SAT-based {CNOT, T} quantum circuit synthesis

Giulia Meuli, Mathias Soeken, and Giovanni De Micheli

École polytechnique fédérale de Lausanne (EPFL), Lausanne, Switzerland

Abstract. The prospective of practical quantum computers has lead researchers to investigate automatic tools to program them. A quantum program is modeled as a Clifford+T quantum circuit that needs to be optimized in order to comply with quantum technology constraints. Most of the optimization algorithms aim at reducing the number of T gates. Nevertheless, a secondary optimization objective should be to minimize the number of two-qubit operations (the CNOT gates) as they show lower fidelity and higher error rate when compared to single-qubit operations. We have developed an exact SAT-based algorithm for quantum circuit rewriting that aims at reducing CNOT gates without increasing the number of T gates. Our algorithm finds the minimum {CNOT, T} circuit for a given phase polynomial description of a unitary transformation. Experiments confirm a reduction of CNOT in T-optimized quantum circuits. We synthesize quantum circuits for all single-target gates whose control functions are one of the representatives of the 48 spectral equivalence classes of all 5-input Boolean functions. Our experiments show an average CNOT reduction of 26.84%.

Keywords: Quantum computing; Clifford+T circuits; SAT-based rewriting algorithm.

1 Introduction

There is a worldwide effort in building the first practical quantum computer and many companies are betting on different technologies, as in this field research on physical devices is moving from academia to companies [5]. Microsoft, Google [13], IBM [11, 14], Intel [12], as well as the rapidly growing startup companies IonQ and Rigetti, are investing significant effort into building the first scalable quantum computer. They all seek quantum supremacy: solving for the first time a problem that cannot be solved classically [4, 9]. Superposition and entanglement are the unique physical properties that provide quantum computers with such potential ability.

– *Superposition:* Qubits do not only have the two classical states $|0\rangle, |1\rangle$ but can be in any superposition of these states $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ with $|\alpha|^2 + |\beta|^2 = 1$. Measurements destroy the superposition forcing the state to collapse into one of the classical states, according to the relative probabilities $|\alpha|^2$ and $|\beta|^2$. This property leads to a high parallelism that can be exploited for computation.

– *Entanglement:* Given two qubits, entanglement is a global property different from the product of their two states and that cannot be accounted classically [10]. Due to this property two qubits in superposition can be correlated with one another, this means that the state of one depends on the state of the other even when placed at large distances.

In addition, accounting for the computational power of quantum systems there is the ability of representing an exponentially larger number of states compared to classical computers, i.e., an n-bit classical computer can represent one out of 2^n classical states while n qubits can represent 2^n of these classical states at the same time. Adding one qubit to the system doubles its performances and its computing capabilities.

Universal libraries of instructions are used to program quantum computers. Instructions that are performed on qubits are also referred to as quantum gates, with a qubit state as the gate's input and output. In this analogy a quantum circuit can be interpreted as a program of quantum instructions.

In this work we target the Clifford+T universal quantum library, composed of the set of Clifford gates (Hadamard, S, and CNOT gates) and of the non-Clifford T gate. In this library the T gate has proven to be the most expensive to implement in fault tolerant circuits [2]. This is the reason why research often focuses on minimizing the number of T gates [1, 18, 23]. On the other hand, the CNOT gate, also included in the library, is the hardest to perform on the physical level because it requires to establish an interaction between two qubits [19]. It has been shown how an increasing number of CNOT gates reduces the probability of getting a distinguishable result [16]. While there are some methods to efficiently synthesize CNOT circuits [20, 21] we propose a SAT-based algorithm to synthesize {CNOT, T} circuits from a phase polynomial representation with the minimum number of CNOT gates and without increasing the T-count, i.e., the number of T gates in the quantum circuit. We show how this synthesis method can be used to rewrite T-optimized [1] Clifford+T circuits achieving an average reduction of 26.84% in the number of CNOT.

2 Preliminaries

2.1 Boolean functions

We call a function $f : \mathbb{B}^n \to \mathbb{B}$ a Boolean function over n variables where $\mathbb{B} = \{0, 1\}$ is the two-element finite field with addition corresponding to Boolean exclusive-OR and multiplication corresponding to Boolean AND. A Boolean function can be represented by its truth table which is a bitstring $b_{2^n-1}b_{2^n-2}\dots b_0$ of size 2^n where

$$b_x = f(x_1, \dots, x_n) \qquad \text{when } x = (x_1x_2\dots x_n)_2.$$

For large functions it is convenient to use a hexadecimal encoding of the bitstring.

Example 1. The truth table of the majority-of-three function $\langle x_1 x_2 x_3 \rangle$ is 1110 1000 or `0xe8` in hexadecimal encoding.

Definition 1. *A Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$ is* reversible *if and only if f is a bijection, i.e., $n = m$ and it performs a permutation of the set of input patterns.*

Definition 2. *A Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ is* linear *if and only if:*

$$f(x_1 \oplus x_2) = f(x_1) \oplus f(x_2).$$

Any linear Boolean function can be written as

$$f(x_1, \dots, x_n) = a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n \tag{1}$$

for constants $a_1, \dots, a_n \in \mathbb{B}$. Given this notation we can write any linear function f as a row vector of the n constant Boolean coefficients: $(a_1 \dots a_n)$.

Example 2. Given the linear Boolean function $f(x_1, x_2, x_3) = x_2 \oplus x_3$ it corresponds to the row vector: (0 1 1).

Definition 3. *A multi-output Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$ is* linear *if and only if each component function f_i is linear, for $i = 1, \dots, m$.*

A multi-output linear Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$ can be represented using an $m \times n$ matrix, in which each row is the row vector representing a component linear function f_i. If the multi-output function is linear and reversible the representative matrix is a non-singular matrix $n \times n$.

Example 3. The controlled-NOT gate (see Fig. 1(a)), implementing the function

$$\text{CNOT} : |x_1\rangle|x_2\rangle \mapsto |x_1\rangle|x_1 \oplus x_2\rangle,$$

is both linear and reversible, while the Toffoli gate implements a function (see Fig. 1(b)) that is reversible but not linear:

$$\text{Tof} : |x_1\rangle|x_2\rangle|x_3\rangle \mapsto |x_1\rangle|x_2\rangle|x_3 \oplus (x_1 \wedge x_2)\rangle.$$

By only using CNOT gates it is possible to build a linear reversible circuit with n inputs, implementing a multi-output reversible linear function $f : \mathbb{B}^n \to \mathbb{B}^n$, with n linear reversible Boolean functions as components f_i.

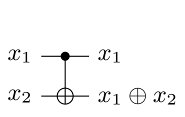
Example 4. The linear reversible circuit shown in Fig. 2 computes four different linear functions $f_i : \mathbb{B}^4 \to \mathbb{B}$:

$$f_1 = x_2 \oplus x_3, f_2 = x_1 \oplus x_2 \oplus x_3, f_3 = x_3, f_4 = x_3 \oplus x_4.$$

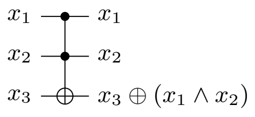
If we represent them using row vectors (see Ex. 2) of their Boolean coefficients and group such vectors in an $n \times n$ matrix, we obtain a matrix G representing a multi-output linear reversible Boolean function:

$$G = \begin{pmatrix} 0\,1\,1\,0 \\ 1\,1\,1\,0 \\ 0\,0\,1\,0 \\ 0\,0\,1\,1 \end{pmatrix}.$$

We will use this representation in the encoding of our SAT problem.



(a) CNOT.



(b) Toffoli.

Fig. 1. Examples of two gates implementing reversible functions.

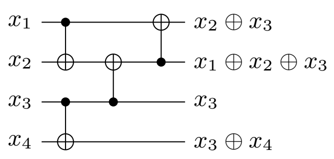


Fig. 2. Example of a linear reversible CNOT circuit.

2.2 Clifford+T universal quantum library

Quantum circuits are described in terms of a small library of gates that interact with one or two qubits. The most frequently considered universal library is the so-called Clifford+T gate library that consists of two single-qubit operations (the Hadamard gate, abbreviated H, and the T gate) and one two-qubit operation (the reversible CNOT gate).

All single-qubit operations can be represented using a 2×2 matrix U. The only condition for such matrix to be a valid quantum operation is to be unitary, that means $U^\dagger U = UU^\dagger = I$ where $U^\dagger$ is the complex conjugate transpose of U. This condition guarantees that the quantum state resulting from the operation will have $|\alpha|^2 + |\beta|^2 = 1$.

- *X gate:* The X gate is the quantum equivalent to the classical NOT gate, that complements the state of a classical bit. Given as input $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$, it swaps the amplitudes and returns $|\phi\rangle = \beta|0\rangle + \alpha|1\rangle$.
- *H gate:* The Hadamard gate is often used to create superposition. It transforms a $|0\rangle$ into a $(|0\rangle + |1\rangle)/\sqrt{2}$ and a $|1\rangle$ into a $(|0\rangle - |1\rangle)/\sqrt{2}$. The resulting state is halfway between $|0\rangle$ and $|1\rangle$ and collapses into one of these classical states with 50% probability. For example, given n qubits initialized to $|0\rangle$, if a H gate is applied to each of them we have:

$$\frac{1}{\sqrt[n]{2}}(|00\dots0\rangle + |00\dots1\rangle + \cdots + |01\dots1\rangle + |11\dots1\rangle)$$

  This means that the system is receiving all the possible input combinations at the same time, with the same probability $1/2^{2/n}$.
- *T gate:* This gate does not belong to the Clifford library and it is necessary to achieve universality, that means adding this gate to the Clifford ones makes it possible to approximate any unitary matrix arbitrarily precise. The

T gate is sufficiently expensive in most approaches to fault tolerant quantum computing [2].

The corresponding unitary matrices for these gates are:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, T = \begin{pmatrix} 1 & 0 \\ 0 & e^{\mathrm{i}\pi/4} \end{pmatrix}.$$

The controlled-NOT (or CNOT) operation is the only two qubits operation in the library and for this reason is the most complex to implement on the physical level. It complements the state of one qubit called *target* accordingly to the state of the other qubit called *control*. We can write $|00\rangle \mapsto |00\rangle, |01\rangle \mapsto |01\rangle, |10\rangle \mapsto |11\rangle, |11\rangle \mapsto |10\rangle$, the corresponding matrix is:

$$U_{\mathrm{CNOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

2.3 Phase polynomial representation

A {CNOT, T} n-qubit quantum circuit, i.e., a circuit composed only of CNOT and T gates, implements a unitary matrix U that can be expressed using a linear reversible function g and a polynomial $p(x_1, \ldots, x_n)$ defining a diagonal phase transformation. This circuit description is called *phase polynomial representation* [2] and more than one {CNOT, T} circuit can share the same representation.

Lemma 1. *The action of a {*CNOT*, T} circuit on the initial states* $|x_1, \ldots, x_n\rangle$ *has the form:*

$$|x_1, \ldots, x_n\rangle \mapsto e^{\frac{\pi}{4}\mathrm{i}p(x_1,\ldots,x_n)}|g(x_1, \ldots, x_n)\rangle,$$

$$\textit{with } p(x_1, \ldots, x_n) = \sum_{i=1}^{l} (c_i \bmod 8) f_i(x_1, \ldots, x_n), \quad (2)$$

where $g : \mathbb{B}^n \to \mathbb{B}^n$ *is a linear reversible function,* p *is a linear combination of linear Boolean functions* $f_i : \mathbb{B}^n \to \mathbb{B}$ *with the coefficients reduced modulo 8. The coefficients* $c_i \in \mathbb{Z}$ *measure the number of rotations of* $\pi/4$ *that are applied to the corresponding* f_i*, e.g., each* T *gate gives a* $\pi/4$ *rotation (*$c_i = 1$*), an* S *gate gives a* $\pi/2$ *rotation (*$c_i = 2$*), a* $T^\dagger$ *gate gives a* $7\pi/4$ *rotation (*$c_i = 7$*). The phase polynomial notation is uniquely specified by:*

$$g, f_i, c_i \textit{ for } i = 1, \ldots, l$$

where l *is the number of phase gates.*

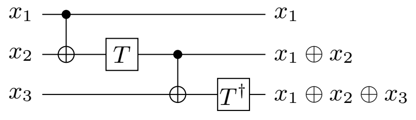


Fig. 3. Example of a {CNOT, T} circuit.

Example 5. The circuit in Fig. 3 implements a transformation on the input qubit states x_1, x_2, x_3 characterized by the linear reversible function

$$g : |x_1\rangle|x_2\rangle|x_3\rangle \mapsto |x_1\rangle|x_1 \oplus x_2\rangle|x_1 \oplus x_2 \oplus x_3\rangle$$

and by a phase polynomial

$$p(x_1, x_2, x_3) = 1(x_1 \oplus x_2) + 7(x_1 \oplus x_2 \oplus x_3)$$
$$\text{with } c_1 = 1, c_2 = 7, f_1 = x_1 \oplus x_2, f_2 = x_1 \oplus x_2 \oplus x_3.$$

Where the T gate gives a phase of $\pi/4$ while its conjugate complex $T^\dagger$ gives a phase of $7\pi/4$.

Given a {CNOT, T} circuit, we can extract its phase polynomial representation. Our proposed algorithm, taking this representation as input, finds the corresponding {CNOT, T} quantum circuit with the minimum number of CNOT, solving iteratively a satisfiability problem.

2.4 Boolean satisfiability

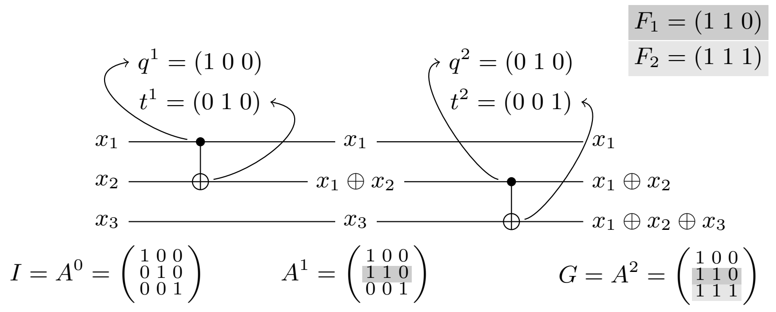
Given a Boolean function $f(x_1, \ldots, x_n)$, the Boolean satisfiability problem (or SAT problem) consists of determining an assignment V to the variables $x_1, \ldots, x_n$ such that f is satisfied (evaluates to true). If such an assignment exists we call it a satisfying assignment, otherwise the problem is unsatisfiable.

SAT solvers [3,15] are software programs in which f is specified as conjunctive normal form (CNF) consisting of a conjunction of clauses where each clause is a disjunction of literals. We define a literal as instance of a variable or its complement. SAT can be summarized as follows: given a list of clauses, determine if there exists a variable assignment that satisfies all of them.

3 SAT-based algorithm for CNOT reduction

Problem 1. Given a phase polynomial description of a unitary transformation (g, f_i, c_i,) and an integer K, determine if there exists a {CNOT, T} quantum circuit implementing it with K CNOT gates. We denote an instance of this problem *HasCNOT(g, f_i, c_i, K).*

We represent the linear reversible function g using a $n \times n$ matrix G with entries $G_{i,j}$. At the same time we define F_i as the row vector representation of f_i with entries $F_{i,j}$.



$F_1 = (1\ 1\ 0)$
$F_2 = (1\ 1\ 1)$
$q^1 = (1\ 0\ 0)$
$t^1 = (0\ 1\ 0)$
$q^2 = (0\ 1\ 0)$
$t^2 = (0\ 0\ 1)$
x_1
x_2
x_3
$x_1 \oplus x_2$
$x_1 \oplus x_2 \oplus x_3$
$I = A^0 = \begin{pmatrix}1\,0\,0\\0\,1\,0\\0\,0\,1\end{pmatrix}$
$A^1 = \begin{pmatrix}1\,0\,0\\1\,1\,0\\0\,0\,1\end{pmatrix}$
$G = A^2 = \begin{pmatrix}1\,0\,0\\1\,1\,0\\1\,1\,1\end{pmatrix}$

Fig. 4. Illustration of SAT encoding for sample circuit in Fig. 3.

Example 6. The phase polynomial representation shown in Ex. 5 is represented by:

$$G = \begin{pmatrix}1\,0\,0\\1\,1\,0\\1\,1\,1\end{pmatrix}, \{c_1 = 1, F_1 = (1\ 1\ 0)\}, \{c_2 = 7, F_2 = (1\ 1\ 1)\}$$

3.1 Encoding

Example 7. In Fig. 4 we show an example of how the problem of synthesizing a circuit for the phase polynomial representation in Ex. 5 with two CNOT gates is encoded.

If the specified transformation is performed using K CNOT gates, there must be K gate transformations that map $A^{k-1} \to A^k$, for $1 \leq k \leq K$, where A^0 is the identity matrix satisfying:

$$A^K = G \text{ and } \forall_j \exists A_i^k.(A_i^k = F_j) \tag{3}$$

The latter means that at least one row A_i^k is equal to the specified linear Boolean functions.

Example 8. In the case of our example:

$$A^0 = \begin{pmatrix}1\,0\,0\\0\,1\,0\\0\,0\,1\end{pmatrix}, A^1 = \begin{pmatrix}1\,0\,0\\1\,1\,0\\0\,0\,1\end{pmatrix}, A^2 = \begin{pmatrix}1\,0\,0\\1\,1\,0\\1\,1\,1\end{pmatrix} = G, \text{ and}$$
$$A_1^1 = A_1^2 = (1\ 1\ 0) = F_1, A_2^2 = (1\ 1\ 1) = F_2$$

Each CNOT gate is represented by two vectors: $q^k = (q_1^k \ldots q_n^k)$ describing the gate control and $t^k = (t_1^k \ldots t_n^k)$ for the target, where $q_i^k(t_i^k) = 1$ only if the i^{th} variable is a control (target) of the gate (see Fig. 4).

First we need to ensure that those variables are describing valid CNOT gates, characterized by one control and one target variable. We define the following one-hot clauses:

$$\forall_{1\leq k\leq K}[(q_1^k \vee \dots \vee q_n^k) \wedge \bigwedge_{1\leq i<j\leq n} (\bar{q}_i^k \vee \bar{q}_j^k)] \tag{4}$$

and

$$\forall_{1\leq k\leq K}[(t_1^k \vee \dots \vee t_n^k) \wedge \bigwedge_{1\leq i<j\leq n} (\bar{t}_i^k \vee \bar{t}_j^k)] \tag{5}$$

In addition, the control and target of each gate need to be acting on different variables to represent a valid CNOT:

$$\forall_{1\leq k\leq K} \bigwedge_{i=1}^{n} (q_i^k \neq t_i^k) \tag{6}$$

After we have defined the matrices $A_0, \dots, A_K$ (3) and given conditions over the gate vectors q^k and t^k describing the mapping $A^{k-1} \mapsto A^k$ (4),(5),(6), we need to encode the functionality of this mapping. In order to do so we define some intermediate expressions h_j^k for each row of a matrix A^{k-1}, defining whether it intersects with the control vector q^k

$$\forall_{1\leq k\leq K}, \forall_{j=1}^{n} h_j^k = \bigoplus_{i=1}^{n} A_{ij}^{k-1} \wedge q_i^k$$

By means of such variables we define in which cases the application of a CNOT gate causes elements of the matrix A to switch.

$$\forall_{1\leq k\leq K}, \forall_{i=1}^{n}, \forall_{j=1}^{n} A_{i,j}^k = A_{i,j}^{k-1} \oplus (t_i^k \wedge h_j^k) \tag{7}$$

Example 9. Consider the first CNOT gate in Fig 4, it can be represented by the vectors:

$$q^1 = (1\ 0\ 0) \text{ and } t^1 = (0\ 1\ 0)$$

verifying conditions (4),(5),(6). We can compute the intermediate variables h_j^1 checking whether one or more elements in the matrix A^0 should switch.

$$h_1^1 = A_{1,1}^0 \wedge q_1^1 \oplus A_{2,1}^0 \wedge q_2^1 \oplus A_{3,1}^0 \wedge q_3^1 = 1$$
$$h_2^1 = A_{1,2}^0 \wedge q_1^1 \oplus A_{2,2}^0 \wedge q_2^1 \oplus A_{3,2}^0 \wedge q_3^1 = 0$$
$$h_3^1 = A_{1,3}^0 \wedge q_1^1 \oplus A_{2,3}^0 \wedge q_2^1 \oplus A_{3,3}^0 \wedge q_3^1 = 0$$

Then the only element in the matrix A that switches is $A_{2,1}^1 = A_{2,1}^0 \oplus (t_2^1 \wedge h_1^1)$ and:

$$A^0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mapsto A^1 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3.2 SAT problem: Summary

The specifications are given to our problem as: a matrix G representing a linear reversible Boolean function g, the set of coefficients c_i and the linear Boolean functions f_i.

SAT variables. For each gate we define: (i) control variables q_i^k, defining which variable i is the control of gate k, (ii) target variables t_i^k, defining which variable i is the target of gate k.

In order to evaluate the evolution of the implemented function gate by gate, we define matrices representing intermediate synthesized linear transformations: A^k for $1 \leq k \leq K$. We also define some auxiliary variables h_j^k that are used to define the mapping $A^{k-1} \mapsto A^k$

SAT clauses. The following clauses define our SAT problem:

- Initial clauses: $A^0 = I$ where I is the identity matrix.
- Final clauses: $A^K = G$ and $\forall_j \exists A_i^k.(A_i^k = F_j)$. The final linear transformation implements the desired G and all the functions f_i are present at some position in the circuit so that c_i $\pi/4$ phase shifts can be applied.
- Validity clauses: only one variable q_i^k (t_i^k) is equal to one for each gate k and $q^k \neq t^k$. We consent only one control and one target defined on different variables.
- Dependency clauses: $A_{i,j}^k = A_{i,j}^{k-1} \oplus (t_i^k \wedge h_j^k)$ defining the relation between the gate variables q_i^k, t_i^k with respect to the intermediate matrix variables. This clause defines the mapping $A^{k-1} \mapsto A^k$. If we consider all the K gates, we will have $I = A^0 \mapsto A^1 \mapsto \cdots \mapsto A^{K-1} \mapsto A^K = G$.

3.3 SAT-based rewriting algorithm

Our synthesis algorithm aims at minimizing the number of CNOT gates without increasing the number of T gates. For this reason the starting point is a Clifford+T circuit with an optimized number of T gates, obtained using the optimization algorithm T-par [1]. The overall procedure is described by Alg. 1.

The first step of the algorithm is to extract from the input circuit, built on the whole range of Clifford+T gates, some {CNOT, T} subcircuits. This partition is required because we aim at minimizing all possible quantum circuits, hence defined over the Clifford+T library, using an exact method for reducing {CNOT, T} circuits. The extraction method *ct_ extract* is performed in such a way that after the synthesis performed by the SAT-solver, the sub-circuits can be recombined to restore the initial functionality.

The next step is to re-synthesize each subcircuit. In order to input a subcircuit in our SAT problem we first need to extract the phase polynomial representation (g, f_i, c_i), this is performed by the procedure *phase*. The SAT

Algorithm 1 SAT-based rewriting algorithm

```
1: $c \leftarrow$ T-optimized quantum circuit
2: ct_extract($c$)   *(extract {CNOT, T} sub-circuits)*
3: for each sub-circuit $c'$ do
4:     $(g, f_i, c_i) \leftarrow$phase($c'$)   *(set the phase polynomial representation from $c'$)*
5:     $K \leftarrow 0$   *(number of gates)*
6:     repeat
7:         Solve(HasCNOT($g, f_i, c_i, K$))
8:         if SAT then
9:             $c'' \leftarrow$Extract {CNOT, $T$} circuit
10:            Replace $c'$ by $c''$ in $c$
11:        else UNSAT
12:            $K \leftarrow K + 1$
13:    until a solution is found
```

solver is used iteratively to solve our encoded problem *HasCNOT*. It tries to find a satisfying solution with K gates and adds an additional gate if the problem is unsatisfiable. This procedure is repeated until a valid solution is found. This solution will describe a circuit with the minimum number of CNOT gates, given the polynomial representation.

4 Results

We have implemented Algorithm 1 in C++ on top of RevKit [22] and have used the Z3 prover [6] to encode and solve the SAT problem. As benchmarks we have used Clifford+T circuits for 48 single-target gates. A single-target gate is a generalized Toffoli gate, which instead of 2 control lines allows any set of k control lines and a k-input Boolean control function. A single-target gate inverts its target line if and only if the control function evaluates to true on the current control line assignment. RevKit contains a database of optimized Clifford+T circuits [17] for single-target gates where the control function is a representative of one of the 48 spectral equivalent classes for 5-input Boolean functions [8]. Several reversible logic synthesis algorithms such as Young subgroup based synthesis [7] and LUT-based hierarchical reversible logic synthesis [24] use single-target gates as intermediate representation, which can be directly mapped to their pre-computed optimized Clifford+T implementation, if it does not have more than 6 control lines. Consequently, improving on these implementations has a large positive impact on the results of these synthesis methods.

While in [17] the database circuits are optimal only in the number of T gates, our proposed optimization algorithm can reduce the number of CNOT gates by keeping the T-count unchanged. Results are shown in Table 4, we report for each equivalent class: (i) the hexadecimal encoding of the representative function, (ii) the number of T gates in the initial circuit, (iii) the number of CNOT in the initial circuit, (iv) the reduced number of CNOT in the circuit synthesized using our method, (v) the percentage reduction and (vi) the runtime in seconds. We also report the average percentage reduction and the maximum measured reduction, 26.84% and 45.45%, respectively. The method used to decompose the initial Clifford+T circuit into {CNOT, T} subcircuit, namely *ct_ extract* impacts the final optimized results and the runtime. As it was expected, the presence of larger subcircuits leaves larger space for optimization, but slows down the operation of the SAT solver.

5 Conclusion

In this work we propose a SAT-based synthesis method to build {CNOT, T} circuits with the minimum number of CNOT gates for a given phase polynomial representation. Most optimized synthesis methods in the literature aim at reducing the number of T gates, which are expensive in fault tolerant circuits. Nevertheless, since CNOT gates are difficult to implement from the physical point of view, relying on a delicate interaction between two qubits, we believe that methods for CNOT reduction should also be investigated. While there exist algorithms dealing with linear reversible CNOT circuits, our method is the first to our knowledge being capable of optimizing the number of CNOT gates in Clifford+T circuits, without changing the circuit functionality. We have validated our method showing an average 26.84% CNOT reduction synthesizing all the 48 functions representative of the 5-input equivalent classes. In future works we plan to extend our algorithm to support the optimization of the more general class of circuits that are composed of X gates, CNOT gates and phase gates with continuous rotation angles [19].

References

1. Amy, M., Maslov, D., Mosca, M.: Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning. IEEE Trans. on CAD of Integrated Circuits and Systems **33**(10), 1476–1489 (2014)
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 of Integrated Circuits and Systems **32**(6), 818–830 (2013)
3. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. IOS Press (2009)
4. Boixo, S., Isakov, S.V., Smelyanskiy, V.N., Babbush, R., Ding, N., Jiang, Z., Bremner, M.J., Martinis, J.M., Neven, H.: Characterizing quantum supremacy in near-term devices. arXiv preprint arXiv:1608.00263v3 (2017)
5. Castelvecchi, D.: Quantum computers ready to leap out of the lab in 2017. Nature News **541**(7635), 9 (2017)
6. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008)
7. De Vos, A., Van Rentergem, Y.: Young subgroups for reversible computers. Advances in Mathematics of Communications **2**(2), 183–200 (2008)

Table 1. Results

	Class	T-count	CNOT initial	CNOT final	%CNOT	Runtime
0	0x00000000	0	0	0	0.00	0.00
1	0x80000000	31	61	55	9.84	137.24
2	0x80008000	24	45	38	15.56	8.68
3	0x00808080	51	112	97	13.39	24.93
4	0x80808080	16	47	30	36.17	7.82
5	0x88800800	48	94	79	15.96	13.81
6	0x88088020	75	175	143	18.29	574.99
7	0x88808080	47	100	79	21.00	133.77
8	0x2a808080	32	89	54	39.33	202.16
9	0x70080088	56	127	96	24.41	814.43
10	0xf3c0dd00	48	118	79	33.05	257.79
11	0xc0c8c0c8	29	65	54	16.92	9.46
12	0x734470c8	111	221	176	20.36	2148.00
13	0xe0a0c000	63	156	111	28.85	135.59
14	0xe8080808	71	178	128	28.09	293.18
15	0x8808a808	63	136	102	25.00	17.44
16	0x08888888	36	82	73	10.98	445.86
17	0x88888888	7	11	6	45.45	0.92
18	0xd5808080	32	89	58	34.83	215.22
19	0x70807080	15	40	23	42.50	3.08
20	0xe1808880	88	194	130	32.99	16.48
21	0xea808080	56	140	84	40.00	10.14
22	0xcc808880	55	117	88	24.79	10.53
23	0xe4404440	55	118	83	29.66	5.71
24	0x7f008000	23	39	32	17.95	4.22
25	0xe0a8c800	91	219	161	26.48	821.58
26	0xe8818880	115	291	210	27.84	832.18
27	0xe8a08880	80	195	168	13.85	368.34
28	0xf8808880	80	182	143	21.43	57.50
29	0xe2222220	56	123	85	30.89	1771.19
30	0xa0c8a088	63	192	149	22.40	385.03
31	0xe6804c80	39	96	58	39.58	5.94
32	0x7f808080	19	60	44	26.67	388.19
33	0x0231da51	79	192	138	28.12	8.77
34	0xa008bc88	95	204	151	25.98	50.76
35	0xd8887888	43	112	70	37.50	360.20
36	0xeca08088	80	180	118	34.44	10.77
37	0xf0888888	56	144	97	32.64	860.57
38	0x8a80cac0	47	108	82	24.07	523.50
39	0x78807880	36	74	56	24.32	1668.62
40	0xbca08488	79	208	154	25.96	863.73
41	0xfca08880	96	225	153	32.00	53.71
42	0xdac08a80	76	190	157	17.37	356.73
43	0xf8887888	43	107	91	14.95	363.48
44	0x78887888	12	24	15	37.50	869.91
45	0xa6cc60a0	47	126	83	34.13	17.35
46	0x62c8ea40	35	95	63	33.68	25.30
47	0x6ac8e240	48	107	81	24.30	11.43

Average achieved CNOT minimization: 26.84%
Max achieved CNOT minimization: 45.45%

8. Edwards, C.R.: The application of the Rademacher-Walsh transform to Boolean function classification and threshold logic synthesis. IEEE Trans. on Computers **24**(1), 48–62 (1975)
9. Harrow, A.W., Montanaro, A.: Quantum computational supremacy. Nature **549**(7671), 203–209 (2017)
10. Hill, S., Wootters, W.K.: Entanglement of a pair of quantum bits. Physical Review Letters **78**(26), 5022 (1997)
11. IBM: IBM builds its most powerful universal quantum computing processors (2017), press release by IBM, posted online May 17, 2017
12. Intel: Intel delivers 17-qubit superconducting chip with advanced packaging to QuTech (2017), press release by Intel, posted online October 10, 2017
13. Kelly, J.: A preview of Bristlecone, Google's new quantum processor. Google Research Blog (2018)
14. Knight, W.: IBM rasises the bar with a 50-qubit quantum computer. Sighted at MIT Review Technology: https://www. technologyreview. com/s/609451/ibm-raises-the-bar-with-a-50-qubit-quantum-computer (2017)
15. Knuth, D.E.: The Art of Computer Programming, Volume 3, Second Edition. Addison-Wesley (1998)
16. Linke, N.M., Maslov, D., Roetteler, M., Debnath, S., Figgatt, C., Landsman, K.A., Wright, K.E., Monroe, C.: Experimental comparison of two quantum computing architectures. Proceedings of the National Academy of Sciences **114**(13), 3305–3310 (2017)
17. Meuli, G., Soeken, M., Roetteler, M., Miller, D.M., Amy, M., Wiebe, N., De Micheli, G.: Estimating single-target gate T-count using spectral classification. Int'l Workshop on Logic and Synthesis (2018)
18. Meuli, G., Soeken, M., Roetteler, M., Wiebe, N., De Micheli, G.: A best-fit mapping algorithm to facilitate ESOP-decomposition in Clifford+T quantum network synthesis. In: Proceedings of the 23rd Asia and South Pacific Design Automation Conference. pp. 664–669. IEEE Press (2018)
19. Nam, Y.S., Ross, N.J., Su, Y., Childs, A.M., Maslov, D.: Automated optimization of large quantum circuits with continuous parameters. arXiv preprint arXiv:1710.07345 (2017)
20. Patel, K.N., Markov, I.L., Hayes, J.P.: Efficient synthesis of linear reversible circuits. arXiv preprint quant-ph/0302002 (2003)
21. Shende, V.V., Markov, I.L., Bullock, S.S.: Minimal universal two-qubit controlled-not-based circuits. Physical Review A **69**(6), 062321 (2004)
22. Soeken, M., Frehse, S., Wille, R., Drechsler, R.: RevKit: A toolkit for reversible circuit design. Multiple-Valued Logic and Soft Computing **18**(1), 55–65 (2012)
23. Soeken, M., Roetteler, M., Wiebe, N., De Micheli, G.: Design automation and design space exploration for quantum computers. In: Design, Automation and Test in Europe. pp. 470–475 (2017)
24. Soeken, M., Roetteler, M., Wiebe, N., De Micheli, G.: Hierarchical reversible logic synthesis using LUTs. In: Design Automation Conference. pp. 78:1–78:6 (2017)