Efficient Synthesis of Quantum Circuits Implementing Clifford Group Operations

Philipp Niemann[1] Robert Wille[1,2,3] Rolf Drechsler[1,2]
[1]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[2]Cyber Physical Systems, DFKI GmbH, 28359 Bremen, Germany
[3]Faculty of Computer Science, Technical University Dresden, 01187 Dresden, Germany
e-mail: {pniemann,rwille,drechsle}@informatik.uni-bremen.de

Abstract— **Quantum circuits established themselves as a promising emerging technology and, hence, attracted considerable attention in the domain of computer-aided design. As a result, many approaches for synthesis of corresponding netlists have been proposed in the last decade. However, as the design of quantum circuits faces serious obstacles caused by phenomena such as superposition, entanglement, and phase shifts, automatic synthesis still represents a significant challenge. In this paper, we propose an automatic synthesis approach for quantum circuits that implement Clifford Group operations. These circuits are essential for many quantum applications and cover core aspects of quantum functionality. The proposed approach exploits specific properties of the unitary transformation matrices that are associated to quantum operations. Furthermore, Quantum Multiple-Valued Decision Diagrams (QMDDs) are employed for an efficient representation of these matrices. Experimental results confirm that this enables a compact realization of the respective quantum functionality.**

I. Introduction

Quantum computation provides a new way of computation based on so called qubits [1]. In contrast to conventional bits, qubits do not only allow to represent the (Boolean) basis states 0 and 1, but also superpositions of both. By this, qubits can represent multiple states at the same time which enables massive parallelism. By additionally exploiting further quantum mechanical phenomena such as phase shifts or entanglement, quantum computation enables asymptotical speed-ups for many problems (e.g. database search or integer factorization). Driven by these prospects as well as recent accomplishments in the physical design of corresponding devices (e.g. [2]), quantum circuits established themselves as a promising emerging technology and, hence, attracted considerable attention in the domain of computer-aided design.

However, in order to formalize the above mentioned phenomena, states of qubits are modelled as vectors in high-dimensional Hilbert spaces and are manipulated by quantum operations which can be described by unitary matrices – possibly including complex numbers. This already poses serious challenges to the representation, but even more to the development of proper and efficient methods for quantum circuit synthesis. By this, we mean the task of determining a cascade of building blocks (so called quantum gates) whose sequential application to the qubits realizes a given quantum operation.

As a special case, there are many existing synthesis approaches for Boolean reversible circuits (i.e. permutation matrices), see e.g. [3], [4], [5], [6], [7], [8], [9]. In the last decade, these approaches became very efficient and allow for synthesis of rather complex (Boolean) functionality. Since many important quantum algorithms such as Grover's search algorithm [10] or Shor's factorization algorithm [11] include a substantial Boolean component, these are vital techniques for quantum circuit synthesis. Especially, since the resulting circuits can be mapped to elementary quantum gates using methods such as [12], [13], [14].

However, they do not allow for the realization of more general quantum functionality. For this matter, several approaches have been proposed, all of which rely on the fact that one-qubit gates together with the *controlled NOT* (CNOT) gate form a *universal* gate library that is sufficient to realize any given unitary matrix [12]. In fact, determining corresponding quantum circuits has a rich history (see e.g. [12], [15], [16], [17], [18], [19]). The drawback of these generic approaches is that they lead to a significant amount of gates (even for a small number of qubits) and that they rely on a set of arbitrary one-qubit gates. The latter poses a severe obstacle since, in physical realizations, these must be approximated by a restricted set of gates, in particular when fault tolerant methods are applied [1].

In this work, we provide an alternative synthesis approach that considers synthesis of quantum circuits implementing Clifford Group operations. By this, we avoid relying on a generic gate library and, instead, can realize quantum functionality with a precise and established set of gates including Hadamard, Phase, and CNOT gates which can be implemented in a fault tolerant way [20]. At the same time, this restricts the applicability of our approach (arbitrary unitary matrices are not supported). However, Clifford group circuits are essential for many quantum applications and cover core aspects of quantum functionality such as superposition, entanglement, and phase shifts [21]. Moreover, their functionality is sufficient for various quantum applications, particularly as stabilizer circuits for error-correcting codes [22], but also for the realization of the Greenberger-Horne-Zeilinger experiment [23], for quantum teleportation [24], or dense quantum coding [25].

This compromise on applicability enables a synthesis methodology allowing for the realization of considerably more compact quantum circuits. We explicitly exploit the effects of the clearly defined gate library in order to directly modify the given unitary matrices to be synthesized. In contrast to the previously proposed (generic) approaches and as confirmed by an experimental evaluation, this enables a reduction of the circuit sizes by several orders of magnitude. By additionally using a compact data-structure, namely *Quantum Multiple-valued Decision Diagrams* (QMDDs) [26], our approach enables an efficient processing of the respective unitary matrices.

The remainder of the paper is structured as follows. In Section II, the background of quantum computation, quantum circuits, as well as QMDDs is briefly reviewed. Section III introduces the main concepts of the proposed synthesis scheme, while the resulting synthesis algorithm is described in detail in Section IV. An experimental evaluation is presented in Section V and, finally, Section VI concludes the paper.

II. Background

This section briefly reviews the basics on quantum computation and quantum circuits. Furthermore, we introduce the basic ideas of *Quantum Multiple-valued Decision Diagrams (QMDDs)*, a data-structure used to efficiently represent quantum functionality. For more detailed introduction, we refer to [1], [26].

A. Quantum Computation and Circuits

Quantum systems are composed of *qubits*. Analogously to conventional bits, a qubit can be in one of the computational *basis states* $|0\rangle$ and $|1\rangle$, but also in a so called *superposition* $\alpha|0\rangle + \beta|1\rangle$ for complex-valued α, β with $|\alpha|^2 + |\beta|^2 = 1$. An n-qubit quantum system can be in one of the 2^n basis states ($|0\dots00\rangle, |0\dots01\rangle, \dots, |1\dots11\rangle$) or a superposition of these states. Accordingly, the state of a quantum system is represented by a *state vector* (of dimension 2^n).

By the postulates of quantum mechanics, the evolution of a quantum system due to a quantum operation can be described by a *unitary transformation matrix*, i.e. an invertible complex-valued matrix whose inverse is given by the adjoint matrix.

Example 1. *Commonly used quantum operations include the* Hadamard *operation H (setting a qubit into a superposition), the* phase shift *operations S (Phase gate) and $Z = S^2$, as well as the* NOT *operation X which flips the basis states $|0\rangle$ and $|1\rangle$. The corresponding unitary matrices are defined as*

$$H = \tfrac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix},\ S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix},\ Z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix},\ X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}.$$

Besides these operations that work on a single target qubit*, there are also controlled operations on multiple qubits. The state of the additional* control qubits *determines which operation is performed on the target qubit. An example is the* controlled NOT *(CNOT) operation on two qubits which is defined by*

$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$$

and applies the NOT operation to the target if the control is in the $|1\rangle$-state.

Realizations of quantum operations are represented by *quantum gates* g_i which eventually form a *quantum circuit* $G = g_1 \dots g_d$ with $1 \le i \le d$. For the purpose of visualization each qubit is represented by a solid line and gates are arranged in a cascade from left to right indicating the application of the respective unitary matrices to the qubits over time.

Example 2. *Fig. 1 shows the corresponding gate representations of the quantum operations specified by the matrices from Example 1. Note that for the CNOT gate a black circle represents the control line connection, while a "$\bigoplus$" represents the target line connection.*

In this work, we consider the synthesis of quantum circuits implementing Clifford Group operations. It has been shown in [27] that each Clifford Group operation can be realized by a cascade composed of Hadamard, Phase, and CNOT gates. Conversely, any cascade of these gates forms a Clifford Group operation. Therefore, these gates are also called *generators* of the Clifford Group.

B. Quantum Multiple-valued Decision Diagrams

QMDDs [26] have been introduced as a means for the efficient representation and manipulation of quantum gates and circuits. The fundamental idea is a recursive partitioning of the respective transformation matrix and the use of edge and vertex weights to represent various complex-valued matrix entries. More precisely, a transformation matrix of dimension $2^n \times 2^n$ is successively partitioned into four sub-matrices of dimension $2^{n-1} \times 2^{n-1}$. This partitioning is represented by a directed acyclic graph – the QMDD. The following example illustrates main aspects.

Example 3. *Fig. 2a shows a transformation matrix for which a QMDD as shown in Fig. 2b has been built. Starting with a single* terminal vertex $\boxed{1}$ *that represents the lowest partitioning level, i.e. single matrix entries, the next upper level of 2×2 matrices is represented by vertices labeled x_2. For each entry, there is an outgoing edge to the terminal vertex with an* edge weight *corresponding to the respective complex value. For simplicity, we omit edge weights equal to 1 and indicate edges with a weight of 0 by stubs. The vertices are* normalized *by dividing the weights of all outgoing edges by a normalization factor (here: such that the "leftmost" edge with a non-zero weight has weight 1). This factor is propagated to referencing edges, e.g. the factor $\frac{1}{2}$ is propagated upwards from the x_2-level to the x_0-level in Fig. 2b. By this, structurally equivalent sub-matrices are compressed to a shared vertex (highlighted in grey in Fig. 2a and 2b, respectively). This procedure is repeated for each level until a single vertex labeled by x_0 is created for the top level. This vertex is called the* root vertex*. Finally, a possible normalization factor of this vertex is assigned to the weight of the* root edge *which points to the root vertex, but has no source.*

To obtain the value of a particular matrix entry, one has to follow the corresponding path from the root to the terminal vertex and multiply all edge weights on this path. For example, the matrix entry $\frac{i}{2}$ from the top right sub-matrix of Fig. 2a (highlighted bold) can be determined as the product of the weights on the highlighted path of the QMDD in Fig. 2b.

It can be shown that normalization as described above enables canonical QMDD representations [26].

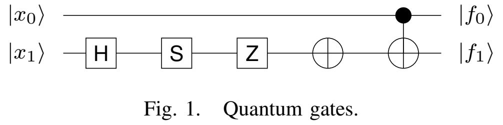


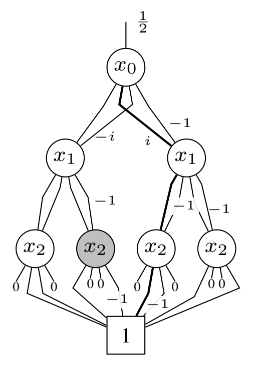
Fig. 1. Quantum gates.

Inputs

$$\begin{array}{c|cccc:cccc}
 & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111\\
\hline
000 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \mathbf{\frac{i}{2}} & 0 & \frac{-i}{2}\\
001 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{-i}{2} & 0 & \frac{i}{2} & 0\\
010 & \frac{1}{2} & 0 & \frac{-1}{2} & 0 & \frac{-i}{2} & 0 & \frac{-i}{2} & 0\\
011 & 0 & \frac{-1}{2} & 0 & \frac{1}{2} & 0 & \frac{-i}{2} & 0 & \frac{-i}{2}\\
\hdashline
100 & 0 & \frac{-i}{2} & 0 & \frac{-i}{2} & 0 & \frac{-1}{2} & 0 & \frac{1}{2}\\
101 & \frac{-i}{2} & 0 & \frac{-i}{2} & 0 & \frac{1}{2} & 0 & \frac{-1}{2} & 0\\
110 & \frac{-i}{2} & 0 & \frac{i}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0\\
111 & 0 & \frac{i}{2} & 0 & \frac{-i}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2}
\end{array}$$

Outputs

(a)



(b)

Fig. 2. Matrix and QMDD representation of a 3-qubit quantum circuit.

III. Main Concepts of the Synthesis Approach

In this section, we introduce the general idea of the proposed synthesis approach. Furthermore, we describe the effect of the Clifford Group generators (Hadamard, Phase, CNOT) to a given transformation matrix. Based on that, the actual synthesis algorithm is afterwards described in Section IV.

A. General Idea

The task of synthesis is to determine a quantum circuit representing the desired quantum functionality F given in terms of a transformation matrix. We already know that all circuits considered in this work can be realized by a cascade composed of Hadamard, Phase, and CNOT gates. Therefore, applying transformation matrices representing these gates modifies F and for a distinct choice of these matrices we will eventually reach the identity matrix. Hence, the main goal of the proposed synthesis approach is to determine a sequence of quantum gates $g_1 \dots g_d$ with this property. For this purpose, we identify the following three steps that are also illustrated in Fig. 3.

1) *Eliminate superposition*, i.e. apply quantum gates so that all multiple non-zero matrix entries in rows/columns are removed. This leads to a matrix that has a single non-zero entry per row/column (as illustrated in Fig. 3b), each of which has magnitude 1. Thus, the matrix is structurally equivalent to a permutation matrix. The corresponding circuit maps basis states to (possibly different) basis states and potentially applies phase shifts.
2) *Diagonalize*, i.e. establish the structure of a diagonal matrix as shown in Fig. 3c. By this, the structure of the transformation matrix is already equal to the structure of the identity matrix, i.e. each basis state is mapped onto itself. However, phase shifts still might be applied.
3) *Eliminate phase shifts* to eventually reach the identity matrix as shown in Fig. 3d.

All gates applied in order to perform these steps lead to a circuit realizing the inverse of the given transformaion matrix F. Since the Hadamard and CNOT gates are their own inverses and the inverse of the Phase gate is $S^3 = S \cdot Z$, the actually desired circuit can then be derived by simply inverting the order of all gates and replacing Phase gates by their inverses. Alternatively, we can convert F to its inverse in advance which is a simple operation using QMDDs.

B. Effect of the Clifford Group Generators

All synthesis steps sketched above can be accomplished by a clever sequential application of Hadamard, Phase, and CNOT gates, the generators of the Clifford Group. But before the respective algorithm is described in detail, we briefly investigate the effect of these gates to a transformation matrix F.

For simplicity, we illustrate all operations on a 4×4 transformation matrix F over two qubits x_0 and x_1. The generalization to larger matrices with more qubits is straightforward. In the following, we write U_{target} for the transformation matrix that represents the (uncontrolled) operation of a U-gate applied to a target qubit. Similarly, we use $U_{\text{target}}^{\text{control}}$ for a controlled U-gate, i.e. $X_{x_1}^{x_0}$ denotes the CNOT gate with control qubit x_0 and target x_1.

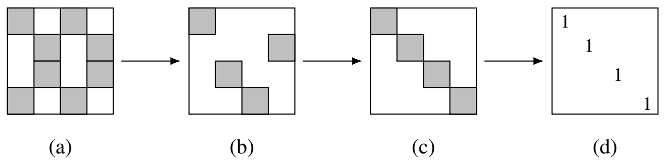


Fig. 3. General scheme of the synthesis procedure.

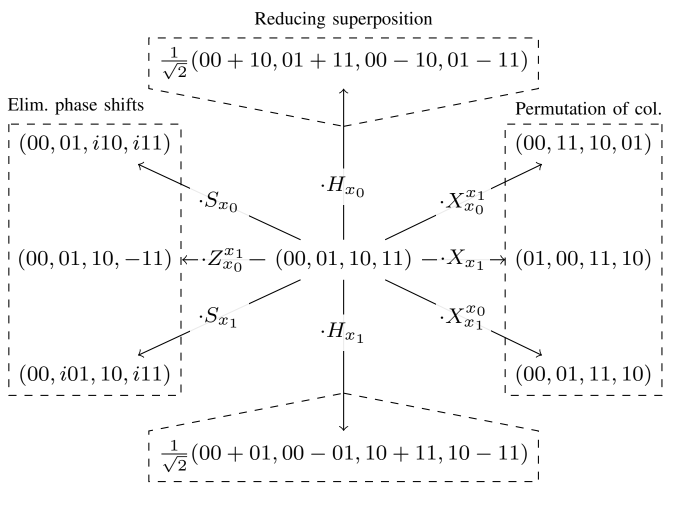


Fig. 4. Operation scheme of gate matrices.

Fig. 4 summarizes how the application of important Clifford operations affects a given transformation matrix F with columns $00, 01, 10$, and 11, i.e. $F = (00, 01, 10, 11)$. For our purposes, three main effects are important:

1) Permutation of columns: The CNOT gates $X_{x_1}^{x_0}$ and $X_{x_0}^{x_1}$ lead to matrices with columns 10 and 11 or 01 and 11 being interchanged, respectively. To additionally interchange column 00 with another one, an (uncontrolled) NOT gate (X) can be applied. This gate can be generated with Hadamard and Phase gates, more precisely $X = H \cdot S \cdot S \cdot H$. Hence, by composing gates $X_{x_1}^{x_0}$, $X_{x_0}^{x_1}$, and X_{x_1}, the columns of F can be re-arranged until any desired permutation is achieved. Note that for larger matrices we are no longer able to achieve any desired permutation of the columns, but we can still move a (single) column to any desired place.

2) Reducing superposition: Hadamard gates H_{x_0} and H_{x_1} link together pairs of columns as illustrated in Fig. 4, thereby allowing to create or reduce superposition. Hence, this operation is important for the first step of the synthesis approach sketched above. However, reduction of superposition is only possible for suitable pairs of columns (i.e. for 00 and 10 or 01 and 01) and only if the columns differ by no more than signs of the entries. If other pairs need to be linked together (e.g. 00 and 11), the corresponding columns have to be re-arranged before. For this purpose, permutation of columns as described above can be applied.

3) Eliminating phase shifts: Finally, Fig. 4 shows that Phase gates S_{x_0} and S_{x_1} apply a phase shift of i to a particular subset of columns of F. Beyond that, they do not alter the order of the columns. Applied again, they change this phase shift to -1 and $-i$ successively and, finally, remove the phase shift completely. Obviously, this is relevant for the third step of the synthesis approach sketched above. As Phase gates also only work on pairs of columns, we additionally might need so called controlled Z gates defined by

$$Z_{x_1}^{x_0} = Z_{x_0}^{x_1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

These apply a phase shift by -1 to a more restricted set of columns than Phase gates (here: a single column). This gate type can readily be constructed using Hadamard and CNOT gates, more precisely $Z_{x_1}^{x_0} = H_{x_0} \cdot X_{x_0}^{x_1} \cdot H_{x_0}$.

Exploiting these effects, the sketched synthesis approach can be realized as described next.

IV. Algorithm

Based on the main concepts introduced in the previous section, we now describe the resulting synthesis approach in detail. The approach follows the three steps from Fig. 3. Furthermore, we employ QMDDs as reviewed in Section II-B as an efficient representation of transformation matrices on which all steps are conducted[1]. All steps are illustrated by a running example, namely the transformation matrix depicted in Fig. 2a and the corresponding QMDD depicted in Fig. 2b. We will exploit the specific property of Clifford group transformation matrices that all non-zero entries are multiples of a *basis weight*. More precisely, there is a complex number ω such that each non-zero entry is of the form $u \cdot \omega$ for $u \in \{\pm 1, \pm i\}$. This property follows from the theory of stabilizer circuits as discussed in [28].

A. Eliminating Superposition

As discussed in the previous section, superposition can be eliminated using Hadamard gates. More precisely, superposition involving a suitable pair of columns with no phase shifts can straightforwardly be tackled using a single Hadamard gate applied to the corresponding qubit.

Example 4. *In the running example from Fig. 2a, it can be seen that column* 001 *matches pairwise to column* 011 *(i.e. the entries differ only by sign). Hence, applying a Hadamard gate on qubit x_1 reduces the superposition of the matrix and leads to a matrix as shown in Fig. 5a.*

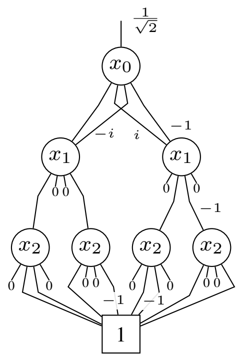
However, cases might be encountered where no Hadamard gates can be applied directly. Then, the columns have to be re-arranged using CNOT gates and possible phase shifts have to be removed using Phase gates. This is illustrated by the following example before the actually applied elimination scheme is described next.

Example 5. *The resulting matrix from Fig. 5a still includes superposition. In order to eliminate this, we need to derive a "valid" pair of columns for which a Hadamard gate can be applied. We choose columns 001 and 111 and first use a CNOT gate $X_{x_1}^{x_0}$ to move column 111 to 101. Then, we perform a Phase gate on qubit x_0 to remove the phase shift to column 001. Finally, the application of a Hadamard gate at qubit x_0 eventually eliminates all superposition in this matrix. The resulting matrix is shown in Fig. 6a.*

[1]Note that the proposed approach can as well be applied to alternative representations of unitary matrices.

$$\begin{array}{c|cccc:cccc}
{}^{x_0 x_1 x_2} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
\hline
000 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & \frac{i}{\sqrt{2}} \\
001 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & \frac{-i}{\sqrt{2}} & 0 \\
010 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & \frac{-i}{\sqrt{2}} & 0 & 0 & 0 \\
011 & 0 & 0 & 0 & \frac{-1}{\sqrt{2}} & 0 & \frac{-i}{\sqrt{2}} & 0 & 0 \\
\hdashline
100 & 0 & \frac{-i}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\sqrt{2}} \\
101 & \frac{-i}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\
110 & 0 & 0 & \frac{-i}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\
111 & 0 & 0 & 0 & \frac{i}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 & 0
\end{array}$$

(a)

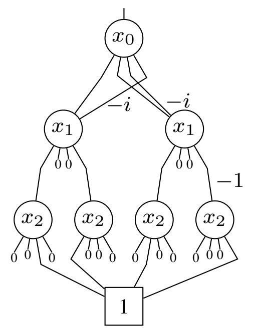


(b)

Fig. 5. Running example after applying H_{x_1}.

$$\begin{array}{c|cccc:cccc}
{}^{x_0 x_1 x_2} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
\hline
000 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
001 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
010 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
011 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
\hdashline
100 & 0 & -i & 0 & 0 & 0 & 0 & 0 & 0 \\
101 & 0 & 0 & 0 & 0 & -i & 0 & 0 & 0 \\
110 & 0 & 0 & 0 & 0 & 0 & 0 & -i & 0 \\
111 & 0 & 0 & 0 & i & 0 & 0 & 0 & 0
\end{array}$$

(a)



(b)

Fig. 6. Running example after eliminating superposition.

Overall, superposition can gradually be eliminated by repeating the following steps.

1) Determine the first two non-zero entries in the first row and store their quotient q.
2) Rearrange the columns in such a way that their column index only differs in one place x_d. This can be done by choosing a place x_d where the indices initially differ and applying controlled NOT gates (controlled by x_d) on any other place where the indices differ.
3) If $q = \pm i$, perform a Phase gate on x_d.
4) Afterwards, perform a Hadamard gate on x_d.

Using QMDDs, these steps can be conducted efficiently. Since the basis weight ω of a matrix can be obtained through the weight of the root edge of its QMDD, it is easy to check whether there is superposition in the matrix or not. The resulting QMDD is shown in Fig. 6b.

B. Diagonalization

Once superposition has been removed from the matrix, there is a single non-zero entry per row and column (as in Fig. 6a). By unitarity, all of these entries and, hence, also the basis weight must have magnitude 1, i.e. the matrix is structurally equivalent to a permutation matrix and the corresponding circuit maps basis states to other basis states – possibly with phase shifts. Hence, in order to derive the structure of a diagonal matrix, the respective columns have to be permuted only. This can be done by applying suitable NOT and CNOT gates. In order to determine which gates to apply, we compute *line functions* f_{x_i} for each qubit x_i. These denote the (logic) formula that expresses for which inputs (columns) we get an output $|1\rangle$ on the respective qubit.

Example 6. *We read from the current matrix in Fig. 6a that*

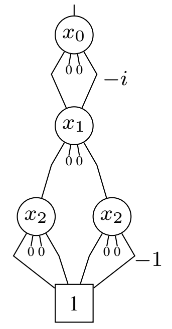
$$\begin{aligned}
f_{x_0} &= 001 \vee 011 \vee 100 \vee 110 &&= x_0 \oplus x_2 \\
f_{x_1} &= 010 \vee 011 \vee 110 \vee 111 &&= x_1 \\
f_{x_2} &= 000 \vee 011 \vee 100 \vee 111 &&= \overline{x_1 \oplus x_2}
\end{aligned}$$

In order to achieve a diagonal structure, we need $f_{x_i} = x_i$. To establish this for x_0, we need to XOR x_2 by applying $X_{x_0}^{x_2}$. For x_2, this can similarly be accomplished by applying $X_{x_2}^{x_1}$ and X_{x_2}. This leads to a matrix as shown in Fig. 7a.

It can be shown, that the line functions are always such XOR products of the qubits. However it may happen that x_i does not appear in f_{x_i} (but $x_j \neq x_i$ does). In this case, we firstly need to swap the qubits by applying $X_{x_j}^{x_i}$, $X_{x_i}^{x_j}$, and $X_{x_j}^{x_i}$.

$$\begin{array}{c|cccc|cccc} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ 000 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 001 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 010 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 011 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ \hline 100 & 0 & 0 & 0 & 0 & -i & 0 & 0 & 0 \\ 101 & 0 & 0 & 0 & 0 & 0 & -i & 0 & 0 \\ 110 & 0 & 0 & 0 & 0 & 0 & 0 & -i & 0 \\ 111 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & i \end{array}$$

(a)



(b)

Fig. 7. Running example after diagonalization.

Thus, the respective gates to be applied are derived by computing the line function for each qubit and, based on it, successively applying NOT/CNOT gates until the desired diagonal structure results. Using QMDDs, the line functions can easily be computed in a compressed form (as a Binary Decision Diagram) if the respective qubit is on the top level of the diagram.

More precisely, the following algorithm is applied (all variables are initially marked unvisited):

1) Pick an unvisited variable x and move it to the top of the QMDD by variable interchange.
2) Compute the output function f_x of x.
3) If f_x does not depend on x, i.e. if the top vertex of its compressed representation is not labeled by x (but by y), perform the CNOT gates X_y^x, X_x^y, and X_y^x.
4) For each variable $y \neq x$ on which f_x depends, perform a CNOT gate with target x and controlled by y, i.e. X_x^y.
5) Mark x as visited.

After all variables have been visited, each basis state is mapped onto itself and, thus, a diagonal matrix structure has been achieved. For the running example, this leads to the QMDD as shown in Fig. 7b.

C. Eliminating Phase Shifts

Finally, possible phase shifts are eliminated, i.e. all diagonal entries of the matrix are equalized. This is conducted by Phase gates (performing shifts by $\pm i$) and controlled Z gates (performing partial shifts by -1).

Example 7. *The phase shifts in the current matrix shown in Fig. 7a can be eliminated by applying a Phase gate at qubit x_0 (S_{x_0}). This transforms the values $\pm i$ to ± 1. To remove the phase shift -1 of $|011\rangle$, a controlled Z gate with target x_2 and controlled by x_1 ($Z_{x_2}^{x_1}$) is applied. This eventually leads to the identity matrix and, hence, terminates the synthesis.*

Note that we do not require the first entry of the diagonal (first row, first column) to have value $+1$. Any complex value of magnitude 1 is accepted and all other diagonal entries are transformed to the same value. This leads to a matrix that might not be the identity matrix itself, but that is equivalent up to *global phase* and, hence, physically indistinguishable [1].

Since phase shifts are indicated by edge weights $\pm i$ and -1 in the QMDD, they can easily be eliminated by applying the corresponding gate to qubits as illustrated in Fig. 8. In order to address edges on lower levels, re-ordering of the QMDD structure is applied. Eventually, all phase shifts can be eliminated by "moving" the respective qubit variable to the root level and applying the gates as shown in Fig. 8.

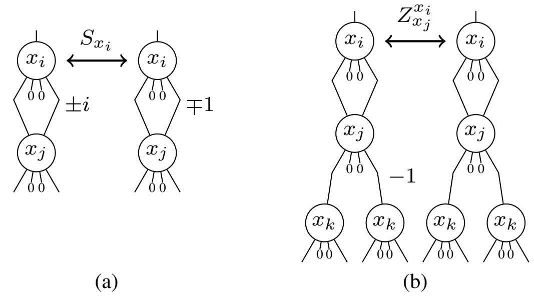


Fig. 8. Removing non-trivial edge weights from a QMDD.

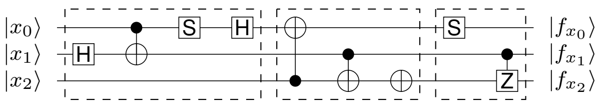


Fig. 9. Resulting quantum circuit.

Overall, performing the respective steps as described above eventually leads to the quantum gate cascade depicted in Fig. 9. As explained in Section III-A, this realizes the inverse of the given unitary matrix of our running example from Fig. 2.

A detailed proof for the algorithm's convergence is available from the authors, but will not be given here due to page limitations.

V. Experimental Results

The synthesis approach discussed above has been implemented in C on top of the original QMDD package presented in [26]. In this section, we evaluate the results obtained by the approach and compare them to synthesis schemes previously proposed in [18], [19]. To this end, arbitrary transformation matrices with up to 20 qubits (denoted *arbitrary*) as well as quantum functionality taken from [22] and realizing Shor's 9-qubit error correcting code (denoted by *9qubitN1* and *9qubitN2*), a 7-qubit encoding (denoted by *7qubitcode*), as well as an error syndrome measurement circuit for a 5-qubit code (denoted by *5qubitcode*) have been used.

The results are summarized in Table I. The first column provides the identifiers of the respective benchmarks followed by its number of qubits. In the remaining columns, the costs, i.e. the number of gates, of the resulting circuits are provided. It is a common understanding that (physical) implementations of CNOT gates are more error-prone and have greater delays than one-qubit gates. Therefore and in accordance with the evaluation of [18], [19], we distinguish between the number of one-qubit gates and the number of CNOTs in Table I. Furthermore, the best available results from [18], [19] are provided for comparison. Finally, the run-time of the proposed synthesis approach is provided in the last column. All experiments have been conducted on a 2.8 GHz Intel Core i7 machine with 8 GB of main memory running Linux.

First of all, we emphasize again that the approaches proposed in [18], [19] rely on a gate library composed of an arbitrary set of one-qubit gates together with a CNOT. This poses a severe obstacle since physical realizations and particularly fault tolerant methods impose limitations on the set of gates that may be used. In contrast, the approach presented here can realize the considered quantum functionality with a precise and established set of gates including only Hadamard, Phase, and CNOT gates[2].

[2]Note again that the controlled Z gates applied e.g. for eliminating phase shifts are eventually realized by a Hadamard–CNOT–Hadamard cascade.

TABLE I
EXPERIMENTAL EVALUATION

		Prev. Approach [18]	Prev. Approach [19]		Proposed Approach		
Benchmark	#Qubits	#CNOTs	#CNOTS	#one-qubit gates	#CNOTS	#one-qubit gates	Time (s)
Arbitrary transformation matrices							
arbitrary4	4	100	112	138	15	33	<0.01
arbitrary5	5	444	480	537	26	43	<0.01
arbitrary6	6	1 868	1 976	2 209	36	46	<0.01
arbitrary7	7	7 660	8 040	8 528	45	55	0.02
arbitrary8	8	31 020	32 456	33 455	61	72	<0.01
arbitrary9	9	124 844	130 408	134 415	68	61	0.03
arbitrary10	10	500 908	522 920	531 022	87	89	0.05
arbitrary11	11	2 006 700	2 094 376	2 110 669	102	98	0.10
arbitrary12	12	8 032 940	8 382 888	8 448 077	144	135	0.28
arbitrary15	15	$\approx 5.14 \cdot 10^8$	–	–	203	173	2.37
arbitrary20	20	$\approx 5.27 \cdot 10^{11}$	–	–	217	222	26.40
Quantum functionality taken from [22]							
5qubitcode	9	124 844	–	–	28	28	<0.01
7qubitcode	7	7 660	–	–	11	19	<0.01
9qubitN1	9	124 844	–	–	8	3	<0.01
9qubitN2	17	$\approx 8.23 \cdot 10^9$	–	–	34	4	<0.01

Benchmark: Name of benchmark – #Qubits: Number of qubits – #CNOTs: Number of CNOT gates – #one-qubit gates: Number of one-qubit gates

Besides that, Table I clearly shows that, using the proposed method, much more efficient quantum circuits can be realized for Clifford Group functionality compared to the more generic approaches presented before. In fact, reductions of several orders of magnitudes of CNOT gates can easily be obtained.

VI. Conclusions

In this work, we proposed a synthesis approach for quantum circuits implementing Clifford Group operations. Clifford group circuits are essential for many quantum applications such as stabilizer circuits, quantum teleportation, and more. In contrast to previous approaches for synthesis, we avoid relying on a generic gate library and, instead, exploit the specific effects of the Clifford Group generators to the unitary matrix to be synthesized. Experiments confirmed that, compared to the generic approaches presented before, quantum circuits with several orders of magnitude less gates can be realized using the proposed approach. Future work focuses on extending the proposed method to address more general quantum functionality.

References

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

[2] R. V. Meter and M. Oskin, "Architectural implications of quantum computing technologies," *J. Emerg. Technol. Comput. Syst.*, vol. 2, no. 1, pp. 31–63, 2006.

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

[4] G. Yang, X. Song, W. N. N. Hung, and M. A. Perkowski, "Fast synthesis of exact minimal reversible circuits using group theory," in *ASP Design Automation Conf.*, 2005, pp. 1002–1005.

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

[6] M. Saeedi, M. S. Zamani, M. Sedighi, and Z. Sasanian, "Synthesis of reversible circuit using cycle-based approach," *J. Emerg. Technol. Comput. Syst.*, vol. 6, no. 4, 2010.

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

[8] R. Wille, S. Offermann, and R. Drechsler, "SyReC: A programming language for synthesis of reversible circuits," in *Forum on Specification and Design Languages*, 2010, pp. 184–189.

[9] M. Soeken, R. Wille, C. Hilken, N. Przigoda, and R. Drechsler, "Synthesis of reversible circuits with minimal lines for large functions," in *ASP Design Automation Conf.*, 2012, pp. 85–92.

[10] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Theory of computing*, 1996, pp. 212–219.

[11] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Foundations of Computer Science*, pp. 124–134, 1994.

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

[13] D. M. Miller, R. Wille, and Z. Sasanian, "Elementary quantum gate realizations for multiple-control Toffolli gates," in *Int'l Symp. on Multi-Valued Logic*, 2011, pp. 288–293.

[14] R. Wille, M. Soeken, C. Otterstedt, and R. Drechsler, "Improving the mapping of reversible circuits to quantum circuits using multiple target lines," in *ASP Design Automation Conf.*, 2013, pp. 85–92.

[15] J. J. Vartiainen, M. Mottonen, and M. M. Salomaa, "Efficient decomposition of quantum gates," *Phys. Rev. Lett.*, vol. 92, no. 177902, 2004.

[16] V. Bergholm, J. J. Vartiainen, M. Mottonen, and M. M. Salomaa, "Quantum circuits with uniformly controlled one-qubit gates," *Phys. Rev. A*, vol. 71, no. 052330, 2005.

[17] Y. Nakajima, Y. Kawano, and H. Sekigawa, "A new algorithm for producing quantum circuits using KAK decompositions," *Quantum Information & Computation*, vol. 6, no. 1, pp. 67–80, 2006.

[18] V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum-logic circuits," *IEEE Trans. on CAD*, vol. 25, no. 6, pp. 1000–1010, 2006.

[19] M. Saeedi, M. Arabzadeh, M. S. Zamani, and M. Sedighi, "Block-based quantum-logic synthesis," *Quantum Information & Computation*, vol. 11, no. 3&4, pp. 262–277, 2011.

[20] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, "A new universal and fault-tolerant quantum basis," *Information Processing Letters*, vol. 75, no. 3, pp. 101–107, 2000.

[21] V. Kliuchnikov, D. Maslov, and M. Mosca, "Asymptotically optimal approximation of single qubit unitaries by Clifford and T circuits using a constant number of ancillary qubits," *CoRR*, vol. abs/1212.0822, 2012.

[22] N. D. Mermin, *Quantum Computer Science: An Introduction*. Cambridge University Press, 2007.

[23] D. M. Greenberger, M. A. Horne, and A. Zeilinger, *Bells Theorem, Quantum Theory, and Conceptions of the Universe*. Kluwer Academic Press, 1989.

[24] C. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres, and W. Wootters, "Teleporting an unknown quantum state by dual classical and EPR channels," *Phys. Rev. Lett.*, vol. 70, no. 1895-1898, 1993.

[25] C. Bennett and S. J. Wiesner, "Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states," *Phys. Rev. Lett.*, vol. 69, no. 2881, 1992.

[26] D. M. Miller and M. A. Thornton, "QMDD: A decision diagram structure for reversible and quantum circuits," in *Int'l Symp. on Multi-Valued Logic*, 2006, p. 6.

[27] D. Gottesman, "Stabilizer codes and quantum error correction," *arXiv preprint quant-ph/9705052*, 1997.

[28] ——, "The Heisenberg representation of quantum computers," *arXiv preprint quant-ph/9807006*, 1998.