Synthesis of Phase Oracles Using CNOT+T Circuits and ESOP Minimization

David Clarino
Ritsumeikan University
dclarino@fc.ritsumei.ac.jp

Chitranshu Arya
Netaji Subhas
University of Technology
chitranshu.arya.ug22@nsut.ac.in

Shigeru Yamashita
Ritsumeikan University
ger@cs.ritsumei.ac.jp

Zanhe Qi
Ritsumeikan University
goose@ngc.is.ritsumei.ac.jp

***Abstract*—Phase oracles are essential components of quantum algorithms such as Grover's algorithm. These circuits apply a phase of -1 to an input state when a particular Boolean function f is true. For functions of fewer than 3 variables, these are well known to be exactly implementable using only the CNOT, T, and T† gates. Amy [?] established that this class of circuits can be provably optimized for T-count and T-depth. We identify a class of Boolean functions of 4 or more variables which can be synthesized using such a circuit, by way of a library of circuits that implement phase oracles of 3 or fewer variables. We then use this to synthesize Boolean functions of 4 or more variables, by minimizing the number of ESOP terms of more than 3 variables and using a library of CNOT+T 3 variable circuits to synthesize cubes of 3 or fewer. This ensures as much of the circuit is implemented using only CNOT+T as possible. We compare the results against Qiskit's `PhaseOracle` method and find ...DCTODO**

I. Introduction

One of quantum computing's most significant results is the development of Grover's algorithm DCTODO. This algorithm performs an unstructured search in the theoretical lower bound of such a search DCTODO. Various applications have been proposed for such an algorithm, including the acceleration of solving NP-hard problems DCTODO.

Grover's algorithm uses a phase oracle, a quantum circuit that applies a phase of -1 when the Boolean function it implements is true, to increase the probability of returning a fulfilling assignment.

Phase oracles are interesting because a restricted class of them can be implemented using only phase gates such as the T, T†, and CNOT gates (hereafter referred to as CNOT+T). In fact, phase oracles of 3 or fewer variables can all be implemented exactly using this set of gates, and these circuits can be calculated exactly using the Boolean Fourier transform. Amy et al. [1] demonstrates that CNOT+T circuits can be optimized for both T-count and T-depth [2].

However, in general, it is impossible to implement Boolean functions of 4 or more variables using only this gate set [DCTODO]. There will inevitably be a need to use a Hadamard (H) gate. The addition of this gate makes optimal solutions non-unique, and several methods to optimize for T-count and T-depth when including the H gate are only heuristic [?], [1], [5].

Phase oracles are well known to be efficiently synthesizable from an Exclusive Sum of Products (ESOP) expression. Since each product maps to a Boolean function, it is easy to see that any such function expressible with product terms of at most 3 variables can be implemented as a CNOT+T circuit. We utilize this observation in our proposal.

In this work, we propose a solution that utilizes CNOT+T circuits to reduce the T-count and T-depth for general Boolean functions of arbitrary variable count.

Our contributions include:

- A precomputed library of circuits that implement all Boolean functions of 3 variables or fewer.
- A synthesis method that uses ESOP minimization to limit the number of product terms involving 4 or more variables.

We find that our solution... [DCTODO: complete with results]

The rest of this paper is structured as follows: First we introduce some preliminary knowledge. In addition to the basics of qubits and quantum circuits, this includes the basics of the mathematics of phase polynomials and their mapping to phase oracles and CNOT+T circuits. Then we detail our proposal, including how to generate a library of up to 3-qubit phase oracles, and how to use it to generate phase oracles of 4 or more qubits. Finally, we test our results against Qiskit PhaseOracle. Finally, we conclude with our findings and propose some future research.

II. Preliminaries

A. Quantum Bits and Quantum Gates

Quantum computers internally represent data as *qubits*, which are quantum systems that can take on the quantum states $|0\rangle$ and $|1\rangle$,

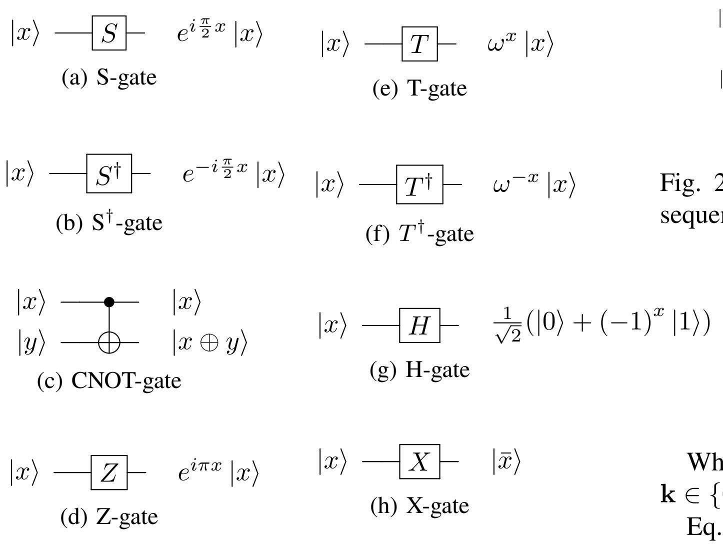
Additionally, qubits can also exist in quantum states that are a linear combination of $|0\rangle$ and $|1\rangle$, which is called *superposition*:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle, \alpha_0, \alpha_1 \in \mathbb{C} \tag{1}$$

Because these coefficients α_0 and α_b are complex numbers, they can also be expressed as a complex exponential $a \cdot e^{i\theta}$. θ in this exponential is often referred to as a *phase*.

Qubits can be taken together as tensor products to create multiqubit states.

$$|\psi\rangle = |\psi\rangle_0 \otimes |\psi\rangle_1 = (\alpha_{00} |0\rangle_0 + \alpha_{01} |1\rangle_0) \otimes (\alpha_{10} |0\rangle_1 + \alpha_{11} |1\rangle_1) \tag{2}$$


$|x\rangle$ S $e^{i\frac{\pi}{2}x}|x\rangle$
$|x\rangle$ T $\omega^{x}|x\rangle$
$|x\rangle$ S† $e^{-i\frac{\pi}{2}x}|x\rangle$
$|x\rangle$ T† $\omega^{-x}|x\rangle$
$|x\rangle$ $|x\rangle$
$|y\rangle$ $|x\oplus y\rangle$
$|x\rangle$ H $\frac{1}{\sqrt{2}}(|0\rangle+(-1)^{x}|1\rangle)$
$|x\rangle$ Z $e^{i\pi x}|x\rangle$
$|x\rangle$ X $|\bar{x}\rangle$

(a) S-gate

(e) T-gate

(b) S†-gate

(f) $T^{\dagger}$-gate

(c) CNOT-gate

(g) H-gate

(d) Z-gate

(h) X-gate

Fig. 1: Quantum gates

Note in Equation 2 that In particular, the tensor product of several basis states is known as a *computational basis state*, and is written as a bit string consisting of the values in the component tensor multiplicands.

$$|x_0\rangle\otimes|x_1\rangle\otimes\cdots\otimes|x_{n-1}\rangle = |x_0x_1\cdots x_{n-1}\rangle = |\mathbf{x}\rangle, \mathbf{x}\in\{0,1\}^n \tag{3}$$

These computational basis states can similarly be in linear superposition.

$$|\psi\rangle = \sum_{\mathbf{k}\in\{0,1\}^n}\alpha_{\mathbf{k}}|\mathbf{k}\rangle, \sum_{\mathbf{k}\in\{0,1\}^n}\alpha_{\mathbf{k}} = 1 \tag{4}$$

Quantum gates describe transformations in qubits. The main ones that concern this work are demonstrated in Fig. 1. The $T/T^{\dagger}$, $S/S^{\dagger}$, and Z gates will be referred to as *phase gates*.

All of these gates besides $T/T^{\dagger}$ are known as *Clifford gates*. For fault-tolerant quantum computers, Clifford gates are relatively inexpensive to implement. However, without $T/T^{\dagger}$-gates, computations that display quantum advantage cannot be implemented. Thus, *T-count*, the number of $T/T^{\dagger}$-gates, and *T-depth*, the longest such serial chain of them in a circuit, are often used as a cost metric [1].

All of the gates in Fig. 1 can in turn be composed into *quantum circuits*.

B. Phase Oracles and Phase Polynomials

A *phase oracle* is one such quantum circuit that implements a Boolean function as a quantum circuit, implementing a mapping of a computational basis state $|\mathbf{x}\rangle \mapsto e^{i\pi f(\mathbf{x})}|\mathbf{x}\rangle$, where $f(\mathbf{x})$ is a Boolean function $\{0,1\}^n \to \{0,1\}$. Phase oracles are key elements in algorithms such as Grover's Algorithm.

Previous research has demonstrated that some phase oracles can be implemented using only CNOT and phase gates (CNOT+T) [3]. When done in such a manner, $f(\mathbf{x})$ can be expressed as a *phase polynomial*, defined in Eq. 5.

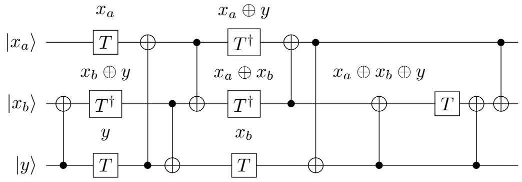


Fig. 2: Implementing $x_a\cdot x_b\cdot y$ using CNOT and T gates, sequenced using [2]

$$f(\mathbf{x}) = \sum_{\mathbf{k}\neq 0}\hat{f}(\mathbf{k})\cdot\chi_k(\mathbf{x}),$$
$$\hat{f}(\mathbf{k})\in\mathbb{R}$$

Where $\chi_k(\mathbf{x}) = (k_0x_0\oplus k_1x_1\oplus\cdots\oplus k_{n-1}x_{n-1})$ for some $\mathbf{k}\in\{0,1\}^n$.

Eq. 5 demonstrates one such phase polynomial. It can be verified from Table I that this is indeed true.

$$\begin{aligned} x_a\cdot x_b\cdot y = \frac{1}{4}x_a + \frac{1}{4}x_b + \frac{1}{4}y - \frac{1}{4}(x_a\oplus x_b) \\ -\frac{1}{4}(x_a\oplus y) - \frac{1}{4}(x_b\oplus y) + \frac{1}{4}(x_a\oplus x_b\oplus y) \end{aligned} \tag{5}$$

Such a phase polynomial can be implemented as a phase oracle using the following method.

- Coefficients of the phase polynomial describe the phase that needs to be driven as multiples of $i\pi$ (i.e. $3/4$ will be driven as an S-gate followed by T-gate).
- The corresponding $\chi_k(\mathbf{x})$ functions can be implemented as a network of CNOT and X gates.
- The phase gates are scheduled according to their corresponding $\chi_k(\mathbf{x})$ functions.
- CNOT and X networks are synthesized to drive each phase gate's corresponding $\chi_k(\mathbf{x})$ function
- additional CNOT and X logic is added to return the state to the input state.

Scheduling these T-gates can be done using the matroid partitioning algorithm described in **??**. This guarantees a minimal T-depth for CNOT+T circuits. The terms in Eq. 5 are synthesized and scheduled using that algorithm as demonstrated in Fig. 2.

The next section shows how $\hat{f}(\mathbf{k})$ may be obtained for the three or fewer qubit case using the Boolean Fourier transform.

C. Boolean Fourier Transform

Observe that the $\chi_k(\mathbf{x})$ from Eq. 5 form an orthonormal basis with respect to the below inner product.

$$\langle f(\mathbf{x}), g(\mathbf{x})\rangle = \frac{1}{2^n}\sum_{\mathbf{x}}(-1)^{f(\mathbf{x})}(-1)^{g(\mathbf{x})}, \quad \mathbf{x}\in\{0,1\}^n \tag{6}$$

for $n = |\mathbf{x}| \leq 3$.

That means, we can calculate $\hat{f}(\mathbf{k})$ using the below inner product

x_a	x_b	y	T_{x_a}	T_{x_b}	T_y	$T_{x_a \oplus x_b}$	$T_{x_a \oplus y}$	$T_{x_b \oplus y}$	$T_{x_a \oplus x_b \oplus y}$	$x_a \cdot x_b \cdot y$
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	$\frac{1}{4}$	0	$-\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	0
0	1	0	0	$\frac{1}{4}$	0	$-\frac{1}{4}$	0	$-\frac{1}{4}$	$\frac{1}{4}$	0
0	1	1	0	$\frac{1}{4}$	$\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{4}$	0	0	0
1	0	0	$\frac{1}{4}$	0	0	$-\frac{1}{4}$	$-\frac{1}{4}$	0	$\frac{1}{4}$	0
1	0	1	$\frac{1}{4}$	0	$\frac{1}{4}$	$-\frac{1}{4}$	0	$-\frac{1}{4}$	0	0
1	1	0	$\frac{1}{4}$	$\frac{1}{4}$	0	0	$-\frac{1}{4}$	$-\frac{1}{4}$	0	0
1	1	1	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	$\frac{1}{4}$	1

TABLE I: Truth table-like values of the pseudo-Boolean representation of $x_a \cdot x_b \cdot y$

$$\hat{f}(\mathbf{k}) = \langle f(\mathbf{x}), \chi(\mathbf{x})_{\mathbf{k}} \rangle \forall \mathbf{k} \tag{7}$$

$\hat{f}(\mathbf{k})$ is also known as the Boolean Fourier transform of $f(\mathbf{x})$ [6] [1].

Example 1: As an example, let's derive the coefficients for Eq. 5 with $\mathbf{k} = (k_{x_a}, k_{x_b}, k_y)$. For $\mathbf{k} = 100$, $\chi_{100} = x_a$. Table II enumerates the values and the term $(-1)^{f(\mathbf{x})}(-1)^{\chi_{100}(\mathbf{x})}$ for the inner product.

x_a	x_b	y	$f(\mathbf{x})$	χ_{100}	$(-1)^{\chi_{100}(\mathbf{x})}(-1)^{f(\mathbf{x})}$
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	1	-1
1	0	1	0	1	-1
1	1	0	0	1	-1
1	1	1	1	1	1

TABLE II: The value of χ_{100} and the inner product $\langle f(\mathbf{x}), \chi_{100} \rangle$

Summing the rightmost column, as in Eq. 6:

$$\hat{f}(100) = \langle f(\mathbf{x}), \chi_{100} \rangle = \frac{1}{8}(1 + 1 + 1 + 1 - 1 - 1 - 1 + 1) = \frac{1}{4} \tag{8}$$

Thus,

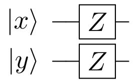
$$f(\mathbf{x}) = \frac{1}{4} x_a + \sum_{\mathbf{k} \neq 000, 100} \hat{f}(\mathbf{k}) \cdot \chi_{\mathbf{k}} \tag{9}$$

The rest of the calculation is omitted for brevity, but readers can verify that it results in Eq. 5.

Synthesizing the phase polynomial of a Boolean function using the Boolean Fourier transform for 3 variables or fewer has an interesting consequence. First, the number of odd multiples of $i\frac{\pi}{4}$ of $\hat{f}(\mathbf{k})$ corresponds to the T-count needed to implement $f(\mathbf{x})$ [5]. This is easy to check, since any even multiple of $i\frac{\pi}{4}$ can be implemented using only S and Z gates, which do not take any T-cost, and any odd multiple can be realized using S and Z gates, plus one T-gate. Since each $\hat{f}(\mathbf{k})$ is unique to each $f(\mathbf{x})$ when $|mathbfx| \leq 3$, $\hat{f}(\mathbf{k})$ has the optimal T-count needed to implement $f(\mathbf{x})$ as a phase oracle.

D. *Synthesizing Phase Oracles from ESOP*

A transformation $|x\rangle |y\rangle \rightarrow (-1)^{x \oplus y} |x\rangle |y\rangle$ can be realized using the below circuit



Thus, given an Exclusive Sum of Products expression of a Boolean function $f(\mathbf{x}) = x_a x_b \cdots x_g \oplus x_i x_j \cdots x_m \oplus \cdots$, we can implement its phase oracle as controlled phase gates of the following form [DCTODO diagram]

This method is used by Qiskit's Phase Oracle [DCTODO ref]. Our proposal revolves around the effective use of such an Exclusive Sum of Products to implement such a phase oracle.

Example 2: We attempt to synthesize the Boolean function $x_a x_b x_c \oplus x_a x_b \bar{x_c}$ into Fig. **??**. The first term is merely a 3-input AND, which is already seen in Fig. 2. Thus we merely insert this circuit to implement the first term. The second term has a negated input, so we insert X on the x_c qubit after the first circuit to negate x_c, again insert Fig. 2 to implement the phase, and finally place another X on x_c to return its value to the original.

The resulting circuit is depicted in Fig. **??**.

III. Motivation

A. *CNOT+T Circuit Optimization*

CNOT+T circuits are notable because, it is straightforward to optimize the T-count. Note that because the phase is a scalar, the actions of phase gates on different qubits all multiply together into the same scalar (essentially adding the powers of eßπ. Therefore, the result of all phase gates in a circuit can be computed thus:

- For every phase gate, calculate the state driving it as a function of the input states, which will be a linear function of the same form as the $\chi_{\mathbf{k}}$ from the previous section.
- Multiply the phase driven by the phase gate by this function.
- Add the phases from all such similar functions

DCTODO

Fig. 3: a phase oracle for $x_a x_b x_c \oplus x_a x_b \bar{x_c}$

[1] All equations that are taken from [6] are written here in terms of the $\{0, 1\}$ basis and $\chi(\mathbf{x})_{\mathbf{k}}$ functions from [2]

- Resynthesize the circuit from the resulting phase polynomial using the methods from Sec. **??**.

The result is that any phase gates driven by the same function will merge together. This means that there is a possibility that T-gates can merge together into an S-gate or Z-gate, reducing the total T-count.

The 3-input case is interesting because it is the largest class of Boolean functions representable by a CNOT+T circuit. Since there are only 256 such Boolean functions, there are potential uses for a library of such oracles.

IV. Proposal

A sum-of-paths representation of a Boolean function can be derived from the Boolean Fourier decomposition [6]. However, this decomposition forms a complete basis for functions with 3 variables or fewer when using the CNOT+T gate set. We demonstrate that composing such decompositions enables the exact implementation of a subset of k-variable Boolean functions.

We use a modified Boolean Fourier decomposition approach, based on [10], to construct our 3-qubit-or-less library. Our synthesis is ancilla-free and schedules T gates following [**?**] to minimize T-depth.

We then apply a modified form of the relative phase function synthesis described in [10] to complete our circuit synthesis.

References

[1] M. Amy, D. L. Maslov, M. Mosca, and M. Rötteler, “A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, pp. 818–830, 2012.

[2] M. Amy, D. Maslov, and M. Mosca, “Polynomial-time t-depth optimization of clifford+t circuits via matroid partitioning,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1476–1489, 2014.

[3] M. Amy, P. Azimzadeh, and M. Mosca, “On the controlled-not complexity of controlled-not–phase circuits,” *Quantum Science and Technology*, vol. 4, no. 1, p. 015002, sep 2018. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/aad8ca

[4] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, “Elementary Gates for Quantum Computation,” *Phys. Rev. A*, vol. 52, pp. 3457–3467, Nov 1995. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.52.3457

[5] M. Amy and M. Mosca, “T-count optimization and reed–muller codes,” *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 4771–4784, 2019.

[6] R. O’Donnell, *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[7] D. Herr, F. Nori, and S. J. Devitt, “Lattice surgery translation for quantum computation,” *New Journal of Physics*, vol. 19, no. 1, p. 013034, jan 2017. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/aa5709

[8] A. G. Fowler and S. J. Devitt, “A bridge to lower overhead quantum computation,” 2013. [Online]. Available: https://arxiv.org/abs/1209.0510

[9] R. Acharya and et al., “Quantum error correction below the surface code threshold,” 2024. [Online]. Available: https://arxiv.org/abs/2408.13687

[10] M. Amy and N. J. Ross, “Phase-state duality in reversible circuit design,” *Phys. Rev. A*, vol. 104, p. 052602, Nov 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.104.052602