Quantum Information Processing manuscript No.
(will be inserted by the editor)

Advanced Exact Synthesis of Clifford+T Circuits

Philipp Niemann · Robert Wille · Rolf Drechsler

the date of receipt and acceptance should be inserted later

Abstract Quantum systems provide a new way of conducting computations based on so-called qubits. Due to the potential for significant speed-ups, this field received significant research attention in recent years. The Clifford+T library is a very promising and popular gate library for these kinds of computations. Unlike other libraries considered so far, it consists of only a small number of gates for all of which robust, fault-tolerant realizations are known for many technologies that seem to be promising for large-scale quantum computing. As a consequence, (logic) synthesis of Clifford+T quantum circuits became an important research problem. However, previous work in this area has several drawbacks: corresponding approaches are either only applicable to very small quantum systems or lead to circuits that are far from being optimal. The latter is mainly caused by the fact that current synthesis realizes the desired circuit by a local, i.e. column-wise, consideration of the underlying unitary transformation matrix to be synthesized. In this paper, we analyze the conceptual drawbacks of this approach and propose to overcome them by taking a global view of the matrices and perform a separation of concerns regarding individual synthesis steps. We precisely describe a corresponding algorithm and as well as its efficient implementation on top of decision diagrams.

P. Niemann
University of Bremen, D-28359 Bremen, Germany
DFKI GmbH, D-28359 Bremen, Germany
E-mail: philipp.niemann@dfki.de

Robert Wille
Institute for Integrated Circuits, Johannes Kepler University, A-4400 Linz, Austria
DFKI GmbH, D-28359 Bremen, Germany
Software Competence Center Hagenberg GmbH (SCCH), Hagenberg, Austria

Rolf Drechsler
University of Bremen, D-28359 Bremen,Germany
DFKI GmbH, D-28359 Bremen, Germany

Experimental results confirm the resulting benefits and show improvements of up to several orders of magnitudes in costs compared to previous work.

Keywords quantum logic synthesis · quantum computing · quantum decision diagrams

1 Introduction

Quantum computation is an emerging technology where operations are performed on quantum bits (*qubits*) rather than conventional bits. In contrast to conventional bits, qubits are not limited to a discrete set of states (Boolean 0 and 1), but—exploiting quantum-physical effects—also allow for arbitrary superpositions of these Boolean basis states. This can be utilized to gain significant speed-ups for several interesting and practically relevant problems. Prominent examples include factorization, database search, or the simulation of chemical dynamics, for which corresponding quantum algorithms have been proposed e.g. in [36], [15], and [19], respectively.

To this end, complex quantum computations are usually described in terms of a cascade of simple quantum operations (*quantum gates*) which eventually form a *quantum circuit* [26]. In contrast to conventional circuits, quantum circuits are not supposed to describe a network of wires and physical gates to be connected, but basically define in which order the quantum gates/operations are applied to the qubits.

While in theory there exists an infinite number of quantum gates (even for the case of one-qubit gates), practical quantum circuits need to be composed of a subset of elementary gates which can be physically realized in a fault-tolerant fashion in the considered quantum technology. The latter is required in order to increase the robustness of the operations, as all technologies considered to date are very sensitive to environmental perturbations and, thus, require sophisticated mechanisms for error correction. This obviously motivates to study how desired quantum functionality is realized in terms of elementary quantum gates—a problem known as quantum (logic) synthesis [17, 18, 23, 32, 34].

In this context, the Clifford+T gate library [5] received significant attention in the recent past by providing a set of elementary gates that is universal (i.e. any quantum functionality can be realized with it up to an arbitrary small error ϵ) and consists only of a small number of gates—all of which are very well compatible to many established error correction schemes and can be physically implemented in all quantum technologies that seem promising for large-scale quantum computations.

Initial work on quantum logic synthesis utilized a two-stage scheme in which the desired quantum functionality is first synthesized into a quantum circuit composed of arbitrary one-qubit gates and so-called controlled NOT gates (using solutions as e.g. proposed in [3, 25, 25, 32, 34, 41]). Afterwards, all one-qubit gates are individually *compiled* to the targeted gate library (using solutions as e.g. proposed in [4, 10, 23] or, specifically targeting the Clifford+T

library, in [20–22, 38]). Since controlled NOT gates themselves are contained in the Clifford+T library, this yields the desired circuit description. However, following this approach has the main drawback that, despite the fact that it requires expensive decompositions and yields resulting quantum circuits with a tremendous number of gates, only an approximation of the desired quantum functionality is derived. In fact, the decompositions introduce numerical errors (through rounding) that are later on amplified during the compilation of one-qubit gates to Clifford+T. Hence, only an approximate realization is determined even if the initially given quantum functionality would allow for an exact one.

In order to overcome this drawback, researchers considered *exact* synthesis of Clifford+T quantum circuits in [2, 11, 13, 14, 22, 31], i.e. the desired quantum functionality is realized without any rounding errors rather than being approximated with respect to an ϵ. However, while the approach in [22] is restricted to 1-qubit operations, the approaches in [2] and [11, 14] perform an exhaustive search in the space of all Clifford+T circuits up to a given depth and are, thus, practically limited. Accordingly, these works only report results for circuits with up to three or four qubits (although the approach in [11] is based on a highly parallel implementation and was run on a supercomputer using 8192 cores). In contrast, the approach by Giles and Selinger [13] was the first to provide a generic, constructive synthesis algorithm based on algebraic properties of exactly synthesizable transformation matrices. However, while guaranteeing exactness, the respective scheme still bears lots of potential for improvement.

In fact, the solution proposed by Giles and Selinger [13] synthesizes the given unitary transformation matrix (representing the desired quantum functionality) in a local fashion, i.e. column by column. This is disadvantageous since the manipulation of a single column does have an effect to all other columns—frequently making the remaining columns harder to synthesize. Furthermore, the column-wise consideration yields rather expensive circuits, since often several columns could be considered much more efficiently in one step than in several individual steps (both issues are explained and illustrated in more detail later in Section 3). Overall, this leaves significant room for improvement. This is also confirmed by the authors of [13] stating that the resulting circuits are "far from optimal".

In this work[1], we are explicitly tackling these shortcomings. To this end, we first discuss the previously proposed synthesis approach and its shortcomings. Afterwards, we propose a global consideration of the unitary matrix to be synthesized which does not restrict to a single column during a particular synthesis step, but always keeps track of the entire matrix. This eventually yields an alternative and improved synthesis algorithm which allows to realize the desired quantum functionality significantly cheaper than before. Experimental evaluations confirm these benefits and show improvements of up to several orders of magnitude compared to previous work. However, due to the

[1] A preliminary version of this paper has been published in [29].

exponential complexity of representing and manipulating the desired quantum functionality, an implementation on top of straight-forward matrix representations quickly becomes infeasible (as confirmed by the experimental evaluation performed in [29]). In order to cope with this issue, we employ a dedicated data-structure for the compact representation and efficient manipulation of quantum functionality, namely the *Quantum Multiple-valued Decision Diagram* (QMDD, [30]). This improves the applicability of the proposed approach significantly, especially by exploiting certain QMDD characteristics, which is also confirmed by experimental evaluations.

The remainder of this work is structured as follows: The next section reviews some background on quantum computation as well as the considered Clifford+T library. Section 3 discusses and illustrates previously conducted synthesis of Clifford+T circuits—including an analysis of the corresponding shortcomings and their potential for improvement. This leads to an improved synthesis algorithm which is described in detail in Section 4. The implementation of the algorithm on top of the QMDD data-structure is outlined in Section 5. Finally, Section 6 summarizes the results of the conducted experimental evaluation before the paper is concluded in Section 7.

2 Background and Terminology

This section briefly reviews the basics of quantum computation and circuits as well as the Clifford+T gate library as required for the discussion in this paper. For a more detailed introduction to the field, we refer to [26].

2.1 Quantum Computation and Circuits

Quantum computations are performed on systems of *qubits*. Analogously to conventional bits, a qubit has two stable *basis states* commonly denoted as $|0\rangle$ and $|1\rangle$. Additionally taking into account the modeling of quantum-mechanical phenomena, qubits can also assume an arbitrary *superposition* $\alpha|0\rangle + \beta|1\rangle$ for complex-valued α, β with $|\alpha|^2 + |\beta|^2 = 1$. Accordingly, an n-qubit quantum system can be in one of 2^n basis states $(|0\ldots00\rangle, |0\ldots01\rangle, \ldots, |1\ldots11\rangle)$ or a superposition of these states. The state of such a quantum system is represented by a unit vector of dimension 2^n (denoted as *state vector*). The effect of applying a quantum operation is obtained by multiplying the state vector with a corresponding *unitary transformation matrix*, i.e. an invertible complex-valued matrix whose inverse is given by the adjoint matrix. In other words, all quantum operations are linear operations on the state space where the k-th column of the matrix describes the image of the k-th basis state.

Example 1 Commonly used quantum operations include

- the *Hadamard* operation H (setting a qubit into a balanced superposition, e.g. mapping $|0\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$),

- the *NOT* operation X (flipping the basis states $|0\rangle$,$|1\rangle$),
- as well as the *phase shift* operations T ($\pi/4$ gate), $S = T^2$ (Phase gate), and $Z = S^2 = T^4$.

The corresponding unitary matrices are defined as
$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$, $X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$, and $T = \begin{pmatrix}1 & 0\\ 0 & \omega\end{pmatrix}$ where $\omega = \frac{1+i}{\sqrt{2}} = e^{i\pi/4}$.
Besides these operations that work on a single qubit, there are also operations on multiple qubits. Usually, these realize *controlled operations* that only manipulate a single qubit (denoted as the *target qubit*) depending on the state of a set of *control qubits*. The most prominent example for such operations is the *controlled NOT* (CNOT) operation on two qubits whose transformation matrix is defined by

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

This operation performs a *NOT* operation on the target qubit if the control is in the $|1\rangle$-state and does not change anything (i.e. performs the identity transformation) if the control is in the $|0\rangle$-state.

Realizations of (complex) quantum operations are represented by *quantum gates* g_i which eventually form a *quantum circuit* $G = g_1 \dots g_d$ with $1 \leq i \leq d$. The unitary matrix of the entire circuit is computed as the matrix product of the individual gate matrices (in reversed order).

2.2 Clifford+T Library

In order to realize a (complex) quantum operation in a particular quantum technology, the operation has to be mapped to a dedicated—possibly technology-specific—gate library. In this work, we focus on the Clifford+T gate library [5] which is popular for its *universality* (any quantum operation, i.e. unitary transformation matrix, can be realized up to an arbitrary precision) as well as *fault-tolerance* (robust, fault-tolerant implementations of these gates are known for most technologies that are considered promising for large-scale quantum computers). The most elementary gates in this library are the Clifford group gates (H, CNOT, S) as well as the T gate, as discussed in Example 1, i.e. without any (additional) controls. While the Clifford group gates on their own only allow for the realization of a restricted set of quantum functionality, it is the T gate that makes the Clifford+T library a universal gate library. This is reflected in the fact that the cost of implementing a T gate (in a fault-tolerant way) is significantly higher—around a factor of 100—as compared to Clifford gates [12].

In addition to the basic, i.e. uncontrolled, Clifford+T gates, there are also well-known constructions for multiple-controlled versions of these gates (see

Table 1 Cost metric for multiple-controlled Clifford+T gates

#Controls	CNOT	H	T
0	0	0	1
1	0	1	5
2	2	7	21
3	12	21	33
4	32	33	69
5	68	69	105
6	104	105	129
⋮	+24 per add. control		

e.g. [13]). By employing the additional controls, the basic Clifford+T operations can be applied on a dedicated subset of basis states. At the level of unitary matrices, this corresponds to manipulating a certain subset of columns in the original matrix (c.f. the controlled NOT gate in Example 1). However, these constructions have a high realization cost as can be seen from the detailed overview on the costs of multiple-controlled Clifford+T gates provided in Table 1 (based on [2, 13]). Here, the cost is given in terms of T-depth, i.e. the number of sequential T gates in the circuit, assuming the availability of one additional helper qubit (*ancilla*). Regarding exact synthesis, Kliuchnikov et al. conjectured in [22] and Giles and Selinger proved in [13] that a unitary matrix can be realized exactly in the Clifford+T library (i.e. without any rounding error) with the help of at most one ancilla if, and only if, all entries are complex numbers of the form $\frac{1}{\sqrt{2}^k}(a\omega^3 + b\omega^2 + c\omega + d)$ for coefficients $a, b, c, d, k \in \mathbb{Z}$ and $\omega = \frac{1+i}{\sqrt{2}} = e^{i\pi/4}$ (as in Example 1).[2] Consequently, we will focus on such matrices in the following.

3 Synthesis of Clifford+T Circuits

In this work, we consider the synthesis of quantum functionality to elementary circuit descriptions based on the Clifford+T library. More precisely, the task is considered in which a quantum functionality represented in terms of a transformation matrix F is decomposed into a sequence $G = g_1 \dots g_d$ of elementary quantum operations (i.e. quantum gates g_i with $1 \leq i \leq d$). The resulting sequence eventually forms a quantum circuit and is supposed to be composed of Clifford+T gates as reviewed above. In the following, we re-visit the current state-of-the-art approach which has been proposed for this purpose and discuss its main drawbacks. Based on this, we sketch the general idea of the improved quantum circuit synthesis proposed in this work. Details on that are later provided in Section 4.

[2] From an algebraic perspective, these numbers form the ring $\mathbb{D}[\omega] = \mathbb{D}[\sqrt{2}, i]$ (where $\mathbb{D} = \{\frac{d}{2^k} \mid d \in \mathbb{Z}, k \in \mathbb{N}\} \subset \mathbb{Q}$ are the dyadic fractions). This ring has a variety of interesting properties, two of which we will later on make use of in Section 4.1, though without going into much detail, as this is beyond the scope of this paper. A more detailed discussion and derivation of these properties can be found in [13].

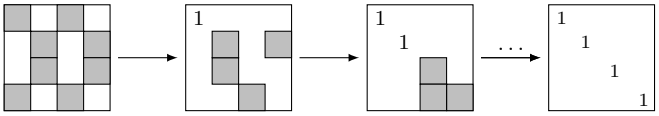


Fig. 1 Local, i.e. column-wise, synthesis scheme [13].

3.1 State-of-the-Art and Motivation

Figure 1 sketches the current state-of-the-art in the synthesis of Clifford+T quantum circuits as proposed by Giles and Selinger [13]. The main idea is to apply quantum gates so that, eventually, the given matrix to be synthesized (sketched on the left-hand side of Fig. 1) is transformed to the identity matrix (sketched on the right-hand side of Fig. 1). To this end, the matrix is transformed column by column (as sketched in the middle of Fig. 1). For each column, three steps are applied:

(a) *Eliminate superposition*, i.e. apply quantum gates so that all multiple non-zero matrix entries in the column are combined to a single non-zero entry.
(b) *Move to diagonal*, i.e. apply quantum gates which move the remaining non-zero entry to the diagonal of the matrix.
(c) *Remove phase shifts*, i.e. apply quantum gates which transform the diagonal entry to 1—eventually yielding a column of the identity matrix.

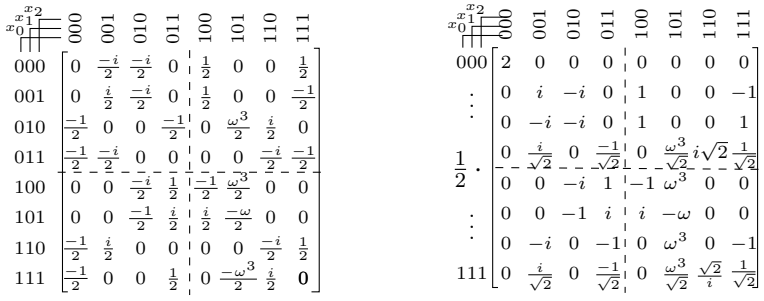
Each of these steps is achieved by using so-called *two-level operations* that modify pairs of entries in the current column. More precisely, the following operations are utilized:

- *Combine entries:* $(a, b) \Rightarrow \frac{1}{\sqrt{2}}(a + b, a - b)$
- *Exchange entries:* $(a, b) \Rightarrow (b, a)$
- *Modify phase shift:* $(a, b) \Rightarrow (a, b \cdot \omega)$ where $\omega = e^{i\pi/4}$.

Example 2 Consider the matrix in Fig. 2a which represents a quantum operation on 3 qubits x_0, x_1, x_2. The four non-zero entries in the first column can be combined pairwise from $(\frac{-1}{2}, \frac{-1}{2})$ to $(\frac{-1}{\sqrt{2}}, 0)$. The resulting pair $(\frac{-1}{\sqrt{2}}, \frac{-1}{\sqrt{2}})$ is combined to $(-1, 0)$. Finally, the -1 is exchanged with the 0 entry in the first row and a phase shift by -1 is applied. This leads to the matrix shown in Fig. 2b where the first column is of the desired form (note the extracted scalar factor of $\frac{1}{2}$).

However, this approach has two major drawbacks:

1. Two-level operations do not solely have an effect on that particular column, but on all columns of the matrix. Consequently, locally synthesizing one column without taking the global view, i.e. the remaining columns, into consideration may significantly worsen the degree of superposition in these columns (as already became evident in the previous example where the overall number of non-zero entries increased from Fig. 2a to Fig. 2b).



(a) Initial Matrix (b) After synthesizing the first column

Fig. 2 Application of the local, i.e. column-wise, synthesis scheme.

2. Two-level operations rely on multiple-controlled Clifford+T gates: Hadamard for combining, CNOT for exchanging, and T for modifying the phase shift between entries. These gates have many controls (in fact, $n-1$ controls where n is the number of qubits). Since the number of control lines significantly increases the costs of those gates (cf. Table 1), this leads to high costs of the resulting circuits.

Regarding the first drawback, it seems to be promising to globally consider the whole matrix at once and to only apply operations that do not lead to a worsening in any other column. Overall, this may lead to fewer steps to be conducted in order to derive the desired identity matrix.

Regarding the second drawback, it can be observed that Clifford+T matrices often exhibit similar structures occurring globally throughout the matrix. In many cases, the corresponding two-level operations applied to these structures are similar and can be combined to a joint, again global, operation that can be realized with lower costs. That is, controls (which enforce a local change in the matrix only) often can be saved as the correspondingly more global change can be conducted at once.

Example 3 The first two combine operations in the previous example correspond to multiple-controlled Hadamard gates with a target on qubit x_2 and controls on x_0, x_1. The only difference between the gates is that the control connection on x_0 is positive in one case and negative in the other case. Thus, this control can be dropped and both gates can be compressed to a single-controlled Hadamard gate. Hence, the two steps cannot only be conducted with a single gate (rather than two); but the required gate is even significantly cheaper. Overall, this reduces the cost from $2 \cdot 7 = 14$ to 1 (cf. Table 1).

Note that simple optimizations as described in Example 3 could probably be conducted as a post-synthesis optimization process. However, such optimizations can only be found if the corresponding quantum gates are located close to each other in the final circuit. To this end, the approach from [13]

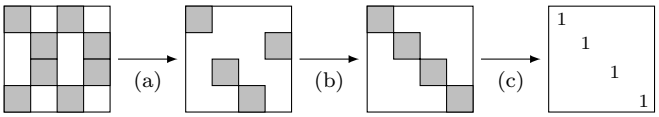


Fig. 3 Proposed global synthesis scheme.

yields circuits which have a structure that is completely unsuitable for such post-synthesis optimization. This is because the two-level operations eventually have to be decomposed into elementary quantum gates which hardly allow the detection of redundancies any more. In fact, the resulting cascades of CNOT gates will, in the vast majority of cases, prevent that operations on rows/columns whose index differs in one position only appear close to each other.

3.2 General Idea of Proposed Approach

Inspired by these observations, the general idea of the proposed approach for logic synthesis of Clifford+T circuits is the following: Globally consider multiple columns simultaneously and establish recurrent structures that allow to combine as many two-level operations as possible. More precisely, the three steps (a)-(c) that have so far been conducted locally for the individual columns are now globally performed on the entire matrix as sketched in Fig. 3:

(a) Regarding the elimination of superposition, we aim to gradually reduce superposition in the whole matrix, i.e. establish recurrent structures that allow to reduce the superposition in all columns at once. To this end, all entries of the matrix have to be rearranged and (possibly) phase shifted in such a way that all entries form suitable pairs that can be combined.

Example 4 Consider again the matrix from Fig. 2a. After exchanging rows 011 and 111 and applying a phase shift by i to row 101, all entries are grouped in pairs that can be combined with each other. Thus, an uncontrolled Hadamard gate can be applied on qubit x_2. As a consequence, the number of non-zero entries in the matrix is reduced by a factor of 2 in a single step. The resulting matrix is shown in Fig. 4a. Here, only rows 101 and 111 need to be swapped before another Hadamard gate on qubit x_0 eliminates the remaining superposition and leads to the matrix shown in Fig. 4b.

(b) Regarding the movement of entries to the diagonal, there is a large body of research on how to achieve this for permutation matrices (i.e. transformation matrices with Boolean entries only). In fact, this problem is known as *reversible circuit synthesis* and most of the approaches employed for this purpose (see e.g. [16, 33, 35, 37, 44]) are based on multiple-controlled Toffoli gates (which are exactly realizable in the Clifford+T library; see e.g. [13, Sec. 5]). Taking into account that potential phase shifts within the matrix do not affect the applicability of these (highly optimized) approaches, any of these can be utilized here.

Example 5 Consider the resulting matrix from the previous step (shown in Fig. 4b). Applying a CNOT gate with control qubit x_2 and target qubit x_0 exchanges rows 001 and 011 with 101 and 111, respectively. This establishes zero sub-matrices in the upper-left and lower-right quadrant of the matrix and moves the $-i$ in row 101 to the diagonal (as shown in Fig. 4c).

Finally, rows 000 and 010 as well as 101 and 111 can be swapped by two CNOTs on x_1 (one with a negative control on x_0 and the other with a positive control on x_2). This leads to the diagonal matrix depicted in Fig. 4d.

(c) Finally, regarding the removal of phase shifts, similar phase shifts can be taken care of simultaneously and the corresponding two-level operations can be joined.

Example 6 After shifting the phase of row 101 by $-\omega$ (from $\frac{-1+i}{\sqrt{2}} = \omega^3$ to $\omega^2 = i$) and the phases of rows 001, 010, 101, and 110 by $-i$ (from $\pm i$ to ± 1), the remaining phase shifts can be removed by a Z gate on x_0 (without controls).

Overall, this yields substantial improvements compared to the local, i.e., column-wise, synthesis scheme for all three steps. How these ideas are implemented in detail is described in the following two sections. Afterwards, Section 6 shows the resulting improvements by means of a summary of the conducted experimental evaluations.

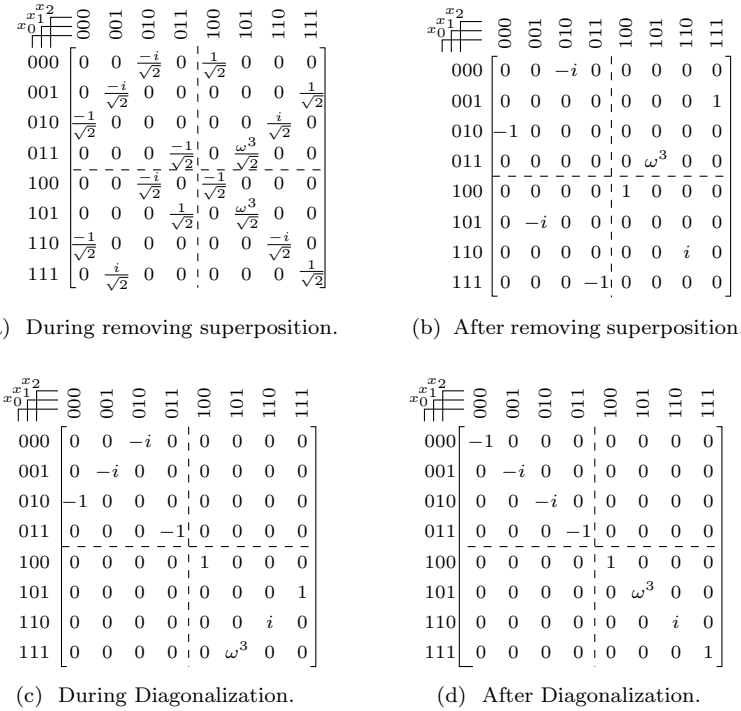
4 Implementation

The proposed approach is described along the three steps discussed above and summarized in Fig. 3.

4.1 Eliminating Superposition

As discussed in Section 2.2, each entry of a Clifford+T matrix can be written as $\frac{1}{\sqrt{2}^k}(a\omega^3 + b\omega^2 + c\omega + d)$. More precisely, one can show that, for $\alpha \neq 0$, there exists a representation with a unique *smallest denominator exponent* $k_{\min}$ such that integer coefficients a, b, c, d can be found for $k_{\min}$, but not for any greater $k > k_{\min}$.[3] The maximum of these exponents in the entire matrix (denoted as $k_{\max}$ in the following) determines the degree of superposition and needs to be decreased to 0 in order to completely eliminate superposition. The only way to potentially reduce $k_{\max}$ is the application of (controlled) Hadamard gates. In fact, a Hadamard gate on qubit x_i combines pairs of entries of the matrix whose row index only differs at position x_i. More precisely, two entries α, β are replaced by their (weighted) sum/difference $\frac{1}{\sqrt{2}}(\alpha + \beta)$ and $\frac{1}{\sqrt{2}}(\alpha - \beta)$, respectively.

[3] For more detailed discussion on that we refer to [13]. For the special case of 0-entries we assume a representation via $k = a = b = c = d = 0$.



(a) During removing superposition.

(b) After removing superposition.

(c) During Diagonalization.

(d) After Diagonalization.

Fig. 4 Application of the proposed global synthesis scheme.

In the case that the weighted sum/difference indeed has a smaller exponent than α, β, it is required that both have the same exponent k and that the parity of the coefficients a, b, c, d, i.e. their remainder modulo 2, is pairwise the same.[4] To this end, [13] introduces the notion of *residues* where the (k-)residue of $\frac{1}{\sqrt{2}^k}(a\omega^3 + b\omega^2 + c\omega + d)$ is defined as $(a\%2, b\%2, c\%2, d\%2)$ where $\%$ denotes the modulo operation (e.g., the residue of $5\omega^3 + 2\omega^2 - \omega$ is $(1, 0, 1, 0)$). Accordingly, a pair (α, β) is termed *reducible* and α, β are termed *reduction partners* if, and only if, both have the same exponent k and the same (k-)residue. Using this notation, the resulting algorithm to gradually eliminate superposition is as follows:

1. If $k_{\max} = 0$, terminate, as there is no more superposition in the matrix. Otherwise, identify and restrict to the entries with the maximum smallest denominator exponent $k_{\max}$ (i.e., consider the other entries as being 0 in the following).
2. Determine the most promising qubit x_p for which we expect to require a smaller number of matrix operations in order to prepare the matrix for the

[4] Roughly speaking, the multiplicative factor $\frac{1}{\sqrt{2}}$ increments the exponent such that k can only be decreased if a factor of 2 can be extracted from $\alpha \pm \beta$. This, in turn, is only possible if the coefficients of α, β have the same parity.

application of a (controlled) Hadamard gate. To this end, first the residue of all entries is computed. Then, to determine the number of already existing reduction partners for the first qubit, we compare entries in adjacent rows (whose row index only differs at the last position). Likewise for the second qubit, we compare entries in rows whose row index only differs at the second-last position and so on. Finally, the qubit with the maximum number of reduction partners is chosen. In case of a tie, the first qubit with this property is chosen.

3. Create a list R of row indices of all rows that contain at least one non-zero entry which does not have a reduction partner w.r.t. x_p so far. Mark all columns as unvisited.
   While $R \neq \emptyset$ and there is an unvisited column left:
   - Consider the leftmost unvisited column c, create a list P_c containing all row indices corresponding to non-zero entries in that column (within the rows from R) that do not have a reduction partner (w.r.t. x_p).
   - While $|P_c| \geq 2$:
     - Choose two row indices $A, B \in P_c$, preferably such that the corresponding entries in column c (α_c and β_c) have the same residue.
     - Move row B to the appropriate position such that its row index differs from A only at position x_p, i.e. such that α_c, β_c will be combined by a Hadamard gate on x_p. Align the residues of α_c and β_c (if necessary) and remove the indices of A, B as well as of all further rows with newly established reduction partners (in column c) from P_c and R.
   - Mark the column as visited.
4. If all entries have a suitable partner w.r.t. x_p, apply the (controlled) Hadamard gate directly and proceed with Step 1. Otherwise, eliminate superposition in the first column of the matrix using columnwise synthesis (do not perform diagonalization or elimination of phase shift) and continue with Step 1 on the remaining columns.

As each iteration of the algorithm either reduces $k_{\max}$ in the entire matrix or yields one additional column with a single non-zero entry, it is guaranteed to terminate at some point— yielding a matrix with exactly one non-zero entry per row.

Concerning the movement of rows in the inner while-loop in Step 3, note that a movement is necessary if, and only if, the indices A, B differ in at least one position $x_d \neq x_p$. Then, CNOTs with a control on x_d can be used to align the indices of A, B (except for position x_d) and, if necessary, establish different entries at position x_p—such that only x_d needs to be aligned in order to move B to the desired position. Finally, the x_d position of A, B as well as the residues of α_c, β_c can be aligned by applying multiple-controlled NOT and T/H gates on x_d (with controls corresponding to the joint index of A, B). The existence of an adequate sequence of T and H gates is guaranteed by Lemma 4 (row operation) from [13]. All these gates preserve already established pairs of reduction partners, but may—by chance—create additional pairs.

Example 7 Consider the matrix in Fig. 4a. Here, all non-zero entries have the same exponent $k_{\max} = 1$, so the entire matrix is considered in Step 1. As there are already four pairs of reduction partners (in columns 000, 010, 100 and 110), $x_p = x_0$ is chosen in Step 2 of the algorithm. Entries without reduction partner are found in rows $R = \{001, 011, 101, 111\}$. For the first column, there are no entries in these rows, i.e. $P_{000} = \emptyset$. For the second column, we have $P_{001} = \{001, 111\}$. Both non-zero entries have the same residue. As the indices already differ at exactly two positions $x_0 = x_p$ and $x_1 = x_d$, a two-controlled NOT on x_1 with positive controls on x_0 and x_2 ($1X1$) is employed to exchange rows 101 and 111. This gate does not only create a pair of reduction partners in column 001, but establishes pairs of reduction partners in all remaining rows from R, such that $R = \emptyset$ and an H gate is applied to qubit x_0 in Step 4. This yields the matrix in Fig. 4b where $k_{\max} = 0$, i.e. the algorithm terminates.

4.2 Moving to the Diagonal

In the diagonalization phase, all non-zero entries need to be moved to the diagonal. As already outlined above, there are various approaches available to conduct this task. We propose to use the approach from [37] which is one of the most efficient and scalable approaches for this purpose. Moreover, its ability to exploit recurrent structures in order to reduce the cost of the resulting circuit (T-depth) has been demonstrated in [43]. The general idea of this approach is to successively establish a block matrix structure, as already outlined in Example 5. More precisely, the first step is to swap the columns of the matrix such that all non-zero entries are gathered in the top-left as well as bottom-right quadrant of the matrix, while the off-diagonal quadrants, i.e. the top-right and bottom-left quadrant, become zero matrices. In the following steps, the same procedure is likewise applied to the smaller sub-matrices (top-left and bottom-right quadrant) that were obtained in the previous step until a diagonal matrix is established. For more details about the approach, we refer the interested reader to [37, 43, 44].

4.3 Removing Phase Shifts

Finally, we end up with a diagonal matrix whose entries all have norm 1 which means that they are of the form ω^m for $m \in \{0, \ldots, 7\}$ (as shown in [13, Lemma 5]). In order to remove possible phase shifts, all the exponents have to be transformed to the same value m_0. Note that we do not require $m_0 = 0$ such that the resulting matrix might not be the identity matrix itself, but is equivalent up to *global phase* and, hence, physically indistinguishable [26].

In order to align the exponents, the approach from [38] could be applied. However, it is targeted to approximating arbitrary diagonal matrices, while the matrices considered here have the above-mentioned restriction that all diagonal entries are of the form ω^m. Thus, we apply a customized approach. At

first, we take care of odd exponents. Instead of applying a multiple-controlled T gate for each and every of these columns, we aim to merge multiple gates by using techniques from classic logic synthesis. To this end, we interpret the indices of the corresponding rows as the ON-set of a Boolean function and synthesize this function in terms of an *Exclusive-Sum-Of-Products* (ESOP) [6]. The individual products of this representation are then mapped to controlled T gates (with fewer controls). Afterwards, all exponents in the matrix are even (i.e. all entries are from the set $\{\pm 1, \pm i\}$). Then, we proceed similarly for all imaginary entries and, once these became ± 1, finally also for the -1 entries.

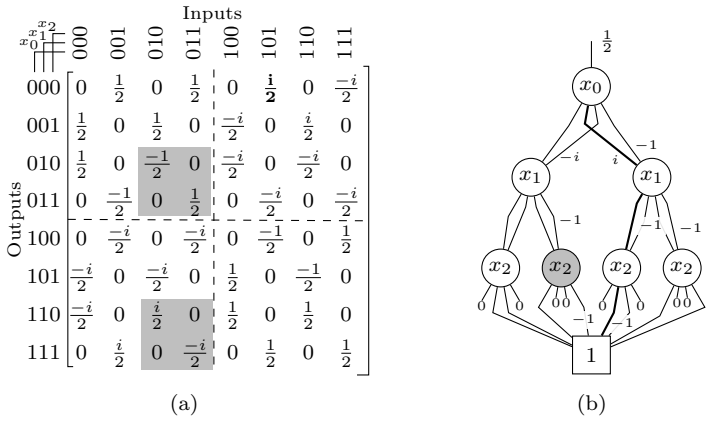
Example 8 Consider the matrix in Fig. 4d. There is a single entry with an odd exponent, namely ω^3 in row 101. The corresponding Boolean function is given as $f = x_0 \cdot \overline{x}_1 \cdot x_2$. Thus, a two-controlled $T^\dagger$ gate (inverse of T) on x_2 with a positive control on x_0 and a negative control on x_1 shifts this entry to $\omega^2 = i$. Then, the indices of the imaginary entries are given by the ON-set of $g = x_1 \oplus x_2$. Consequently, two uncontrolled S gates on x_1 and x_2 shift these entries to ± 1 such that the Boolean function for all rows with a -1 entry (namely 000,101,110,111) becomes: $h = x_0 \oplus \overline{x}_1\overline{x}_2$. The first term directly translates to an uncontrolled Z gate on qubit x_0. However, since only positive literals can be used as targets of the phase shift gates, some more work is required for the second literal. More precisely, we first need to apply a basis transformation using an uncontrolled NOT gate on x_2, then perform the phase shift by a controlled Z gate on x_2 (with a negative control on x_1) a finally undo the basis transformation with another uncontrolled NOT gate on x_2.

Overall, performing the three steps as described above eventually transforms any given Clifford+T matrix F to the identity and yields a quantum circuit that realizes F by solely using Clifford+T gates.

Regarding the complexity of the proposed algorithm, notice that Step 4 of the algorithm for eliminating superposition (cf. Section 4.1) uses the column-wise synthesis from [13] as a fallback. Thus, the worst-case complexity of the proposed algorithm is not improved w.r.t. [13]. However, the experimental evaluations (summarized in Section 6) indeed demonstrate significant improvements of the proposed method compared to the previous work.

5 Exploiting Decision Diagrams

Due to the exponential complexity of representing the desired quantum functionality, using straight-forward matrix representations quickly becomes infeasible. For instance, the reference implementation of the approach from [13] (written in Haskell) failed to produce results for more than seven qubits. In order to cope with this issue and to obtain an implementation of the proposed approach that allows for an application to larger problem instances, we additionally employ decision diagrams, namely QMDDs as reviewed in the following, for a more efficient processing of the unitary matrices.



(a) (b)

Fig. 5 Matrix and QMDD representation of a 3-qubit quantum circuit.

5.1 QMDDs

As the unitary matrices corresponding to quantum operations grow exponentially with the size of the quantum systems, dedicated data-structures are required that exploit redundancies in the matrices in order to enable a more compact representation and efficient manipulation. A very promising candidate for this task is given by the *Quantum Multiple-Valued Decision Diagram* (QMDD, [30]). The general idea behind QMDDs is to represent a (unitary) matrix in terms of a directed acyclic graph such that sub-matrices which occur multiple times are represented by a shared graph structure. While there are several data-structures that follow a similar approach, only QMDDs additionally make use of weighted edges. This unique property exclusively allows them to use shared structures also for sub-matrices which differ by a scalar factor—a case that occurs quite often for the unitary matrices considered in quantum computation.

The QMDD for a given unitary matrix is constructed by a recursive partitioning process, i.e. a $2^n \times 2^n$ matrix is partitioned in to four $2^{n-1} \times 2^{n-1}$ matrices, as illustrated in the following example.

Example 9 Fig. 5a shows a transformation matrix for which a QMDD as shown in Fig. 5b 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 (by default: such that the "leftmost" edge with a non-zero weight has weight 1). This factor is propagated to incoming edges, e.g. the factor $\frac{1}{2}$ is propagated upwards from the x_2-level to the x_0-level in Fig. 5b. By this, structurally equiv-

alent sub-matrices, i.e. sub-matrices that are equal as well as sub-matrices that only differ by a scalar factor, are compressed to a shared vertex (highlighted in grey in Fig. 5a and 5b, 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 (here: $\frac{1}{2}$).

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. 5a (highlighted bold) can be determined as the product of the weights on the highlighted path of the QMDD in Fig. 5b.

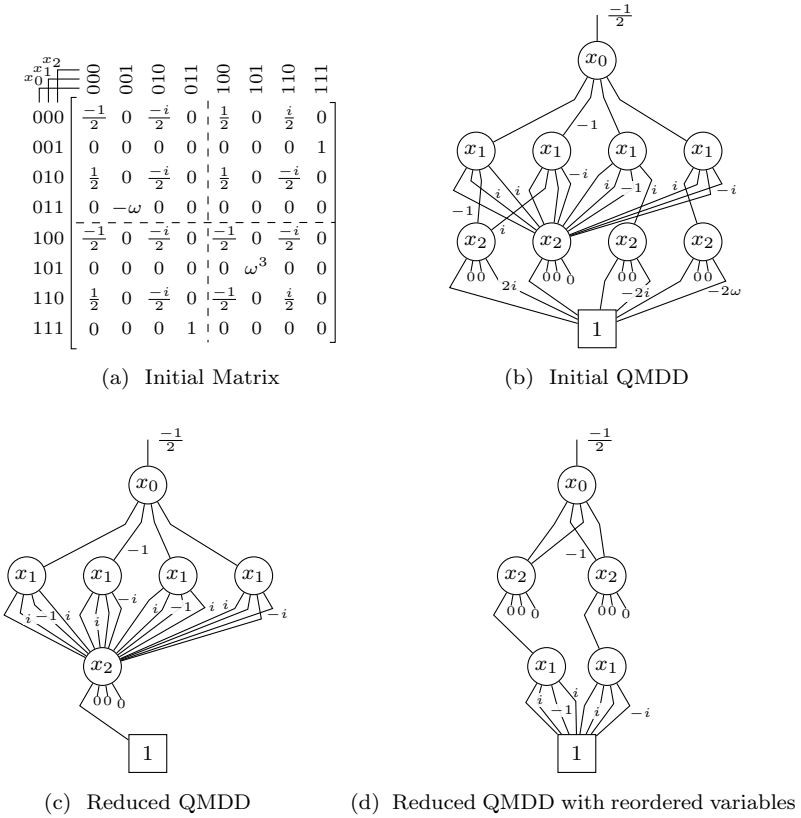
Moreover, efficient algorithms have been presented for applying operations like matrix addition or multiplication directly on the QMDD data-structure. Overall, QMDDs allow for both, a compact representation as well as an efficient manipulation of unitary matrices for quantum systems of considerable size. As a consequence, they have already been used in a broad variety of applications in the design of quantum circuits (e.g., verification [8,9,28,39,42], simulation [45, 47], or synthesis [27, 37, 44]).

5.2 Exploiting QMDDs for an Enhanced Applicability

Accordingly, we are exploiting these benefits of QMDDs for the proposed synthesis scheme. However, in addition to their general efficiency w.r.t. matrix manipulation, we identified particular characteristics of QMDDs that offer further potential for a speed-up of the algorithm. In the following, we provide details of the implementation that illustrate this potential:

Restriction to Entries With Maximum Denominator Exponent An important preprocessing step of the first step of the algorithm is the restriction to matrix entries with the maximum denominator exponent (k_{max}). In order to facilitate this step in QMDDs, a normalization scheme can be chosen that extracts the maximum smallest denominator exponent from the (outgoing) edge weights and propagates it to the incoming edges of a vertex. By this, k_{max} will occur as the exponent of the root edge weight and all other weights in the QMDD will have a denominator exponent less than or equal to 0. As a consequence, only those parts of the QMDD need to be considered that are reachable by edges with a denominator exponentof 0. If an edge has a negative denominator exponent, all entries in the represented sub-matrix have a denominator exponent that is less than k_{max} and can, thus, be ignored (i.e. the edge is modified to represent a zero matrix).

Example 10 Consider the QMDD in Fig. 6b which represents the matrix from Fig. 6a. As expected, the exponent $k_{max} = 2$ occurs in the root edge weight $\frac{-1}{2} = \frac{-1}{\sqrt{2}^2}$, while all other weights have a zero or negative exponent (n.b.



(a) Initial Matrix (b) Initial QMDD

(c) Reduced QMDD (d) Reduced QMDD with reordered variables

Fig. 6 Reduction to entries with exponent k_{max}.

$2 = \frac{1}{\sqrt{2}^{-2}}$). By pruning all parts that are reached via an edge with a negative exponent, we obtain the QMDD shown in Fig. 6c where all vertices at the x_2 level are collapsed to a single vertex.

Determining the Most Promising Qubit for Hadamard Application An even more important step of the algorithm is to determine the most promising qubit by counting the number of already existing pairs of reduction partners in all columns. In order to facilitate this computation in the QMDD, the qubit under consideration (i.e. the corresponding variable) is moved to the bottom of the QMDD by pairs of adjacent variables as shown in [30]. Then each vertex at that level represents two pairs of matrix entries that would be combined by applying a Hadamard at the respective circuit line. If, and only if, both of these are already suitable reduction partners (i.e. they have the same residue), the vertex represents a reducible pattern. If all vertices represent a reducible pattern, a Hadamard gate can be applied directly. Otherwise, we can obtain the

number of existing reduction partners in the matrix by counting the number of paths to the respective vertices in the QMDD.

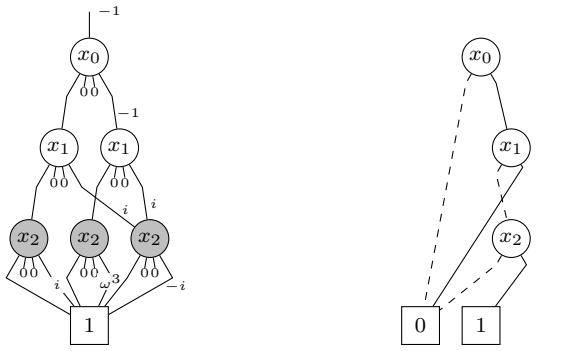
Example 11 Consider the QMDD in Fig. 6c. In the single x_2 vertex, there is a single non-zero edge weight, i.e. the corresponding entry does not have a suitable reduction partner. Consequently, x_2 is not a promising qubit for Hadamard application. By exchanging the adjacent variables x_1 and x_2 we obtain the QMDD depicted in Fig. 6d. Here, the weights of the first and third (second and fourth) outgoing edge in left-most x_1 vertex only differ by a factor of -1. Thus, they have the same residue and correspond to a pair of reduction partners. As there are two paths to this vertex, the corresponding pattern occurs twice in the matrix. As the other x_1 vertex also represents a reducible pattern, a Hadamard gate can be applied directly to qubit x_1.

Determining Phase Shifts of Diagonal Entries In order to determine which diagonal entries have a phase shift that needs to be removed, we can again exploit characteristics of QMDDs. To this end, recall that all diagonal entries are potencies of ω, such that all edge weights in the QMDD also will be potencies of ω. Moreover, recall that our aim in the first step is to express the row indices of all entries with an odd potency as a Boolean function. We construct this function in terms of a corresponding decision diagram representation (*Binary Decision Diagram*, BDD [7]). To this end, we traverse the QMDD in a depth-first manner. When the terminal is reached, a Boolean 0 (0-terminal) is returned. For each non-terminal vertex, a BDD vertex is constructed by taking the resulting BDDs from the first and fourth edge as the low/high child. More precisely, if the corresponding edge weight is an even potency of ω, the original BDD is used, while the negated BDD is used for odd potencies. Note that the resulting BDDs can be stored in a computed table and can be re-used directly without any further computation when the vertex is processed again during the traversal of the QMDD.

Example 12 The QMDD in Fig. 7a represents the matrix from Fig. 4d. The edge weights of the left- and right-most x_2 vertex are either zero or an even potency of ω. Consequently, these vertices return a constant 0 function (represented by the 0-terminal). As the edges pointing to these vertices from the left-most x_1 vertex are annotated with an even potency of ω, the resulting BDD vertex is redundant (both children point to the 0-terminal) and can be removed. Likewise, the x_2 vertex in the center of the QMDD yields a BDD vertex, whose low child is pointing to the 0-terminal (edge weight 1) and whose high child is pointing to the 1-terminal, i.e. the negation of the 0-terminal (edge weight ω^3). Overall, the BDD in Fig. 7b is constructed where edges pointing to the low/high child are indicated by dashed/solid lines.

Likewise, BDDs can be constructed for the row indices that correspond to imaginary or -1 entries, as required in the last steps of the algorithm.

Overall, QMDDs exhibit multiple useful properties that can readily be facilitated for a speed-up of the algorithm—on top of their general efficiency



(a) QMDD of the diagonal matrix (b) Resulting BDD

Fig. 7 Constructing BDD for the indices of odd potencies of ω.

for processing huge matrices that has been demonstrated in a broad variety of applications. The resulting benefits of the implementation on top of QMDDs will be evaluated in the next section.

6 Experimental Results

In this section, we evaluate the results obtained by the approach and compare them to the synthesis scheme previously proposed in [13] in order to demonstrate whether the proposed heuristic is indeed beneficial and overcomes the conceptual shortcomings of previous work (discussed in Section 3), although it has the same worst-case complexity.

To this end, the global synthesis approach discussed above has been implemented in C on top of the QMDD data-structure as outlined in the previous section. Moreover, also the approach from [13] has been re-implemented on top of QMDDs in order to benefit from the efficient matrix processing (two-level operations correspond to certain matrix multiplications). In fact, we found that the straight-forward matrix representations used in the preliminary version of the paper [29] had a very poor scalability and were not able to provide results for circuits with more than 7 qubits due to the expensive matrix multiplications. Motivated by this, we aimed for using a more efficient matrix representation in order to make the approaches applicable to larger problem instances and employed QMDDs for this purpose. As benchmarks, we used

- arbitrary transformation matrices for medium-sized quantum systems (denoted *arbitrary* and covering various cases of multiple smallest denominator exponents in the same matrix),
- quantum functionality taken from [24] and realizing Shor's 9-qubit error correcting code (denoted by *9qubitN1* and *9qubitN2*), a 7-qubit encoding (denoted by *7qubitcode*), and an error syndrome measurement circuit for a 5-qubit code (denoted by *5qubitcode*), as well as
- several classical reversible functions from RevLib [40].

Table 2 Experimental evaluation

Benchmark	#Qubits	State-of-the-Art [13] Costs	Proposed Costs
arbitrary3a	3	94	12
arbitrary3b	3	122	14
arbitrary4a	4	324	76
arbitrary4b	4	1,080	296
arbitrary5a	5	1,696	122
arbitrary5b	5	5,872	428
arbitrary6a	6	7,680	626
arbitrary6b	6	47,649	1,536
arbitrary7a	7	16,736	1,275
arbitrary7b	7	124,927	3,067
arbitrary8	8	T/O	169,220
arbitrary9	9	T/O	482,927
arbitrary10	10	T/O	1,223,376
arbitrary11	11	T/O	2,564,996
arbitrary12	12	T/O	5,658,252
arbitrary13	13	T/O	14,823,945
arbitrary14	14	T/O	33,892,212
arbitrary15	15	T/O	73,168,799
7qbitcode	7	32,869	148
5qbitcode	9	T/O	9,100
9qbitcodeN1	9	173,424	104
9qbitcodeN2	17	150,759,616	60
4mod5-bdd_287	7	7,904	160
alu-bdd_288	7	6,656	412
f2_232	8	23,552	264
rd53_251	8	24,576	1,056
dc1_220	11	307,200	412
z4_268	11	294,400	3,360
cm152a_212	12	573,440	256
0410184_169	14	4,299,776	149,060
cm42a_207	14	3,342,336	348
cnt3-5_179	16	19,193,600	16,142

Benchmark: Name of benchmark – #Qubits: Number of qubits – Costs: Costs w.r.t. T-depth – T/O: time-out

Note that while random benchmarks are kind of unusual in the conventional logic synthesis community (where a wide range of established benchmark libraries exist), this is completely different in the quantum computing community, where random benchmarks are *the* main means to evaluate approaches. Most prominently, this can be observed in the domain of quantum simulation, where random circuits are used to show quantum supremacy. But also the "big players" in the field frequently rely on random benchmarks as can e.g. be seen by the recent competitions conducted by IBM [1, 46].

The results are summarized in Table 2. The first column provides the identifiers of the respective benchmarks followed by its number of qubits. In the remaining columns, the costs of the resulting circuits are provided. As it is a common understanding that (physical) implementations of T gates are significantly more complex than those of Clifford group gates, the costs are provided

in terms of T-depth, i.e. the number of sequential T gates that cannot be conducted in parallel. In order to compute the cost for two-level operations and multiple-controlled Clifford+T gates, we employed the cost metric from Table 1 (based on the elementary circuits and decompositions provided in [2, 13] and assuming the availability of one ancillary qubit).

All experiments have been conducted on a 2.8 GHz Intel Core i7 machine with 8 GB of main memory running Linux. While our QMDD-based implementation of the proposed global synthesis approach easily managed to process matrices for larger quantum systems, the implementations of the approach from [13] (both the reference implementation written in Haskell and the re-implementation on top of QMDDs) in most cases failed to produce results for more than 7 qubits—thereby essentially limiting the size of comparable benchmarks to quantum systems of that size (the time-out was set to 3600 CPU seconds).

Table 2 clearly shows that, using the proposed method, much more compact quantum circuits can be realized for Clifford+T functionality compared to the state-of-the-art approach from [13]. In fact, significant reductions (of up to several orders of magnitudes) can be obtained. The tremendous cost reductions for the error correction circuits can be explained by the fact that the corresponding circuits consist of H and CNOT gates only and that all H gates are located at the very beginning and the very end of the circuit. The approach proposed by us inherently identifies these H gates and, hence, can eliminate the superposition without the need of any T gate—which obviously makes the realization way cheaper.

Besides, the significant reductions for the random benchmarks are a consequence of the "global" view taken by the proposed approach (instead of the local view in [13]). As motivated in Section 3.1, the resulting sequence of two-level operations allows for combining multiple two-level operations to a joint operation that can be realized with lower costs (c.f. Example 3). Thus, the conducted evaluations confirm that the heuristic optimization implemented by the proposed approach is indeed able to overcome the drawbacks of previous works.

Moreover, the results show the enhanced applicability which became possible due to the use of QMDDs as the underlying data-structure—in fact, much larger quantum functionality can be handled now.

7 Conclusions

In this work, we proposed an improved approach for the synthesis of quantum functionality in terms of Clifford+T quantum circuits. To this end, we explicitly addressed shortcomings of previously proposed synthesis, which relies on a local, i.e. column-wise, consideration of the given transformation matrix. The proposed method considers this matrix globally—thereby allowing to conduct several transformations at once and with significantly smaller costs. Although the proposed method is a heuristic optimization with the same worst-case

complexity as previous works, experimental evaluations showed that it yields Clifford+T quantum circuits with up to several orders of magnitude smaller costs. To this end, note that while the proposed method aims to determine a circuit realization with a minimized number of T gates, it is not able to determine whether it found the actual minimum (a problem coined COUNT-T in [14]).

In order to enhance the applicability of the approach, we employed more efficient matrix representations (in terms of QMDDs). For future work, one may also have a look into other matrix representations, though we do not expect this to provide a significant benefit, as the improvements gained by the use of QMDD essentially concern the run-time and general applicability of the approach, but not the cost of the resulting circuits. Thus, we rather plan to investigate the further potential for optimization that is, e.g., offered by the degree of freedom that exists when permuting the rows between the application of the Hadamard gate in step (a) of the proposed algorithm.

Acknowledgments

This work has partially been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria as well as by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG).

References

1. We have winners! ... of the IBM qiskit developer challenge. https://www.ibm.com/blogs/research/2018/08/winners-qiskit-developer-challenge/. Accessed: 2020-02-20
2. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Trans. on CAD **32**(6), 818–830 (2013). DOI 10.1109/TCAD.2013.2244643
3. Barenco, A., Bennett, C.H., Cleve, R., DiVinchenzo, D., Margolus, N., Shor, P., Sleator, T., Smolin, J., Weinfurter, H.: Elementary gates for quantum computation. The American Physical Society **52**, 3457–3467 (1995)
4. Bocharov, A., Roetteler, M., Svore, K.M.: Efficient synthesis of universal repeat-until-success quantum circuits. Physical review letters **114**(8), 080502 (2015)
5. Boykin, P.O., Mor, T., Pulver, M., Roychowdhury, V., Vatan, F.: A new universal and fault-tolerant quantum basis. Information Processing Letters **75**(3), 101–107 (2000)
6. Brayton, R., Mishchenko, A.: ABC: An academic industrial-strength verification tool. In: Computer Aided Verification, pp. 24–40 (2010). DOI 10.1007/978-3-642-14295-6_5
7. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Comp. **35**(8), 677–691 (1986)
8. Burgholzer, L., Wille, R.: Improved DD-based equivalence checking of quantum circuits. In: ASP Design Automation Conf., pp. 127–132 (2020)
9. Burgholzer, L., Wille, R.: Advanced equivalence checking for quantum circuits. arXiv preprint arXiv:2004.08420 (2020)
10. Dawson, C.M., Nielsen, M.A.: The solovay-kitaev algorithm. Quantum Info. Comput. **6**(1), 81–95 (2006). URL http://dl.acm.org/citation.cfm?id=2011679.2011685
11. Di Matteo, O., Mosca, M.: Parallelizing quantum circuit synthesis. Quantum Science and Technology **1**(1), 015003 (2016)

12. Fowler, A.G., Stephens, A.M., Groszkowski, P.: High-threshold universal quantum computation on the surface code. Phys. Rev. A **80**, 052312 (2009). DOI 10.1103/PhysRevA.80.052312. URL https://link.aps.org/doi/10.1103/PhysRevA.80.052312
13. Giles, B., Selinger, P.: Exact synthesis of multiqubit Clifford+T circuits. Phys. Rev. A **87**(3), 032332 (2013). DOI 10.1103/PhysRevA.87.032332
14. Gosset, D., Kliuchnikov, V., Mosca, M., Russo, V.: An algorithm for the t-count. Quantum Information & Computation **14**(15-16), 1261–1276 (2014). URL http://www.rintonpress.com/xxqic14/qic-14-1516/1261-1276.pdf
15. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Theory of computing, pp. 212–219 (1996)
16. Gupta, P., Agrawal, A., Jha, N.K.: An algorithm for synthesis of reversible logic circuits. IEEE Trans. on CAD **25**(11), 2317–2330 (2006)
17. Houshmand, M., Sedighi, M., Zamani, M.S., Marjoei, K.: Quantum circuit synthesis targeting to improve one-way quantum computation pattern cost metrics. J. Emerg. Technol. Comput. Syst. **13**(4), 55:1–55:27 (2017). DOI 10.1145/3064834. URL http://doi.acm.org/10.1145/3064834
18. Jones, N.C.: Logic synthesis for fault-tolerant quantum computers. arXiv preprint arXiv:1310.7290 (2013)
19. Kassal, I., Jordan, S.P., Love, P.J., Mohseni, M., Aspuru-Guzik, A.: Polynomial-time quantum algorithm for the simulation of chemical dynamics. Proceedings of the National Academy of Sciences **105**(48), 18681–18686 (2008)
20. Kliuchnikov, V., Bocharov, A., Roetteler, M., Yard, J.: A framework for approximating qubit unitaries. arXiv preprint arXiv:1510.03888 (2015)
21. Kliuchnikov, V., Maslov, D., Mosca, M.: Asymptotically optimal approximation of single qubit unitaries by Clifford and T circuits using a constant number of ancillary qubits. Physical review letters **110**(19), 190502 (2013)
22. Kliuchnikov, V., Maslov, D., Mosca, M.: Fast and efficient exact synthesis of single-qubit unitaries generated by Clifford and T gates. Quantum Information & Computation **13**(7-8), 607–630 (2013)
23. Lin, C., Chakrabarti, A., Jha, N.K.: FTQLS: fault-tolerant quantum logic synthesis. IEEE Trans. VLSI Syst. **22**(6), 1350–1363 (2014). DOI 10.1109/TVLSI.2013.2269869. URL https://doi.org/10.1109/TVLSI.2013.2269869
24. Mermin, N.D.: Quantum Computer Science: An Introduction. Cambridge University Press (2007)
25. Miller, D.M., Wille, R., Sasanian, Z.: Elementary quantum gate realizations for multiple-control Toffolli gates. In: Int'l Symp. on Multi-Valued Logic, pp. 288–293 (2011)
26. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge Univ. Press (2000)
27. Niemann, P., Wille, R., Drechsler, R.: Efficient synthesis of quantum circuits implementing Clifford group operations. In: ASP Design Automation Conf., pp. 483–488 (2014)
28. Niemann, P., Wille, R., Drechsler, R.: Equivalence checking in multi-level quantum systems. In: Reversible Computation, pp. 201–215 (2014)
29. Niemann, P., Wille, R., Drechsler, R.: Improved synthesis of Clifford+T quantum functionality. In: Design, Automation and Test in Europe, pp. 597–600 (2018)
30. Niemann, P., Wille, R., Miller, D.M., Thornton, M.A., Drechsler, R.: QMDDs: Efficient quantum function representation and manipulation. IEEE Trans. on CAD **35**(1), 86–99 (2016). DOI 10.1109/TCAD.2015.2459034
31. Russell, T.: The exact synthesis of 1- and 2-qubit Clifford+T circuits. ArXiv e-prints (2014)
32. Saeedi, M., Arabzadeh, M., Zamani, M.S., Sedighi, M.: Block-based quantum-logic synthesis. Quantum Information & Computation **11**(3&4), 262–277 (2011)
33. Saeedi, M., Zamani, M.S., Sedighi, M., Sasanian, Z.: Synthesis of reversible circuit using cycle-based approach. J. Emerg. Technol. Comput. Syst. **6**(4) (2010)
34. Shende, V.V., Bullock, S.S., Markov, I.L.: Synthesis of quantum-logic circuits. IEEE Trans. on CAD **25**(6), 1000–1010 (2006)
35. Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Synthesis of reversible logic circuits. IEEE Trans. on CAD **22**(6), 710–722 (2003)

36. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. Foundations of Computer Science pp. 124–134 (1994)
37. Soeken, M., Wille, R., Hilken, C., Przigoda, N., Drechsler, R.: Synthesis of reversible circuits with minimal lines for large functions. In: ASP Design Automation Conf., pp. 85–92 (2012)
38. Welch, J., Bocharov, A., Svore, K.M.: Efficient approximation of diagonal unitaries over the Clifford+T basis. Quantum Information & Computation **16**(1&2), 87–104 (2016). URL http://www.rintonpress.com/xxqic16/qic-16-12/0087-0104.pdf
39. Wille, R., Große, D., Miller, D.M., Drechsler, R.: Equivalence checking of reversible circuits. In: Int'l Symp. on Multi-Valued Logic, pp. 324–330 (2009)
40. Wille, R., Große, D., Teuber, L., Dueck, G.W., Drechsler, R.: RevLib: an online resource for reversible functions and reversible circuits. In: Int'l Symp. on Multi-Valued Logic, pp. 220–225 (2008). RevLib is available at http://www.revlib.org
41. Wille, R., Soeken, M., Otterstedt, C., Drechsler, R.: Improving the mapping of reversible circuits to quantum circuits using multiple target lines. In: ASP Design Automation Conf., pp. 85–92 (2013)
42. Yamashita, S., Markov, I.L.: Fast equivalence - checking for quantum circuits. Quantum Information & Computation **10**(9&10), 721–734 (2010). URL http://www.rintonpress.com/xxqic10/qic-10-910/0721-0734.pdf
43. Zulehner, A., Wille, R.: Improving synthesis of reversible circuits: Exploiting redundancies in paths and nodes of QMDDs. In: Reversible Computation, pp. 232–247 (2017)
44. Zulehner, A., Wille, R.: One-pass design of reversible circuits: Combining embedding and synthesis for reversible logic. IEEE Trans. on CAD (2017)
45. Zulehner, A., Wille, R.: Advanced simulation of quantum computations. IEEE Trans. on CAD **38**(5), 848–859 (2019)
46. Zulehner, A., Wille, R.: Compiling SU(4) quantum circuits to IBM QX architectures. In: ASP Design Automation Conf., pp. 185–190 (2019)
47. Zulehner, A., Wille, R.: Matrix-vector vs. matrix-matrix multiplication: Potential in DD-based simulation of quantum computations. In: Design, Automation and Test in Europe, pp. 90–95 (2019)