Synthesis of Reversible Circuits with Minimal Lines for Large Functions

Mathias Soeken[1] Robert Wille[1] Christoph Hilken[1] Nils Przigoda[1] Rolf Drechsler[1,2]
[1]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{msoeken,rwille,chilken,przigoda,drechsle}@informatik.uni-bremen.de

Abstract— Reversible circuits are an emerging technology where all computations are performed in an invertible manner. Motivated by their promising applications, e.g. in the domain of quantum computation or in the low-power design, the synthesis of such circuits has been intensely studied. However, how to automatically realize reversible circuits with the minimal number of lines for large functions is an open research problem.

In this paper, we propose a new synthesis approach which relies on concepts that are complementary to existing ones. While "conventional" function representations have been applied for synthesis so far (such as truth tables, ESOPs, BDDs), we exploit *Quantum Multiple-valued Decision Diagrams* (QMDDs) for this purpose. An algorithm is presented that performs transformations on this data-structure eventually leading to the desired circuit. Experimental results show the novelty of the proposed approach through enabling automatic synthesis of large reversible functions with the minimal number of circuit lines. Furthermore, the quantum cost of the resulting circuits is reduced by 50% on average compared to an existing state-of-the-art synthesis method.

I. Introduction

Reversible computation is an emerging technology that has established itself as a promising research area. In reversible circuits all computations are performed in an invertible manner, i.e. bijections are realized. This reversibility opens up new prospects for computation technology. For example, the domain of quantum computation – a new way of information processing which enables to solve certain problems exponentially faster compared to conventional methods [1] – profits from enhancements in this area, because every quantum circuit inherently is reversible. For low-power design, reversible logic offers interesting advantages since almost zero power dissipation will only be possible if computation is reversible [2, 3].

Motivated by these applications, researchers started to develop new design methods for such circuits. The number of circuit lines is a major criterion. This is particularly caused by the fact that, in the domain of quantum computation, each circuit line is represented by so called qubits – a highly limited resource. Furthermore, the number of lines has a close relation to the reliability of the circuit. Thus, it is well-accepted that the number of lines in reversible circuits should be kept as small as possible.

Accordingly, synthesis of reversible circuits focused on determining realizations with the minimal number of lines. To this end, the function to be synthesized has been represented in terms of permutations, truth-tables, or similar descriptions (see e.g. [4, 5]). Using such function representations, minimality of the circuit lines can easily be ensured. However, these approaches suffer from the poor scalability caused by the exponential growth of the respective data-structures. As a result, only small functions can be synthesized with them.

In order to overcome this limitation, approaches exploiting more compact function representations, namely *Exclusive Sum of Products* (ESOP) or *Binary Decision Diagrams* (BDDs), have been introduced [6, 7]. While these methods enable synthesis of large functions, they generate circuits whose number of lines is way beyond the optimum[1]. Optimization approaches aiming at the reduction of the number of circuit lines have been proposed to address this drawback [9]. However, until today only synthesis approaches exist that either guarantee the minimality of the number of circuit lines but are not scalable or enable synthesis of large functions at the expense of a high amount of additional circuit lines.

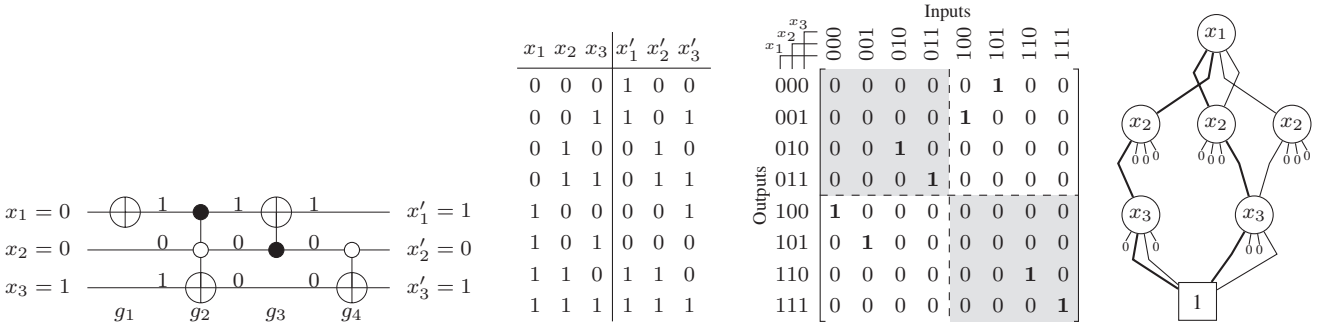
In this paper, a synthesis approach is proposed that provides a compromise between these contradictory properties. Instead of applying non-scalable function descriptions (permutations, truth-tables) or data-structures not directly aimed at the representation of reversible functions (ESOPs, BDDs), we make use of *Quantum Multiple-valued Decision Diagrams* (QMDDs) [10]. QMDDs are tree-like data-structures that offer a compact representation for permutation matrices. Since permutation matrices are used to describe reversible functions, QMDDs are an ideal data-structure to efficiently store and manipulate them. In fact, many relevant reversible functions can be represented in polynomial space using QMDDs, while e.g. truth tables always require an exponential amount.

Given a QMDD that represents the function to be synthesized, the general idea of the proposed approach is to apply reversible gate operations so that every vertex of the tree is transformed into a corresponding identity structure. Then, these gates can be composed into a circuit realizing the given function. The respective transformations are not trivial; however, we show that basically the application of two transformation rules lead to the desired results.

Overall, a synthesis approach is introduced that relies on concepts that are complementary to existing ones. As confirmed by experimental evaluations, this enables the automatic synthesis of large functions with the minimal number of circuit lines for the first time. Furthermore, the quantum cost of the resulting circuits are reduced by 50% on average compared to an existing state-of-the-art synthesis method.

[1]In fact, the minimal number of circuit lines for large functions was unknown until recently, but is now available for a selection of them in [8].

978-1-4673-0772-7/12/$31.00 ©2012 IEEE



x_1	x_2	x_3	x'_1	x'_2	x'_3
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	0	0	1
1	0	1	0	0	0
1	1	0	1	1	0
1	1	1	1	1	1

	000	001	010	011	100	101	110	111
000	0	0	0	0	0	**1**	0	0
001	0	0	0	0	**1**	0	0	0
010	0	0	**1**	0	0	0	0	0
011	0	0	0	**1**	0	0	0	0
100	**1**	0	0	0	0	0	0	0
101	0	**1**	0	0	0	0	0	0
110	0	0	0	0	0	0	**1**	0
111	0	0	0	0	0	0	0	**1**

(a) Reversible circuit (b) Truth table (c) Permutation matrix (d) QMDD

Fig. 1. Reversible circuit and its function representations

The remainder of this paper is structured as follows. The next section briefly reviews the core concepts of reversible circuits as well as the applied QMDD data-structure. Afterwards, Section III introduces the general idea as well as the main concepts of the proposed approach before the actual algorithm is described in detail in Section IV. Section V shows the correctness and completeness of the approach. Finally, Section VI reports experimental results, while Section VII concludes the paper and provides an outlook on future work.

II. Background

To keep this paper self-contained, the following section briefly reviews the basics on reversible functions and circuits. Afterwards, the QMDD data-structure is introduced which is utilized to compactly represent and synthesize reversible functions.

A. Reversible Functions and Circuits

A Boolean function $f : \mathbb{B}^r \to \mathbb{B}^r$ is *reversible* if it is bijective, i.e. if each input pattern is uniquely mapped to a corresponding output pattern. The *synthesis problem* is defined as the task of determining a reversible circuit for a given function f.

Reversible circuits differ from conventional circuits, since e.g. fanout and feedback are not directly allowed [1]. Usually, they are built as a cascade of reversible gates including e.g. the Toffoli gate [11], the Fredkin gate [12], or the Peres gate [13]. In this paper, we focus on circuits composed of Toffoli gates.

Definition 1 *Let $X = \{x_1, \ldots, x_r\}$ be a set of variables or lines. Then, a reversible circuit is described as a cascade $g_1 \ldots g_d$. A gate $g_i = (C_i, t_i)$, $i \in \{1, \ldots, d\}$, is a tuple of a set $C_i \subset \{x^\varrho \mid x \in X, \varrho \in \{-, +\}\}$ of (positive and negative)* control lines *and a* target line *$t_i \in X$ with $\{t_i^-, t_i^+\} \cap C_i = \emptyset$. The target line t_i of a Toffoli gate is inverted if and only if all positive (negative) control lines evaluate to one (zero). The values of all remaning lines are passed through the gate unaltered. That is, the Toffoli gate maps $(x_1, \ldots, x_{t_i}, \ldots, x_r)$ to $(x_1, \ldots, \bigwedge_{x \in C_i} \dot{x} \oplus x_{t_i}, \ldots, x_r)$ with $\dot{x} = x$ for any x^+ and $\dot{x} = \overline{x}$ for any x^-.*

Example 1 *Fig. 1(a) shows a reversible circuit with three lines and composed of four gates. The target lines are denoted by $\bigoplus$, while a $\bullet$ represents a positive control line and a $\circ$ represents a negative control line. For example, assigning the input pattern 001 to the circuit results in the output pattern 101. Due to the reversibility, this computation can be performed in both directions.*

The cost of reversible circuits is usually measured by *quantum cost* as introduced by Barenco et al. in [14]. The quantum cost of a single reversible gate depends on the number of control lines. For example, a Toffoli gate with one or no positive control line has quantum cost of 1, while a Toffoli gate with two positive control lines has quantum cost of 5. In general, the quantum cost of a Toffoli gate with $|C_i|$ positive control lines amount to at most $2^{|C_i|+1} - 3$, i.e. this value increases exponentially in the worst case. If negative control lines occur, the same cost metric is applied except for the case where the Toffoli gate is entirely composed of negative controls. Then, the cost is increased by two [15].

Besides quantum cost, also the number of circuit lines is an important metric. Since for quantum computation each circuit line is represented by qubits, this value should be kept as small as possible. In this paper, we consider synthesis of reversible circuits with the minimal number of circuit lines.

B. Quantum Multiple-valued Decision Diagrams

Common canonical representations for reversible functions are truth tables and permutation matrices. In the following, these representations are briefly reviewed leading to the more compact QMDD data-structure which is utilized in this paper.

A reversible function with r variables describes a permutation σ of the set $\{0, \ldots, 2^r - 1\}$. This permutation can also be described using a *permutation matrix*, i.e. a $2^r \times 2^r$ matrix $F = [f_{i,j}]_{2^r \times 2^r}$ with $f_{i,j} = 1$ if $i = \sigma(j)$ and 0 otherwise, for all $i = 0, \ldots, 2^r - 1$. That is, each column (row) of the matrix represents one possible input pattern (output pattern) of the function. If $f_{i,j} = 1$, then the input pattern in column j maps to the output pattern in row i.

Example 2 *The truth table of the function realized by the circuit in Fig. 1(a) is given in Fig. 1(b). From this truth table, the permutation matrix shown in Fig. 1(c) is obtained.*

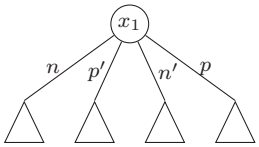


Fig. 2. QMDD and edge labels

As can be seen, the size of a permutation matrix grows exponentially with respect to the number of input/output variables. However, QMDDs [10] provide an efficient data-structure which enables a much more compact representation of permutation matrices.

Definition 2 *A QMDD is a directed acyclic graph composed of*

- *two terminal vertices labeled by* $\boxed{0}$ *and* $\boxed{1}$ *representing the matrix* $[0]_{1\times1}$ *and the matrix* $[1]_{1\times1}$*, respectively, and*
- *non-terminal vertices labeled by a primary input* x_i *and representing sub-matrices.*

The QMDD structure is based on recursively partitioning a $2^r \times 2^r$ *matrix* F *into four sub-matrices, each of dimension* $2^{r-1} \times 2^{r-1}$*. Accordingly, each non-terminal vertex has four outgoing labeled edges targeting child vertices representing from left to right (see also Fig. 2)*

- *the top-left sub-matrix where* x_i *maps from 0 to 0 (edge is denoted as* negative *and labeled* n*),*
- *the top-right sub-matrix where* x_i *maps from 1 to 0 (edge is denoted as* pseudo-positive *and labeled* p'*),*
- *the bottom-left sub-matrix where* x_i *maps from 0 to 1 (edge is denoted as* pseudo-negative *and labeled* n'*), and*
- *the bottom-right sub-matrix where* x_i *maps from 1 to 1 (edge is denoted as* positive *and labeled* p*).*

Equivalent sub-matrices are shared, i.e. the respective vertices have more than one parent. These basic concepts of QMDDs are clarified in the following example.

Example 3 *A QMDD representing the permutation matrix in Fig. 1(c) is shown in Fig. 1(d). For clarity, edges to a* $\boxed{0}$*-terminal are indicated as stubs. The root vertex of this QMDD represents the overall matrix. Since this vertex is labeled with* x_1*, the child-vertices partition this matrix into four sub-matrices with respect to* x_1*. That is, the first successor represents all input/output mappings where* x_1 *maps from 0 to 0 (i.e. the top-left sub-matrix is represented), the second successor represents all input/output mappings where* x_1 *maps from 1 to 0 (i.e. the top-right sub-matrix is represented), and so on. Since the first and the fourth sub-matrix are equal (see shaded boxes in Fig. 1(c)), the respective vertices are shared. Furthermore, consider the leftmost vertex labeled with* x_2 *representing the top-right sub-matrix. This vertex partitions this sub-matrix further (now with respect to* x_2*). Since the last three sub-matrices of these vertices are entirely assigned to 0, the corresponding edges directly point to the* $\boxed{0}$*-terminal. A similar scheme is applied for all remaining vertices.*

An important property of a QMDD is that all paths from the root vertex to an $\boxed{1}$-terminal have the same length and traverse the vertex labels in the same order. Thus, each of these paths can be referenced by the unique sequence of the traversed edge labels. Given a QMDD F, we call a path π from the root vertex to an $\boxed{1}$-terminal an *1-path* and denote it as $\pi \in F$. An empty path is referred to as ε. Using 1-paths, the function represented by the QMDD can be determined as illustrated by the following example.

Example 4 *Consider again the QMDD depicted in Fig. 1(d). The path* npn *(marked bold starting from the first outgoing edge of* $\textcircled{x_1}$ *in Fig. 1(d)) represents the mapping of the input assignment* 010 *to the output assignment* 010*. Since no pseudo edges are in this path, the inputs and the outputs are equal. In contrast, the path* $p'np'$ *(also marked bold in Fig. 1(d)) represents the mapping of the input assignment* 101 *to the output assignment* 000*. Here, a pseudo-positive mapping is applied to* x_1 *and* x_3*, i.e. these values map from 1 to 0. This is consistent with the input/output mappings in the original truth table (see Fig. 1(b)) of that function.*

Overall, permutation matrices, and, therefore, reversible functions, can efficiently be represented using QMDDs. Moreover, e.g. due to the sharing of vertices, QMDDs are much more compact than exponentially large truth table- or matrix-representations. Thus, they enable the treatment of significantly larger functions. Furthermore, operations (e.g. the application of a Toffoli gate to a given function) can easily be performed on the QMDD data-structure. For a more detailed treatment on QMDDs, we refer to [10].

III. Main Concepts of the Synthesis Approach

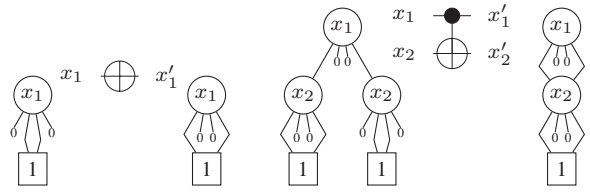
This section presents the main concepts of the proposed synthesis methodology including the general idea as well as the applied transformation rules. While this covers the basic concepts, the actual algorithm is provided in the next section.

A. General Idea

The task of the proposed synthesis approach is to determine a reversible circuit $g_1 \dots g_d$ representing the function given in terms of a QMDD F. The respective gates of the desired circuit can be represented by permutation matrices and, thus, also by QMDDs $T_1, \dots, T_d$. Since the composition of a reversible function with its inverse leads to the identity function (i.e. since $F \times F^{-1} = I$), the synthesis problem can be formulated as the search for Toffoli gates $T_1, \dots, T_d$ so that $F \times T_d \times \dots \times T_1 = I$. In other words, the main goal of the proposed synthesis approach is to determine a sequence of Toffoli gates so that their application to F leads to a QMDD representing the identity.

Clearly, the determination of the respective Toffoli gates is crucial. However, the regular structure of a QMDD representing the identity can be exploited for this purpose. Consider for example Fig. 3 showing a QMDD representing the identity function over two variables. As can be seen, all p'- and n'-edges of this QMDD point to the $\boxed{0}$-terminal, while all n- and p-edges always point to the same child vertex (this is because

Fig. 3. QMDD representing the identity matrix with two variables



(a) Applying a NOT gate (b) Applying a CNOT gate

Fig. 4. Simple QMDD modifications

only 0 to 0 and 1 to 1 mappings occur in an identity matrix). Accordingly, Toffoli gates should be determined in such a way that they establish this structure throughout a given QMDD F. Possible cases illustrating how this can be achieved are given in the following example.

Example 5 *Consider the QMDD on the left-hand side of Fig. 4(a) representing a 1 to 0 and a 0 to 1 mapping of x_1. In order to establish an identity structure, this needs to be transformed into a 1 to 1 and a 0 to 0 mapping of x_1. Since this is a simple inversion of the output values, a single NOT gate, i.e. a Toffoli gate $(\emptyset, x_1)$, leads to the desired result.*

Furthermore, consider the QMDD on the left-hand side of Fig. 4(b). Here, the rightmost vertex labeled x_2 needs to be "inverted" in order to achieve a QMDD representing the identity. However, simply applying a NOT gate as before does not work, since this would also affect the leftmost vertex labeled x_2. Thus, to directly address the respective vertex, control lines are applied. More precisely, a Toffoli gate $(\{x_1^+\}, x_2)$ is added. Because of the positive control line, the inversion only applies if x_1 is assigned the value 1, i.e. only for those vertices that are connected by a p- or p'-edge to the parent vertex labeled x_1. Since the leftmost vertex is connected by an n-edge, this vertex remains unaltered.

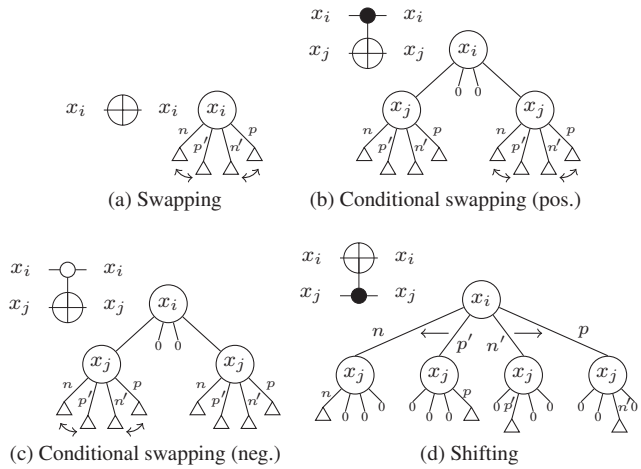
Overall, given a QMDD F, the general idea of the proposed approach is to apply Toffoli gates so that every vertex of F (as illustrated in Fig. 5(a)) is transformed into a corresponding identity structure (i.e. a vertex as illustrated in Fig. 5(b)). Then, the applied Toffoli gates can be composed into a circuit realizing F. While Example 5 only illustrated two special cases, generic transformation rules have been developed that establish the desired identity structure for any given QMDD. These rules are introduced in detail in the next section.

B. Transformation Rules

The purpose of the transformation rules is to transform any given QMDD vertex such that it represents the submatrix $\left[\begin{smallmatrix} A & 0 \\ 0 & B \end{smallmatrix}\right]$ (see Fig. 5). Therefore, succeeding paths of a vertex either need to be swapped or shifted.

(a) Before (b) After

Fig. 5. Structurally transforming a QMDD



(a) Swapping (b) Conditional swapping (pos.)

(c) Conditional swapping (neg.) (d) Shifting

Fig. 6. Reversible gates applied to QMDD vertices

In the following, we refer to these paths as follows:

Definition 3 *Let v be a vertex of a given QMDD. An* e-path *with $e \in \{n, p', n', p\}$ is a path that starts in v with an e edge and results in the $\boxed{1}$-terminal.*

In order to perform the respective swaps and shiftings, two generic rules are sufficient.

Rule 1 (Swapping) *Applying a single NOT gate on line x_i, i.e. a Toffoli gate without any control line as shown in Fig. 6(a), swaps all n- with p'-paths as well as all p- with n'-paths for each vertex labeled x_i. In order to swap only the paths of a specific vertex v, positive and negative control lines are added according to the path from the root vertex to v. More precisely, each time a p- or p'-edge is traversed a positive control line is added (see e.g. Fig. 6(b)) and each time an n- or n'-edge is traversed a negative control line is added (see e.g. Fig. 6(c)). In the remainder of the paper, we refer to a path addressing a specific vertex v as μ. This rule has already been illustrated above in Example 5.*

Using the swapping rule, *all* paths that start in a vertex v are modified. However, often only *selected* paths should be modified (e.g. in the case where all succeeding paths point to a non-terminal and, thus, swapping is not sufficient to generate the identity structure). As a result, another rule is introduced which enables the shifting of specific paths only.

Rule 2 (Shifting Rule) *Similar to the swapping rule, also the shifting rule applies positive and negative control lines in order to address a specific vertex v of the QMDD. But to additionally address specific paths to be modified, also control lines according to these paths are added. More precisely, if x_j is a label of a successive vertex of v, then adding a control line on line x_j will shift only those paths who contain an outgoing p- or p'-edge from vertices labeled x_j. Analogously, adding a*

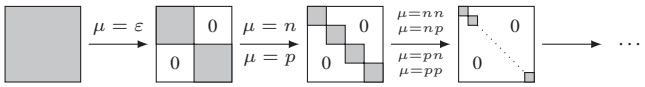


Fig. 7. Flow of the main algorithm

negative control line will consider only paths who contain an outgoing n- or n'-edge. As an example, consider Fig. 6(d). The positive control line x_j leads to a shifting of the second and the third path, since only these paths have either a p- or a p'-edge starting from the x_j-vertices. Furthermore, this principle can also be applied with more than one control line in order to address more selective paths.

Note that the applied control lines will always affect two edges, i.e. p and p' in case of a positive control line and n and n' in case of a negative control line. However, it is sufficient to consider the n- and p'-edges only. This is because after shifting e.g. all p'-paths to the n-edge, not only the p'-edge will point to the $\boxed{0}$-terminal, but also the corresponding n'-edge. This is evidenced by the following theorem.

Theorem 1 *Let M be a permutation matrix $M = \left[\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right]$ of size $2^r \times 2^r$, such that A, B, C, D are of the same size, i.e. $2^{r-1} \times 2^{r-1}$. Then B (i.e. the sub-matrix represented by a p'-edge) entirely is set to 0 if and only if C (i.e. the sub-matrix represented by an n'-edge) entirely is set to 0.*

Proof. A permutation matrix has exactly one 1-element per row and column. If $B = 0$, then for each of the first 2^{r-1} rows, there must be a 1 in A and for each of the last 2^{r-1} rows, there must be a 1 in D. Since a binary unitary matrix does not contain more than 2^r 1's, C must be 0. The opposite direction follows analogously. □

Based on the general concepts outlined above, the main flow of the proposed synthesis algorithm can be summarized. This is done in the next section.

IV. Algorithm

The formal algorithm of the proposed synthesis approach is described in the following. Afterwards, an example illustrates the application.

Algorithm Q (*QMDD-based synthesis*). This algorithm outlines the main flow of the proposed approach.

Q1. [Traverse graph.] Traverse the QMDD from the root vertex to the $\boxed{1}$-terminal in a breadth-first-manner. For each visited vertex v apply step Q2.

Q2. [Apply transformations.] Transform v, such that the identity structure as in Fig. 5 results. This can be done by processing Algorithm P on v with μ being the path from the root vertex to v. Algorithm P is described in detail at the end of this section. ∎

It is important to understand the order in which the respective vertices are considered. For this purpose, consider Fig. 7 showing boxes illustrating (sub-)matrices to be considered by the algorithm.

At the beginning, the algorithm considers the root vertex, i.e. the vertex representing the whole matrix (shadowed in Fig. 7). Applying Step Q2, the QMDD is transformed so that this vertex is structured as shown in Fig. 5(b) (i.e. the p'- and n'-edge point to the $\boxed{0}$-terminal). As a result, only two succeeding vertices (representing the top-left and the bottom-right sub-matrix) are left to be considered (see second box in Fig. 7). In order to individually address them, positive and negative control lines are applied according to the μ-path as already illustrated in Rule 1 and incorporated in the transformation rules. As an example, in the second step of Fig. 7 the same algorithm is applied to the resulting two sub-matrices once by assigning μ to n and once by assigning μ to p. This process is recursively applied until all vertices have been transformed into the identity structure.

The following definitions formalize the correlation between edges and control lines.

Definition 4 (Signature of a path) *Given a path $\pi = e_1 \dots e_r$, the* signature $s(\pi)$ *of that path is $s_1 \dots s_r$, where $s_i = +$ if $e_i \in \{p, p'\}$ and $s_i = -$ otherwise.*

Definition 5 (Path controls) *Given a path $\pi = e_1 \dots e_r$ and its signature $s(\pi) = s_1 \dots s_r$, the* path controls $c(\pi)$ *for that path is the set $\{x_1^{\varrho_1}, \dots, x_r^{\varrho_r}\}$, where $\varrho_i = s_i$.*

Overall, due to the breadth-first-traversal, Step Q2 can exploit that the considered vertex v is always reached by p and n edges (as also illustrated in Fig. 7) and, thus, the application of Toffoli gates can be controlled by the corresponding $c(\mu)$. Having that, in each vertex v of a given QMDD the following Algorithm P used in Step Q2 is applied:

Algorithm P (*Shifting paths from p' to n*). This algorithm modifies a vertex v in a QMDD F such that its p'-edge points to the $\boxed{0}$-terminal. Without loss of generality it is assumed that v is labeled x_i and that v can be reached from the root vertex using a path μ consisting of n- and p-edges only. Let $\Pi_\mu = \{\pi \mid \mu\pi \in F\}$ be all paths in F from v to the $\boxed{1}$-terminal.

P1. [Swap gate?] If $|\{\pi \mid p'\pi \in \Pi_\mu\}| > |\{\pi \mid n\pi \in \Pi_\mu\}|$, i.e. if there are more 1-paths going through the p'-edge than through the n-edge, apply a Toffoli $(c(\mu), x_i)$ to F.

P2. [Shift unique paths.] Let

$$U = \{\pi \mid p'\pi \in \Pi_\mu \wedge \nexists n\pi' \in \Pi_\mu : s(\pi) = s(\pi')\}$$

be all paths that go through p' and have a signature that is not present via a path through n. For each $\pi \in U$, apply a Toffoli gate $(c(\mu) \cup c(\pi), x_i)$ to F.

P3. [Terminate?] If p' points to a $\boxed{0}$-terminal, terminate.

P4. [Unify a path.] Apply a gate that is controlled by at least $c(\mu) \cup \{x_i^+\}$ lines and a target on a successive vertex such that a path that starts with p' can be made unique. Afterwards, return to step P2. ∎

After applying this algorithm, all p'-paths of the considered vertex v are shifted to the n-edge (i.e. p' does point to the $\boxed{0}$-terminal). Then, according to Theorem 1, also the n'-path of v is pointing to the $\boxed{0}$-terminal, i.e. the desired identity structure has been established and Algorithm Q can continue with the next vertex.

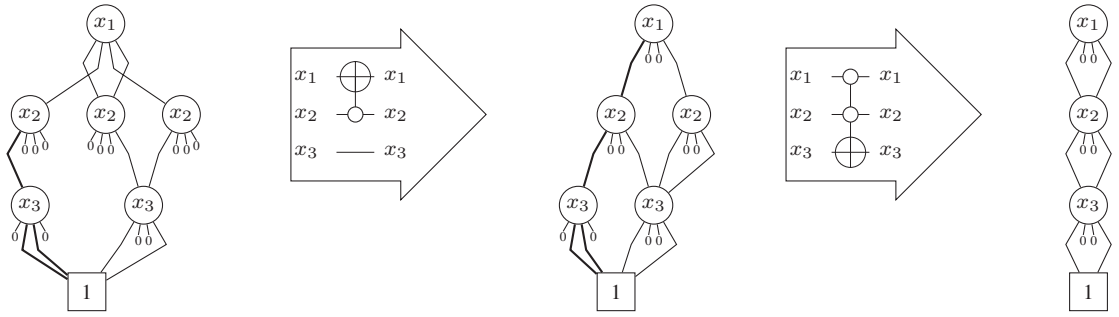


Fig. 8. Algorithm applied to the QMDD in Fig. 1(d)

Example 6 *Fig. 8 illustrates the application of the proposed algorithm by means of the QMDD from Fig. 1(d). Starting at the root vertex, first this vertex is supposed to be transformed such that both, the p'- and the n'-edge point to the $\boxed{0}$-terminal. That is, all p'-paths $\{p'np', p'nn'\}$ need to be shifted to the n-edge. To address all these paths at once it is sufficient to set a negative control on line x_2, since each n-path, i.e. $\{npn, npp\}$, has a positive outgoing edge from x_2-vertices. As a result, applying a Toffoli gate $(\{x_2^-\}, x_1)$ shifts both p'-paths at once to the n-edge resulting in the desired form for the root vertex (as depicted by means of the first arrow in Fig. 8).*

Afterwards, only the left vertex labeled with x_3 has to be swapped. This vertex is reached via the path $\mu = nn$ representing the signature $\{--\}$. Therefore, applying a Toffoli gate $(\{x_1^-, x_2^-\}, x_3)$ finally leads to the QMDD representing the identity matrix as depicted on the right hand side of Fig. 8.

V. Completeness and Correctness

In this section the completeness and the correctness of the algorithm is shown.

Theorem 2 *The algorithm described in Section IV terminates for each QMDD F that represents a permutation matrix.*

Proof. We assume that Algorithm P terminates and first prove that the main algorithm terminates under that condition. A QMDD consists of a finite number of vertices. Therefore, also the number of vertices per level is finite. It is sufficient to show, that once a vertex is brought to the form that its p' and n' edge point to the $\boxed{0}$-terminal, this will not be changed by successive steps. Algorithm P sets target lines only on the considered vertex v or on successive vertices. Since all gates are controlled at least by the signature of μ, no other vertex which is at the same level as v is transformed by this. Since additionally a breadth-first traversal is performed, already considered vertices are never modified afterwards.

It is left to show that Algorithm P terminates. The Steps 1 to 3 of Algorithm P are straightforward. Therefore, it remains to prove that a path going through p' can always be made unique. There must be a signature s that is not represented by a path going through n, since if that would be the case according to Theorem 1 p' must point already to $\boxed{0}$. Controlling the gate with x_i^+ in that step prohibits paths from being changed going through the n edge. Setting a positive control line on the respective line from vertex v can thus transform the paths going through p' independently, resulting in a path representing signature s.

Hence, the algorithm to transform F to the QMDD representing the identity matrix using Toffoli gates is complete. □

The algorithm is correct by construction since the QMDD is transformed in each step according to the applied gate. If the identity matrix results, than the sequence of gates represent a reversible circuit realizing the initial function.

VI. Experimental Evaluation

The QMDD-based synthesis method as introduced above has been implemented in C++ and evaluated on different benchmark functions. In this section, the obtained results are presented. We distinguish between two evaluations. First, the results obtained by the proposed approach are compared to previous work, namely the transformation-based synthesis method [5] and the BDD-based synthesis method [7][2]. Afterwards, results showing the scalability of the proposed approach are discussed. All experiments have been conducted on a 2.66 GHz Intel Core 2 Duo processor with 3 GB of main memory running Linux 2.6. The timeout was set to 2 000 CPU seconds.

A. Comparison to Previous Synthesis Approaches

Both the transformation-based method [5] and the BDD-based method [7] represent state-of-the-art synthesis approaches with respective pros and cons (e.g. minimal number of circuit lines but poor scalability in case of the transformation-based approach versus low quantum cost and good scalability but a large number of circuit lines in case of the BDD-based approach). The proposed QMDD-based synthesis approach provides a promising compromise between these contradictory properties.

In order to show this, all three approaches have been experimentally compared by means of a set of benchmark functions taken from RevLib [17]. The results are presented in Table I. The first column denotes the name of the respective benchmark. Afterwards, the number of lines (r), the number of gates (d), and the quantum cost (QC) of the circuits obtained by the respective synthesis approaches are reported. Column t denotes the run-time in seconds required to generate these results. The columns $\Delta\text{QC}_{\text{TBS}}$ and $\Delta\text{QC}_{\text{BDD}}$ report the absolute difference of the quantum cost measured for the circuits obtained by the transformation-based method and by

[2]In order to conduct these comparisons, we applied the implementations of these approaches as provided by RevKit [16].

TABLE I
COMPARISON TO PREVIOUS SYNTHESIS APPROACHES

	Transformation-based [5]				BDD-based [7]				QMDD-based						
Benchmark	r	d	QC	t	r	d	QC	t	r	d	QC	t	ΔQC_{TBS}	ΔQC_{BDD}	Δr_{BDD}
max46	10	284	11977	0.04	60	191	575	0.02	10	51	42248	0.03	30271	41673	-50
rd73	10	208	5889	0.03	26	85	229	0.02	10	147	13858	0.04	7969	13629	-16
sqn	10	216	7467	0.03	47	160	484	0.01	10	50	3507	0.03	**-3960**	3023	-37
sym9	10	231	4606	0.03	28	69	213	0.02	10	72	59168	0.04	54562	58955	-18
dc1	11	247	12123	0.04	28	77	193	0.03	11	25	426	0.02	**-11697**	233	-17
wim	11	753	32556	0.09	25	62	134	0.04	11	13	239	0.03	**-32317**	105	-14
z4	11	74	1069	0.04	26	75	187	0.04	11	59	3621	0.05	2552	3434	-15
cm152a	12	33	409	0.06	16	24	68	0.07	12	8	208	0.02	**-201**	140	-4
cycle10_2	12	19	1179	0.04	39	78	202	0.07	12	36	10684	0.07	9505	10482	-27
plus63mod4096	12	384	27274	0.12	23	49	89	0.07	12	26	5413	0.04	**-21861**	5324	-11
rd84	12	381	12117	0.11	37	114	314	0.08	12	278	33900	0.16	21783	33586	-25
sqrt8	12	1024	55095	0.23	31	95	259	0.08	12	55	3923	0.07	**-51172**	3664	-19
adr4	13	199	5195	0.16	33	93	237	0.19	13	75	5125	0.15	**-70**	4888	-20
dist	13	741	47281	0.32	94	331	1023	0.2	13	241	20624	0.31	**-26657**	19601	-81
plus127mod8192	13	769	58352	0.35	25	54	98	0.17	13	27	9148	0.1	**-49204**	9050	-12
plus63mod8192	13	447	38070	0.25	25	53	97	0.17	13	32	10282	0.1	**-27788**	10185	-12
radd	13	432	17074	0.24	21	55	95	0.17	13	75	5125	0.17	**-11949**	5030	-8
root	13	1000	59054	0.39	79	277	857	0.2	13	204	18497	0.27	**-40557**	17640	-66
squar5	13	83	1967	0.13	36	99	267	0.18	13	30	704	0.13	**-1263**	437	-23
clip	14	1304	111525	0.88	97	368	1196	0.45	14	232	22495	0.48	**-89030**	21299	-83
cm42a	14	2010	176497	1.21	32	79	151	0.44	14	10	260	0.24	**-176237**	109	-18
cm85a	14	625	29560	0.56	34	78	222	0.38	14	99	13368	0.24	**-16192**	13146	-20
pm1	14	2010	176497	1.2	32	79	151	0.45	14	10	260	0.24	**-176237**	109	-18
sao2	14	1176	101535	1.06	76	237	725	0.35	14	63	9018	0.27	**-92517**	8293	-62
co14	15	8270	617906	13.68	28	76	172	0.75	15	14	458710	0.24	**-159196**	458538	-13
dc2	15	803	43522	1.19	65	197	585	0.98	15	60	2974	0.56	**-40548**	2389	-50
misex1	15	1358	111286	2.4	39	103	283	0.99	15	35	1000	0.74	**-110286**	717	-24

r: Number of lines d: Number of gates QC: Quantum cost t: Run-time
ΔQC_{TBS} (ΔQC_{BDD}): Difference of the QC for circuits obtained by the QMDD-based method and the transformation-based method (BDD-based method)

the BDD-based method, respectively, compared to the circuits obtained by the proposed QMDD-based method. Finally, column Δr_{BDD} reports the differences in the number of circuit lines between the BDD-based method and their minimal value (as ensured by both, the transformation-based and the QMDD-based method).

First of all, it can be seen that run-time is not a crucial factor. If the respective data-structure (i.e. the truth-table, the BDD, or the QMDD) can be built, all synthesis approaches generate the respective circuits very fast. However, the quality of the results vary significantly.

Obviously, the BDD leads to the best results with respect to quantum cost. But this comes at a high price: a very large number of circuit lines which is way beyond the optimum. In particular in the domain of quantum computation, where circuit lines are represented by qubits, this is crucial.

In contrast, both the transformation-based synthesis approach and the QMDD-based synthesis approach lead to circuits with the minimal number of circuit lines. As a consequence, larger quantum cost have to be accepted[3]. But applying the proposed QMDD-based method, this amount can significantly be reduced. In fact, on average circuits with 50% less quantum cost are generated in comparison to the transformation-based approach. Besides that, the QMDD-based method provides much better scalability as shown in the next section.

[3]This already has been observed before in [18] for small functions. Here, significant differences in the quantum cost have been observed already between circuits with the minimal number of lines and circuits with just one or two additional lines.

B. Scalability of the QMDD-based Method

So far, no synthesis approach for large functions has been proposed which ensures the minimality of the number of circuit lines. Accordingly, no established benchmark set including large *reversible* benchmarks is available. Because of this, we show the scalability of the proposed QMDD-based method by means of structural examples (including the *toffoli* and the *graycode* function as well as arithmetic operations such as the *adder*, the *increase* module, or the *multiplier*) enriched by a set of automatically generated *random* functions. Since all these functions cannot efficiently be processed by the transformation-based method anymore (assuming a timeout of 2 000 CPU seconds), only the results obtained by the QMDD-based method are discussed in the following.

The results are presented in Table II. Again, the columns denote the number of lines (r), the number of gates (d), the quantum cost (QC) of the obtained circuits as well the run-time (t) required to generate the respective results.

As can be seen, functions including up to 100 variables can automatically be synthesized with the minimal number of circuit lines. The run-time varies depending on the respective benchmark. While for example the *increase* operation or the *graycode* function can be generated very efficiently, *random* functions or the *mutliplier* do not perform that well. However, it is left for future work to (theoretically) analyze for which functions QMDDs perform well or not. At this point, we assume that, similar to BDDs [19], different classes of functions exist that either show a positive or a negative behavior with respect to their synthesis capability.

Overall, the QMDD-based method is the first automatic synthesis approach which is applicable to larger functions and at the same time guarantees the minimal number of circuit lines.

TABLE II
SCALABILITY OF THE QMDD-BASED METHOD

Benchmark	r	d	QC	t	Benchmark	r	d	QC	t
adder9	18	3595	67768365	56.69	random1267	26	238	20122	1942.24
random1412	20	167	4219884	41.15	graycode40	40	39	39	0.00
random1752	20	15	195	29.74	increaser40	40	39	> 4294967296	0.05
random1242	20	1203	104323	48.73	toffoli40	40	1	> 4294967296	0.00
random1536	20	4	804	20.35	toffoli50	50	1	> 4294967296	0.01
multiplier5	20	11510	2241224	29.71	graycode50	50	49	49	0.00
adder10	20	8204	538649284	311.21	increaser50	50	49	> 4294967296	0.09
random1706	21	16	608	46.49	increaser60	60	59	> 4294967296	0.13
random1739	21	9408	1419008	4.75	toffoli60	60	1	> 4294967296	0.00
random1795	21	37	1975	38.53	graycode60	60	59	59	0.01
random1434	21	23	918	32.99	increaser70	70	69	> 4294967296	0.19
random1865	21	16	608	46.48	toffoli70	70	1	> 4294967296	0.00
random1383	22	2415	464628	184.31	graycode70	70	69	69	0.01
random1475	22	1902	281594	138.67	graycode80	80	79	79	0.01
random1543	22	2209	338220	206.72	toffoli80	80	1	> 4294967296	0.00
random1915	22	69	4196	80.34	increaser80	80	79	> 4294967296	0.32
random1369	22	16	800	71.41	graycode90	90	89	89	0.01
adder11	22	18445	> 4294967296	1677.82	toffoli90	90	1	> 4294967296	0.00
random1862	22	16	800	50.91	increaser90	90	89	> 4294967296	0.45
random1371	23	312880	100299312	489.52	graycode100	100	99	99	0.01
multiplier6	24	68241	17212234	794.38	increaser100	100	99	> 4294967296	0.69
random1201	25	9070	3646604	972.20	toffoli100	100	1	> 4294967296	0.01
random1937	25	19	908	900.31					

r: Number of lines
QC: Quantum cost
d: Number of gates
t: Run-time

VII. CONCLUSIONS AND FUTURE WORK

Ensuring the minimal number of circuit lines during synthesis of reversible circuits is crucial. However, existing synthesis approaches are either only applicable to small functions or lead to circuits that are way beyond the minimum of the lines. In this paper, we proposed a complementary method which overcomes these limitations. Therefore, QMDDs are utilized, that serve as an ideal data-structure to efficiently store and manipulate reversible functions.

Given a QMDD representing the function to be synthesized, Toffoli gates are applied so that every vertex of the QMDD is transformed into an identity structure. Afterwards, the applied Toffoli gates are composed to a circuit realizing the desired function. Experimental results showed that in comparison to similar approaches, circuits with 50% less quantum cost result on average. Furthermore, circuits with the minimal number of lines can automatically be realized for the first time for large functions.

In this sense, the proposed approach provides a new synthesis methodology that also builds the basis for further research. In particular, future work will focus on the theoretical analysis of the proposed approach. As shown in the experimental evaluations, some functions can efficiently be realized using the QMDD methods while other functions do not perform that well. Besides that, also the direct application of the presented ideas to quantum circuits is promising. However, although QMDDs are also capable to represent dedicated quantum operations, the corresponding transformation rules are much harder to develop.

ACKNOWLEDGMENTS

We would like to thank D. Michael Miller for providing us with an implementation of the QMDD package introduced in [10]. This work was supported by the German Research Foundation (DFG) (DR 287/20-1).

REFERENCES

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. New York, NY, USA: Cambridge University Press, Oct. 2000.

[2] R. Landauer, "Irreversibility and Heat Generation in the Computing Process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183–191, July 1961.

[3] C. H. Bennett, "Logical Reversibility of Computation," *IBM Journal of Research and Development*, vol. 17, no. 6, pp. 525–532, Nov. 1973.

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

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

[6] K. Fazel, M. Thornton, and J. Rice, "ESOP-based Toffoli Gate Cascade Generation," in *IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*. IEEE, Aug. 2007, pp. 206–209.

[7] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Design Automation Conference*. ACM, July 2009, pp. 270–275.

[8] R. Wille, O. Keszöcze, and R. Drechsler, "Determining the Minimal Number of Lines for Large Reversible Circuits," in *Design, Automation and Test in Europe*. IEEE Computer Society, Mar. 2011, pp. 1204–1207.

[9] R. Wille, M. Soeken, and R. Drechsler, "Reducing the number of lines in reversible circuits," in *Design Automation Conference*. ACM, June 2010, pp. 647–652.

[10] D. M. Miller and M. A. Thornton, "QMDD: A Decision Diagram Structure for Reversible and Quantum Circuits," in *Int'l Symp. on Multiple-Valued Logic*. IEEE Computer Society, May 2006, pp. 30–30.

[11] T. Toffoli, "Reversible computing," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, J. W. de Bakker and J. van Leeuwen, Eds., vol. 85. Springer, July 1980, pp. 632–644.

[12] E. Fredkin and T. Toffoli, "Conservative logic," *Int'l Journal of Theoretical Physics*, vol. 21, no. 3, pp. 219–253, Apr. 1982.

[13] A. Peres, "Reversible logic and quantum computers," *Phys. Rev. A*, vol. 32, no. 6, pp. 3266–3276, Dec. 1985.

[14] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, no. 5, pp. 3457–3467, Nov. 1995.

[15] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, "Quantum Circuit Simplification and Level Compaction," *IEEE Trans. on CAD*, vol. 27, no. 3, pp. 436–444, Mar. 2008.

[16] M. Soeken, S. Frehse, R. Wille, and R. Drechsler, "RevKit: A Toolkit for Reversible Circuit Design," in *Workshop on Reversible Computation*, July 2010, pp. 69–72, RevKit is available at www.revkit.org.

[17] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An Online Resource for Reversible Functions and Reversible Circuits," in *Int'l Symp. on Multiple-Valued Logic*. IEEE Computer Society, May 2008, pp. 220–225.

[18] D. M. Miller, R. Wille, and R. Drechsler, "Reducing reversible circuit cost by adding lines," in *Int'l Symp. on Multiple-Valued Logic*, May 2010, pp. 217–222.

[19] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. on Computers*, vol. C-35, no. 8, Aug. 1986.