate. Procedia Comput. Sci. **70**, 362–368 (2015)
3. De Vos, A., Van Rentergem, Y.: Young subgroups for reversible computers. Adv. Math. Commun. **2**, 183–200 (2008)
4. Feynman, R.: Quantum mechanical computers. Optic News (1985), pp. 11–20
5. Fredkin, E., Toffoli, T.: Conservative logic. Int. J. Theor. Phys. **21**, 219–253 (1982)
6. Handique, M., Singha, R.: A modified transformation-template based synthesis using Fredkin/swap gates in reversible circuits. Procedia Comput. Sci. **125**, 801–809 (2018)
7. LaRose, R.: Overview and comparison of gate level quantum software platforms. Quantum **3**, 130 (2019)
8. Maslov, D.: Reversible logic synthesis benchmarks page. http://www.cs.uvic.ca/~dmaslov/
9. Miller, D.M., Maslov, D., Dueck, G.W.: A transformation-based algorithm for reversible logic synthesis. In: Proceedings of IEEE/ACM Design Automation Conference (DAC), pp. 318–323 (2003)
10. Moraga, C.: Using negated control signals in quantum computing circuits. Facta Universitatis(Nis) Ser. Electr. Energy **24**(3), 423–435 (2011)
11. Moraga, C.: Mixed polarity reversible Peres gates. Electron. Lett. **50**(14), 987–989 (2015)
12. Myrvold, W., Ruskey, F.: Ranking and unranking permutations in linear time. Inf. Process. Lett. **79**, 281–284 (2001)
13. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
14. Peres, A.: Reversible logic and quantum computers. Phys. Rev. A **32**(6), 3266–3276 (1985)
15. Rahman, M.Z., Rice, J.E.: Templates for positive and negative control Toffoli networks. In: Lecture Notes in Computer Science, vol. 8507 (2014)
16. Ribeiro, A., Kowada, L., Marquezino, F., Figueiredo, C.: A New reversible circuit synthesis algorithm based on cycle representations of permutations. Electron. Notes Discrete Math. **50**, 187–192 (2015)
17. Saeedi, M., Markov, I.L.: Synthesis and optimization of reversible circuits - A survey. CoRR abs/1110.2574 (2011). http://arxiv.org/abs/1110.2574
18. Sasanian, Z., Miller, D.M.: NCV realization of MCT gates with mixed controls. In: Proceedings of 2011 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 567–571 (2011)

19. Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Reversible logic circuit synthesis. In: ICCAD. San Jose, California, USA, pp. 125–132 (2002)
20. Soeken, M., Chattopadhyay, A.: Fredkin-enabled transformation-based reversible logic synthesis. In: Proceedings of the International Symposium on Multiple-Valued Logic, pp. 60–65 (2015)
21. Soeken, M., Thomsen, M.K.: White dots do matter: rewriting reversible logic circuits. In: Lecture Notes in Computer Science, vol. 7948 (2013)
22. Soeken, M., Dueck, G.W., Rahman, M.M., Miller, D.M.: An extension of transformation-based reversible and quantum circuit synthesis. In: Proceedings of the International Symposium on Circuits and Systems, pp. 2290–2293 (2016)
23. Soeken, M., Tague, L., Dueck, G.W., Drechsler, R.: Ancilla-free synthesis of large reversible functions using binary decision diagrams. J. Symb. Comput. **73**, 1–26 (2016)
24. Szyprowski, M., Kerntopf, P.: Low quantum cost realization of generalized Peres and Toffoli gates with multiple-control signals. In: 2013 13th IEEE International Conference on Nanotechnology (IEEE-NANO 2013), pp. 802–807 (2013)
25. Toffoli, T.: Reversible computing. Tech memo MIT/LCS/TM-151, MIT Lab for Comp. Sci (1980)
26. Wang, X., Jiao, L., Li, Y., Qi, Y., Wu, J.: A variable-length chromosome evolutionary algorithm for reversible circuit synthesis. Multiple-valued Logic Soft Comput. **25**, 643–671 (2015)
27. Wille, R., Drechsler, R.: BDD-based synthesis of reversible logic for large functions. In: Proceedings of the Design Automation Conference, pp. 270–275 (2009)
28. Wille, R., Große, D., Teuber, L., Dueck, G.W., Drechsler, R.: RevLib: An online resource for reversible functions and reversible circuits. In: Int'l Symposium on Multi-Valued Logic, pp. 220–225 (2008). RevLib is available at www.revlib.org
29. Yamashita, S., Minato, S., Miller, D.M.: Synthesis of semi-classical quantum circuits. Multiple-Valued Logic Soft Comput. **18**(1), 99–114 (2012)