Symbolic synthesis of Clifford circuits and beyond

Matthew Amy,[1] Owen Bennett-Gibbs,[2] and Neil J. Ross[3]

[1] School of Computing Science, Simon Fraser University

[2] Department of Mathematics and Statistics, McGill University

[3] Department of Mathematics and Statistics, Dalhousie University

May 2, 2022

Abstract

Path sums are a convenient symbolic formalism for quantum operations with applications to the simulation, optimization, and verification of quantum protocols. Unlike quantum circuits, path sums are not limited to unitary operations, but can express arbitrary linear ones. Two problems, therefore, naturally arise in the study of path sums: the unitarity problem and the extraction problem. The former is the problem of deciding whether a given path sum represents a unitary operator. The latter is the problem of constructing a quantum circuit, given a path sum promised to represent a unitary operator.

In this paper, we show that the unitarity problem is **co-NP**-hard in general, but that it is in **P** when restricted to Clifford path sums. We then provide an algorithm to synthesize a Clifford circuit from a unitary Clifford path sum. The circuits produced by our extraction algorithm are of the form C_1HC_2, where C_1 and C_2 are Hadamard-free circuits and H is a layer of Hadamard gates. We also provide a heuristic generalization of our extraction algorithm to arbitrary path sums. While this algorithm is not guaranteed to succeed, it often succeeds and typically produces natural looking circuits. Alongside applications to the optimization and decompilation of quantum circuits, we demonstrate the capability of our algorithm by synthesizing the standard quantum Fourier transform directly from a path sum.

1 Introduction

The circuit model is ubiquitous in quantum computing, from hardware assembly code to the high-level description of algorithms. Quantum compilation typically amounts to a series of circuit-to-circuit transformations, lowering a circuit, described programmatically in a *circuit description language* over a high-level gate set, down through a series of progressively restrictive gate sets with more fault-tolerance and hardware constraints. Accordingly, quantum algorithms are frequently described at the level of quantum circuits, plugging together inputs and outputs of large, complicated circuits. An exception is the classical oracles used in many quantum algorithms, which are often described at the level of classical programs or Boolean logic, and then *synthesized* as a high-level quantum circuit.

Despite this, the circuit model is often a less than ideal representation of quantum computations. Semantically, circuits expose little information about a computation to the naked eye, particularly when low-level gate sets like Clifford+T are used. Likewise, reasoning about quantum circuits is often a difficult affair involving re-write rules derived from circuit *relations*. Complete sets of relations are only known for a small number of low-level non-universal gate sets [28, 3, 22], or higher-level gate sets [10, 21] which result in a large degree of overhead when compiled. Even with complete sets of relations, simplification of quantum circuits using re-write rules is costly and highly local, yielding results which are typically far from optimal.

Recently, alternative models of quantum computation such as those based on diagrammatic calculi [12, 7], have risen in popularity. These models have seen success in circuit simplification, among other applications, due in part to more effective re-writing methods. However, as most existing quantum computers ultimately run on circuit-like languages, a key component in using such models for circuit transformations is the ability to synthesize or *extract* a circuit back from the representation. This problem has seen a great deal of attention

recently in the context of graphical calculi, resulting in methods for the extraction of Clifford and Clifford+T circuits from ZX diagrams admitting a *generalized flow* [17, 8], as well as theoretical results studying the hardness of this extraction problem [15].

In this paper, we study the problem of synthesizing a unitary circuit given a symbolic expression of a linear operator as a *sum-over-paths* or *path sum* [2]. As any linear operator between 2^n-dimensional Hilbert spaces is representable in this form, we first consider the problem of deciding whether a path sum representations a unitary transformation and show that it is generically **coNP**-hard. Restricting to path sums representing Clifford operators we show that the unitarity problem is in **P**, and that a unitary circuit can be synthesized in time polynomial in the number of qubits. This extraction algorithm produces circuits of the form C_1HC_2, where C_1 and C_2 are Hadamard-free circuits decomposable as a product of S, X, CZ, and CNOT gates, and H is a layer of Hadamard gates. As a consequence we obtain a simple, constructive proof of the 7-stage Bruhat decomposition of the Clifford group [11, 23].

For non-Clifford operations, we give a heuristic for the unitary synthesis of general sums. While our heuristic does not always produce a circuit even if the path sum represents a unitary operator, it succeeds often in practice and typically returns efficient, natural circuits. Alongside circuit optimization, this heuristic has applications to the *decompilation* of quantum circuits, whereby a circuit over a low-level gate set such as Clifford+T is re-written over a higher-level gate set such as multiply-controlled Toffoli gates. We further demonstrate the capability of our algorithm by synthesizing the typical quantum Fourier transform circuit directly from its specification as a sum-over-paths.

2 Path sums

We begin by briefly reviewing the theory of *path sums* [2, 31]. A *path sum representation* of a linear operator $\Psi : \mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$ is an expression for Ψ as a sum indexed by binary variables such as

$$\Psi \left|\vec{x}\right\rangle = \mathcal{N} \sum_{\vec{y} \in \mathbb{Z}_2^k} e^{2\pi i P(\vec{x},\vec{y})} \left|f(\vec{x},\vec{y})\right\rangle, \tag{1}$$

where $\mathcal{N} \in \mathbb{C} \setminus \{0\}$ is a *normalization factor*, and $P : \mathbb{Z}_2^m \times \mathbb{Z}_2^k \to \mathbb{R}$ and $f : \mathbb{Z}_2^m \times \mathbb{Z}_2^k \to \mathbb{Z}_2^n$ are real- and Boolean-valued multilinear polynomials, respectively. The path sum in Equation (1) is said to be *amplitude-balanced* because the normalization factor $\mathcal{N}$ is independent of $\vec{x}$ and $\vec{y}$. We sometimes denote the path sum representation of an operator Ψ by $|\Psi\rangle$.

Equation (1) provides a representation the operator Ψ in the sense that instantiating the binary variables $\vec{x}$ and $\vec{y}$ on both sides of the equality yields a true equation. For this reason, we think of a path sum as a symbolic description of the action of a linear operator on computational basis states.

Example 2.1. The phase gates S and T, as well as the Hadamard gate H, can be represented by path sums as follows, where $\omega = e^{i\pi/4}$:

- $\mathrm{S}\left|x\right\rangle = i^x \left|x\right\rangle$,
- $\mathrm{T}\left|x\right\rangle = \omega^x \left|x\right\rangle$, and
- $\mathrm{H}\left|x\right\rangle = \frac{1}{\sqrt{2}} \sum_y (-1)^{xy} \left|y\right\rangle$.

As path sums involve arithmetic and polynomials over Boolean variables in various rings, it is useful to recall that Boolean algebra can be embedded in any (unital) ring $\mathcal{R}$ using the *lifting* construction defined in [2, Lemma 7.1.6] and reproduced below.

$$\overline{0} = 0_{\mathcal{R}} \qquad \overline{f \wedge g} = \overline{f} \cdot \overline{g}$$
$$\overline{1} = 1_{\mathcal{R}} \qquad \overline{f \oplus g} = \overline{f} + \overline{g} - (2 \cdot \overline{f} \cdot \overline{g})$$

Lifting allows one to use Boolean expressions of variables inside path sums coherently, leading to more natural expressions, as in the following example.

Example 2.2. The gates CNOT and TOF admit the following path sum representations:

- $\text{CNOT}\,|x_1x_2\rangle = |x_1\rangle\,|x_1 \oplus x_2\rangle$ and
- $\text{TOF}\,|x_1x_2x_3\rangle = |x_1\rangle\,|x_2\rangle\,|x_1 \oplus (x_2 \cdot x_3)\rangle$.

The sum-over-paths representations of linear operators has been studied extensively in the context of quantum information [13, 9, 27, 24, 19, 14]. Recent work on the connection to graphical calculi [20, 31] has shown that path sums form a universal model for linear operators over 2^n-dimensional Hilbert spaces through direct translations from universal graphical calculi such as the ZH-calculus [7].

Proposition 2.3 (Unversality)**.** *Any linear operator $\Psi : \mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$ admits a representation as a path sum.*

Example 2.4. Path sums can represent linear operators between spaces of different dimensions. The operators $\eta : \mathbb{C} \to \mathbb{C}^2 \otimes \mathbb{C}^2$ and $\varepsilon : \mathbb{C}^2 \otimes \mathbb{C}^2 \to \mathbb{C}$, which act, respectively, as the unit and counit in the category **FdHilb** [31], can be written as the following path sums:

- $\eta\,|\rangle = \sum_y |yy\rangle$ and
- $\varepsilon\,|x_1x_2\rangle = \frac{1}{2}\sum_y (-1)^{y(x_1+x_2)}\,|\rangle$.

When a path sum represents a row or column vector as above, we drop any empty $|\rangle$. A path sum with a single output dimension, representing a $\mathbb{C}$-valued linear map, is said to be *dimensionless*.

In contrast to graphical calculi, which have a *compositional* structure, path sums are effectively *global* expressions of a linear operator. In other words, the composition (sequential or parallel) of two linear operators is reified into an expression of the form of Equation (1). This is accomplished through the substitution of *free variables* — variables that are not summed over, corresponding to inputs of the operator as elements of the computational basis. We denote the free variables of a path sum $|\Psi\rangle$ by $FV(|\Psi\rangle)$. A path sum with free variables may be thought of as a symbolic state vector in indeterminates $\vec{x} = FV(|\Psi\rangle)$. Hence we use the notation $|\Psi(\vec{x})\rangle$ to denote a path sum expression for the operator Ψ with free variables $\vec{x}$. We use $|\Psi(x)\rangle$ to denote a path sum with a distinguished free variable x.

Through the lifting operation described above, we can define a notion of substitution for path sums with free variables. In particular, given a free variable x appearing in a path sum we may substitute x with any Boolean expression f in all relevant contexts (the phase or the state).

Definition 2.5 (Substitution)**.** Let $|\Psi(x)\rangle$ be a path sum with free variable x and let f be a Boolean expression. Then the *substitution* of x with f is denoted $|\Psi(f)\rangle$.

Reasoning with local binders and free variables in the path sums requires care to avoid variable capture. For instance, let $|\Psi(x)\rangle = \sum_y |x\rangle$. It can be observed that $|\Psi(x)\rangle$ represents the linear operator $\Psi = 2I$. However, if x is substituted with the free variable y, the $\sum_y$ captures y, giving the path sum $|\Psi(y)\rangle = \sum_y |y\rangle$, representing the vector $|0\rangle + |1\rangle$. We assume that substitution is capture-avoiding unless otherwise noted.

A variable may also be *bound* by summing over its possible values. A bound variable may be locally viewed as a free variable by pulling the summation outside of an expression. Indeed, if $|\Psi(x)\rangle$ is a path sum with free variable x, then $\sum_x |\Psi(x)\rangle$ is a path sum with free variables $FV(|\Psi\rangle) \setminus \{x\}$. We sometimes refer to bound variables as *path* variables.

Example 2.6. Recall that the Hadamard gate can be represented as $\text{H}\,|x\rangle = \frac{1}{\sqrt{2}}\sum_y (-1)^{xy}\,|y\rangle$. Alternatively, the Hadamard gate can also be written as $\text{H}\,|x\rangle = \sum_y |\Psi(x,y)\rangle$ where $|\Psi(x,y)\rangle = \frac{1}{\sqrt{2}}(-1)^{xy}\,|y\rangle$ is a path sum in the free variables x and y.

By Proposition 2.3, any linear operator admits a path sum representation. In particular, the composition or tensor product of any two linear operators can also be represented as a path sum. The following proposition gives explicit expression for these constructions in the language of path sums.

Proposition 2.7 (Parallel & Sequential composition)**.** *Let* $\Psi : \mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$ *and* $\Phi : \mathbb{C}^{2^s} \to \mathbb{C}^{2^t}$ *be two linear operators and let* $\Psi \left|\vec{x}\right\rangle = \mathcal{N} \sum_{\vec{y} \in \mathbb{Z}_2^k} e^{2\pi i P(\vec{x},\vec{y})} \left|f(\vec{x},\vec{y})\right\rangle$ *and* $\Phi \left|\vec{w}\right\rangle = \mathcal{M} \sum_{\vec{z} \in \mathbb{Z}_2^l} e^{2\pi i Q(\vec{w},\vec{z})} \left|g(\vec{w},\vec{z})\right\rangle$ *be expressions of* Ψ *and* Φ *as path sums. Then*

$$\Psi \otimes \Phi \left|\vec{x}\right\rangle \left|\vec{w}\right\rangle = \mathcal{N}\mathcal{M} \sum_{\vec{y} \in \mathbb{Z}_2^k} \sum_{\vec{z} \in \mathbb{Z}_2^l} e^{2\pi i [P(\vec{x},\vec{y}) + Q(\vec{w},\vec{z})]} \left|f(\vec{x},\vec{y})\right\rangle \otimes \left|g(\vec{w},\vec{z})\right\rangle$$

$$\Phi \circ \Psi \left|\vec{x}\right\rangle = \mathcal{N} \sum_{\vec{y} \in \mathbb{Z}_2^k} e^{2\pi i P(\vec{x},\vec{y})} \left|\Phi(f(\vec{x},\vec{y}))\right\rangle$$

as path sums, where the latter is well-formed if and only if $s = n$*.*

The parallel and sequential composition of path sums provides a method to compute a symbolic expression for a circuit over a set of basic gates with known path sums. Moreover, for typical gate sets of interest, this representation has size polynomial in the size of the circuit. We give one such result below [2, Corollary 2.15] for the class of circuits which will be most relevant for the purposes of this paper.

Proposition 2.8 (Efficiency for Clifford+T)**.** *Any circuit over Clifford+*T *gates of volume* V *can be expressed as a path sum which has size polynomial in* V *and can be computed in time polynomial in* V*.*

Equational reasoning A major utility of the path sum representation [2] comes from the ability to perform *equational reasoning*. Complete equational theories of Clifford unitaries [2] and more general stabilizer operations [31] have previously been developed. We reformulate these theories here using locally free variables to simplify their presentation.

Proposition 2.9. *Let* Ψ *be a path sum such that* $y \notin FV(\Psi)$ *and let* f *be a Boolean expression such that* $x, y \notin FV(f)$*. Then the following equations hold.*

$$\sum_y \left|\Psi\right\rangle = 2 \left|\Psi\right\rangle \tag{2}$$

$$\sum_{x,y} (-1)^{y(x+f)} \left|\Psi(x)\right\rangle = 2 \left|\Psi(f)\right\rangle \tag{3}$$

$$\sum_y i^y (-1)^{yf} \left|\Psi\right\rangle = \omega\sqrt{2}(-i)^f \left|\Psi\right\rangle \tag{4}$$

$$\sum_y \left|\Psi(y)\right\rangle = \sum_y \left|\Psi(y+f)\right\rangle \tag{5}$$

Equations (2), (3) and (4) are restatements of [2, Proposition 3.1]. Equation (5) is a generalization of the (ket) rule given in [31, Figure 3] and can be derived from Equation (3) as follows:

$$\begin{aligned}
\sum_y \left|\Psi(y)\right\rangle &= \sum_y \left[\frac{1}{2} \sum_{x,z} (-1)^{z(x+(y+f))} \left|\Psi(y)\right\rangle\right] && \text{By Equation (3) where } x, z \notin FV(f) \cup FV(\Psi) \\
&= \sum_x \left[\frac{1}{2} \sum_{y,z} (-1)^{z(y+(x+f))} \left|\Psi(y)\right\rangle\right] && \text{Basic arithmetic} \\
&= \sum_x \left|\Psi(x+f)\right\rangle && \text{By Equation (3)} \\
&= \sum_y \left|\Psi(y+f)\right\rangle && \text{Since } y \notin FV(f) \cup FV(\Psi)
\end{aligned}$$

The first equality above uses the instance $\sum_{x,z} (-1)^{z(x+f')} \left|\Psi(y)\right\rangle = 2 \left|\Psi(y)\right\rangle$ of Equation (3), where $f' = y + f$ and $\left|\Psi(y)\right\rangle$ is viewed as a path sum with zero occurrences of the free variable x.

Example 2.10. Consider the dimensionless path sum $\frac{1}{\sqrt{2}} \sum_y i^y$. By Equation (4) it follows that $\frac{1}{\sqrt{2}} \sum_y i^y = \omega$. Hence, Equation (4) symbolically encodes the fact that $\omega = \frac{1+i}{\sqrt{2}}$.

Relationship to post-selected circuits As with the ZX-calculus and variants, path sum expressions correspond naturally to circuits with ancillas and postselection. Given a path sum expression of a linear operator $\Psi : \mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$ of the form of Equation (1), a circuit implementing Ψ up to a constant scalar factor can be achieved through postselection as follows. First, prepare k ancillary qubits in the state $\frac{1}{\sqrt{2}^k} \sum_{\vec{y} \in \mathbb{Z}_2^k} |\vec{y}\rangle$ by applying Hadamard gates to the $|0\rangle^{\otimes k}$ state. The symbolic state is then prepared up to some garbage $|g(\vec{x}, \vec{y})\rangle$ via the unitary transformation

$$\Psi_{PS} : |\vec{x}\rangle \otimes |\vec{y}\rangle \mapsto e^{2\pi i P(\vec{x},\vec{y})} |f(\vec{x}, \vec{y})\rangle \otimes |g(\vec{x}, \vec{y})\rangle .$$

Finally, the garbage is discarded by postselecting $\mathrm{H}^{\otimes m+k-n} |g(\vec{x}, \vec{y})\rangle = \mathcal{N}' \sum_{\vec{z}} (-1)^{\vec{z} \cdot g(\vec{x},\vec{y})} |\vec{z}\rangle$ on $\vec{z} = \vec{0}$.

Since postselected quantum circuits are believed to be strictly more powerful than non-postselected circuits [1], the rest of this paper focuses on the question of synthesizing unitary circuits for path sums representing unitary transformations, up to normalization.

3 Unitarity testing

We are interested in the synthesis of *unitary* quantum circuits implementing a path sum. As a path sum may represent an arbitrary linear operator, a natural question to ask is whether a given path sum represents a unitary transformation, and hence can be extracted to a unitary circuit. We call this the *unitarity testing* problem for path sums and formulate it as a decision problem below.

Definition 3.1 (UNITARY)**.** *UNITARY* is the set of path-sums $|\Psi\rangle$ where Ψ is a unitary transformation.

The unitarity problem is clearly decidable since we can always explicitly compute a matrix representation of Ψ from a path sum $|\Psi\rangle$. However, since the size of the corresponding matrix is exponential in n, this solution is not efficient. As we show in this section, one should not hope for an efficient solution in general.

Recall that the complexity class **co-NP** consists of the decision problems whose complement belongs to **NP**, and hence is widely believed to be intractable. A canonical complete problem for **co-NP** is the tautology problem, recognizing the set of propositional formulas over the connectives $\{\neg, \wedge, \vee\}$ which are satisfied by every variable assignment. We view a propositional formula in n distinct free variables as a function $\mathbb{Z}_2^n \to \mathbb{Z}_2$, using the standard interpretation $\neg x := 1 + x$, $x \wedge y := xy$ and $x \vee y := x + y - xy$. The application of φ to some $\vec{x} \in \mathbb{Z}_2^n$ is denoted by $\varphi(\vec{x})$.

Definition 3.2 (Tautology)**.** A propositional formula φ in n variables is a *tautology* if $\varphi(\vec{x}) = 1$ for every $\vec{x} \in \mathbb{Z}_2^n$, written $\varphi \equiv 1$.

Definition 3.3 (TAUT)**.** *TAUT* is the set of all propositional formulas that are tautologies.

Theorem 3.4 (Karp's 21 NP-complete problems)**.** *The TAUT problem is* ***co-NP****-complete.*

To reduce TAUT to UNITARY, our goal is to encode a propositional formula φ as a path sum representing the linear operator $\Phi |\vec{x}\rangle = \varphi(\vec{x}) |\vec{x}\rangle$ which is the identity if $\varphi \equiv 1$, and non-unitary otherwise. To do so we establish an encoding of φ as a dimensionless path sum of the form $\varphi(\vec{x}) = \mathcal{N} \sum_{\vec{y}} (-1)^{P(\vec{x},\vec{y})}$, where P is a multilinear Boolean polynomial.

It can readily be observed that $x = 2^{-1} \sum_{y \in \mathbb{Z}_2} (-1)^{y(1+x)}$ for any $x \in \mathbb{Z}_2$. This gives an immediate encoding of any propositional formula in a path sum by extending the lifting discussed in Section 2 to propositional negation and disjunction via the equations $\overline{\neg\varphi} = 1 - \overline{\varphi}$, $\overline{\varphi \vee \psi} = \overline{\varphi} + \overline{\psi} - \overline{\varphi} \cdot \overline{\psi}$, and then setting

$$\varphi(\vec{x}) = 2^{-1} \sum_{y} (-1)^{y(1+\overline{\varphi}(\vec{x}))}.$$

However, the lifting of a propositional formula φ may generally have size exponential in the size of φ. To obtain a polynomial size encoding we rely on the *Tseytin transformation* [29].

Given two propositional formulas φ and ψ, we write $\varphi \leftrightarrow \psi$ for the logical equality of φ and ψ, which is satisfied by an assignment $\vec{x}$ if and only if $\varphi(\vec{x}) = \psi(\vec{x})$. The Tseytin transformation takes a propositional formula φ with k distinct subterms and returns an equisatisfiable conjunction of at most $O(k)$ constant-depth formulas by assigning a fresh propositional variable to the value of each subterm. For instance, given a propositional formula $\varphi = x_1 \wedge (x_2 \vee \neg x_3)$, the Tseytin transformation of φ, denoted $\mathcal{T}(\varphi)$, is

$$\mathcal{T}(\varphi) = z_1 \wedge (z_1 \leftrightarrow x_1 \wedge z_2) \wedge (z_2 \leftrightarrow x_2 \vee z_3) \wedge (z_3 \leftrightarrow \neg x_3).$$

Note that $FV(\varphi) \subseteq FV(\mathcal{T}(\varphi))$ and that the satisfying assignments of φ and $\mathcal{T}(\varphi)$ are in a 1-to-1 correspondence and agree on $FV(\varphi)$.

Given a propositional formula φ, we can encode the Tseytin transformation $\mathcal{T}(\varphi)$ of φ in a dimensionless sum over the free variables of φ using the following encoding of logical equality

$$(\varphi \leftrightarrow \psi)(\vec{x}) = \sum_y (-1)^{y\overline{\varphi}(\vec{x}) + y\overline{\psi}(\vec{x})}.$$

Note that for a clause of the Tseytin transformation $z \leftrightarrow \varphi$ where φ has constant depth, $\overline{\varphi}$ has constant size. If we denote the clauses of $\mathcal{T}(\varphi)$ by $z_1 \leftrightarrow c_1, \ldots, z_k \leftrightarrow c_k$, we may encode $\mathcal{T}(\varphi)$ as a polynomial-size sum by taking the product of each clause and distributing over the summations:

$$\begin{aligned}\mathcal{T}(\varphi)(\vec{x}, \vec{z}) = z_1 \prod_i (z_i \leftrightarrow c_i)(\vec{x}) &= 2^{-1} \sum_y (-1)^{y(1+z_1)} \prod_i 2^{-1} \sum_{y_i} (-1)^{y_i(z_i + \overline{c_i}(\vec{x}))} \\ &= 2^{-(k+1)} \sum_y \sum_{\vec{y} \in \mathbb{Z}_2^k} (-1)^{y(1+z_1) + \sum_i y_i(z_i + \overline{c_i}(\vec{x}))}.\end{aligned}$$

Finally, since the satisfying assignments of φ and $\mathcal{T}(\varphi)$ are in a 1-to-1 correspondence, we see that

$$\varphi(\vec{x}) = \sum_{\vec{y}} \mathcal{T}(\varphi)(\vec{x}, \vec{z}) = 2^{-(k+1)} \sum_y \sum_{\vec{y} \in \mathbb{Z}_2^k} \sum_{\vec{z} \in \mathbb{Z}_2^k} (-1)^{y(1+z_1) + \sum_i y_i(z_i + \overline{c_i}(\vec{x}))}.$$

Proposition 3.5. *Let φ be a propositional formula in n variables and let $\mathcal{T}(\varphi) = z_1 \wedge (\bigwedge_{i=1}^k z_i \leftrightarrow c_i)$. Then for any $\vec{x} \in \mathbb{Z}_2^n$,*

$$\varphi(\vec{x}) = 2^{-(k+1)} \sum_y \sum_{\vec{y} \in \mathbb{Z}_2^k} \sum_{\vec{z} \in \mathbb{Z}_2^k} (-1)^{y(1+z_1) + \sum_i y_i(z_i + \overline{c_i}(\vec{x}))}$$

where the sum on the right hand size has size polynomial in k.

Remark 3.6. The encoding of φ in Proposition 3.5 is interesting because it gives a polynomial-size expression *in the same variables as* φ. This is in constrast to the propositional Tseytin transformation which gives an encoding over a superset of free variables, and hence only remains equi-satisfiable. In particular, $\mathcal{T}(\cdot)$ does not preserve tautologies, whilst our encoding does when viewed as a $\{0,1\}$-valued function.

Given the encoding of φ above, we can now prove **co-NP**-hardness of the unitarity testing problem by a reduction from TAUT.

Theorem 3.7. *The unitarity testing problem is **co-NP**-hard*

Proof. By many-one reduction from TAUT to UNITARY. Given a propositional formula φ in n variables, define $\Psi : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ to be the linear operator given by $\Psi |\vec{x}\rangle = \varphi(\vec{x}) |\vec{x}\rangle$. By Proposition 3.5, Ψ admits the following representation as a path sum

$$\Psi |\vec{x}\rangle = 2^{-(k+1)} \sum_y \sum_{\vec{y} \in \mathbb{Z}_2^k} \sum_{\vec{z} \in \mathbb{Z}_2^k} (-1)^{y(1+z_1) + \sum_i y_i(z_i + \overline{c_i}(\vec{x}))} |\vec{x}\rangle$$

which has size polynomial in the number of subterms of φ. Hence all that remains is to show that Ψ is unitary if and only if φ is a tautology. If suffices to observe that, for any $\vec{x} \in \mathbb{Z}_2^n$, $\Psi |\vec{x}\rangle = \varphi(\vec{x}) |\vec{x}\rangle = |\vec{x}\rangle$ if $\varphi(\vec{x}) = 1$ and $\vec{x} \in \mathbb{Z}_2^n$, $\Psi |\vec{x}\rangle = \varphi(\vec{x}) |\vec{x}\rangle = 0$ otherwise. In particular, if $\varphi(\vec{x}) = 1$ for all $\vec{x} \in \mathbb{Z}_2$, then $\Psi = I_n$. Otherwise, there exists $\vec{x} \in \mathbb{Z}_2$ such that $\Psi |\vec{x}\rangle = 0$ and hence Ψ is non-unitary, as required. ∎

4 Clifford synthesis

In this section we look at the problems of synthesis and unitarity testing in the restricted case of Clifford operations. The synthesis of Clifford circuits has applications both to randomized benchmarking, as well as to the design and analysis of error correction circuits. We first review the definition of the Clifford group.

Definition 4.1 (Pauli group)**.** The n-qubit *Pauli group* $\mathcal{P}_n$ is the group of n-fold tensor products of Pauli operators $\{I, X, Y, Z\}$.

Definition 4.2 (Clifford group)**.** The n-qubit *Clifford group* is the group $\mathcal{C}_n = \{U \in U_{2^n} \mid U\mathcal{P}_n U^\dagger = \mathcal{P}_n\}$.

A well-known consequences of the Gottesman-Knill theorem is the fact that, up to global phases, the Clifford group is generated by $\{\text{H}, \text{S}, \text{CNOT}\}$. We may use this fact to give a convenient path sum representation of Clifford operations.

Proposition 4.3. *Every Clifford operator $\Psi : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ can be written as a sum of the form*

$$\Psi \left|\vec{x}\right\rangle = \frac{\omega^l}{\sqrt{2^k}} \sum_{\vec{y} \in \mathbb{Z}_2^k} i^{L(\vec{x},\vec{y})} (-1)^{Q(\vec{x},\vec{y})} \left|f(\vec{x},\vec{y})\right\rangle \tag{6}$$

where $l \in \mathbb{Z}_8$, $L : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \to \mathbb{Z}_4$ is linear, $Q : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \to \mathbb{Z}_2$ is pure quadratic, and $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \to \mathbb{Z}_2$ is affine.

Proof. As the Clifford group generators CNOT, S, and H can be written in the form of Equation (6), it only remains to show that the composition of two such sums can be written in the form of Equation (6). It suffices to note that substitution of a variable with an affine Boolean expression does not increase the degree of Q or f, while substitution in L produces a quadratic form with degree 2 terms divisible by 2. □

We call an expression of the form of Equation (6) a *Clifford* path sum. In the context of *stabilizer states* — states $\left|\psi\right\rangle = C\left|\vec{0}\right\rangle$ for some $C \in \mathcal{C}_n$ — this representation is well-known by various names, including the *quadratic form expansion* [14] and the *affine representation* [16, 26]. Through the inclusion of free parameters we can represent stabilizer states, Clifford unitaries, or Clifford circuits with ancillas in this form.

We next define a *normal form* for Clifford path sums which will be useful for circuit synthesis.

Definition 4.4 (Normal form)**.** A Clifford path sum for Ψ is in *normal form* if, up to a reordering of qubits,

$$\Psi \left|\vec{x}\right\rangle = \frac{\omega^l}{\sqrt{2^k}} \sum_{\vec{y} \in \mathbb{Z}_2^k} i^{L(\vec{x},\vec{y})} (-1)^{Q(\vec{x},\vec{y})} \left|\vec{y}\right\rangle \otimes \left|f(\vec{x},\vec{y})\right\rangle, \tag{7}$$

where $l \in \mathbb{Z}_8$, $L : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \to \mathbb{Z}_4$ is linear, $Q : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \to \mathbb{Z}_2$ is pure quadratic, and $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^k \to \mathbb{Z}_2$ is affine.

The normal form above corresponds to re-writing the sum over a minimal set of vectors spanning the affine subspace of $\mathbb{Z}_2^n$ given by $\{f(\vec{x},\vec{y}) \mid \vec{y} \in \mathbb{Z}_2^k\}$. The following proposition states that Equations (2), (3), (4) and (5) suffice to re-write a unitary Clifford path sum into normal form.

Proposition 4.5. *Let $\left|\Psi\right\rangle$ be a Clifford path sum. There exists a re-writing procedure which will terminate with $\left|\Psi\right\rangle$ in normal form if Ψ is unitary and runs in time polynomial in the size of $\left|\Psi\right\rangle$.*

Proof. For each path variable y_i, if there exists j such that $f_j(\vec{x},\vec{y}) = y_i \oplus f'(\vec{x},\vec{y})$, Equation (5) can be applied to substitute y_i with $y_i \oplus f'(\vec{x},\vec{y})$. If no such j exists, either Ψ is unitary and one of Equations (2), (3) and (4) necessarily applies to eliminate y_i [2], or no rule applies and Ψ is non-unitary. □

Remark 4.6. Proposition 4.5 also holds for non-square $\Psi : \mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$ with $m \leq n$ so long as Ψ is an *isometry* — that is, if Ψ corresponds to a Clifford circuit with some ancillas or fixed inputs.

Corollary 4.7. *The unitarity testing problem for Clifford path sums is in* ***P****.*

Proof. Given a Clifford path sum $|\Psi\rangle$, we can construct a path sum representation of $\Psi^\dagger$ efficiently using η, ϵ, and negating L. Then by Proposition 4.5 we can normalize the path sum representations of $\Psi\Psi^\dagger$ and $\Psi^\dagger\Psi$ each in polynomial time. If either fails to produce a normal form, then one of $\Psi\Psi^\dagger$ or $\Psi^\dagger\Psi$ is non-unitary and hence Ψ is non-unitary. If both are reduced to normal form, it suffices to observe that we can check whether a Clifford path sum in normal form represents the identity transformation in polynomial time. □

It is now straightforward to compute a circuit implementing a (unitary) Clifford path sum from its normalized form. If we decompose f, L, and Q as $f(\vec{x},\vec{y}) = f_x(\vec{x}) + f_y(\vec{y}) + \vec{b}$, $L(\vec{x},\vec{y}) = L_x(\vec{x}) + L_y(\vec{y})$, and $Q(\vec{x},\vec{y}) = Q_x(\vec{x}) + Q_y(\vec{y}) + \sum_{i=1}^k y_i R_i(\vec{x})$ then the normal form can be written as the following sequence of transformations:

$$\begin{aligned}
|\vec{x}\rangle &\mapsto \omega^l i^{L_x(\vec{x})}(-1)^{Q_x(\vec{x})}|\vec{x}\rangle \\
|\vec{x}\rangle &\mapsto |R(\vec{x})\rangle\,|f_x(\vec{x})\rangle \\
|R(\vec{x})\rangle\,|f_x(\vec{x})\rangle &\mapsto \frac{1}{\sqrt{2^k}}\sum_{\vec{y}\in\mathbb{Z}_2^k}(-1)^{\sum_i y_i R_i(\vec{x})}|\vec{y}\rangle\,|f_x(\vec{x})\rangle \\
|\vec{y}\rangle\,|f_x(\vec{x})\rangle &\mapsto |\vec{y}\rangle\,|f_x(\vec{x}) + f_y(\vec{y}) + \vec{b}\rangle \\
|\vec{y}\rangle\,|f(\vec{x},\vec{y})\rangle &\mapsto i^{L_y(\vec{y})}(-1)^{Q_y(\vec{y})}|\vec{y}\rangle\,|f(\vec{x},\vec{y})\rangle
\end{aligned}$$

This gives a circuit of the form $U(\mathrm{H}^{\otimes k}\otimes \mathrm{I}_{n-k})V$ where U and V are generalized permutations contained in the Clifford group. Moreover, $|\vec{x}\rangle \mapsto |R(\vec{x})\rangle\,|f_x(\vec{x})\rangle$ is the only operator which may be non-unitary, and in particular is unitary if and only if Ψ is. Note that the unitarity of Ψ hence forces $\{R_i\}$ to be linearly independent and for R_i to be non-zero. This is summarized in Algorithm 1.

Algorithm 1 Clifford synthesis algorithm

1. Normalize $|\Psi\rangle$ in the form $\frac{\omega^l}{\sqrt{2^k}}\sum_{\vec{y}\in\mathbb{Z}_2^k} i^{L(\vec{x},\vec{y})}(-1)^{Q(\vec{x},\vec{y})}|\vec{y}\rangle \otimes |f(\vec{x},\vec{y})\rangle$ up to qubit reordering.
2. Decompose f, L, and Q as $f(\vec{x},\vec{y}) = f_x(\vec{x}) + f_y(\vec{y}) + \vec{b}$, $L(\vec{x},\vec{y}) = L_x(\vec{x}) + L_y(\vec{y})$ and $Q(\vec{x},\vec{y}) = Q_x(\vec{x}) + Q_y(\vec{y}) + \sum_i y_i R_i(\vec{x})$ where each R_i is linear.
3. Synthesize circuits for the following linear transformations:
   - $D\,|\vec{x}\rangle = i^{L_x(\vec{x})}(-1)^{Q_x(\vec{x})}|\vec{x}\rangle$
   - $U\,|\vec{x}\rangle = |R(\vec{x})\rangle\,|f_x(\vec{x})\rangle$
   - $V\,|\vec{y}\rangle\,|f_x(\vec{x})\rangle = |\vec{y}\rangle\,|f_x(\vec{x}) + f_y(\vec{y}) + \vec{b}\rangle$
   - $P\,|\vec{y}\rangle\,|f(\vec{x},\vec{y})\rangle = i^{L_y(\vec{y})}(-1)^{Q_y(\vec{y})}|\vec{y}\rangle\,|f(\vec{x},\vec{y})\rangle$
4. Return $\omega^l PV(\mathrm{H}^{\otimes k}\otimes \mathrm{I}_{n-k})UD$ with qubits appropriately reordered.

Theorem 4.8. *Let $\Psi : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ be expressed as a Clifford path sum. If Ψ is unitary, then Algorithm 1 produces a circuit over $\{\omega, \mathrm{CNOT}, \mathrm{X}, \mathrm{CZ}, \mathrm{S}, \mathrm{H}\}$ implementing Ψ in time polynomial in the size of the expression. Moreover, this circuit can be written up to global phase as an 8-stage circuit of the form*

$$\mathrm{S}\cdot\mathrm{CZ}\cdot\mathrm{CNOT}\cdot\mathrm{H}\cdot\mathrm{CNOT}\cdot\mathrm{X}\cdot\mathrm{CZ}\cdot\mathrm{S}$$

Proof. That $\Psi = \omega^l PV(\mathrm{H}^{\otimes k}\otimes \mathrm{I}_{n-k})UD$ follows by an easy calculation.

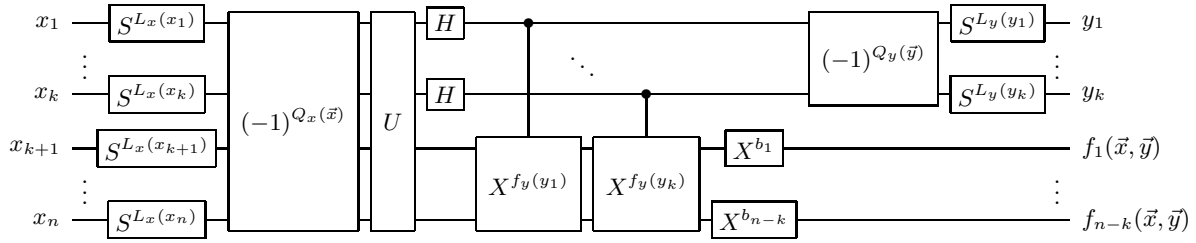
By Proposition 4.5, Ψ can be written up to a permutation of qubits in normal form in polynomial time. Since L_x and L_y are linear, and Q_x and Q_y are pure quadratic, D and P can each be synthesized using a

single stage each of S and CZ gates — one S^m gate for each non-zero term of $L_{\{x,y\}}$ and one CZ gate for each non-zero term of $Q_{\{x,y\}}$. Likewise, since $f_y(\vec{y})$ is linear, V can be synthesized in time polynomial in n using a single stage each of CNOT and X gates — one gate for each non-zero entry of $f_y(\vec{y})$ and $\vec{b}$. Finally, $U|\vec{x}\rangle = |R(\vec{x})\rangle \otimes |f_x(\vec{x})\rangle$ can be synthesized over {CNOT} in polynomial time using Gaussian elimination if and only if U is invertible. Moreover, since

$$U = \omega^{-l}(\mathrm{H}^{\otimes k} \otimes \mathrm{I}_{n-k})V^\dagger P^\dagger \Psi D^\dagger,$$

it follows that U is invertible if and only if Ψ is unitary. □

We give a diagrammatic presentation of Theorem 4.8 showing the circuit schematically below.



Discussion In [23] a 7 stage decomposition of the Clifford group of the form S · CZ · C · H · C · CZ · S was given, where C is circuit implementing an affine permutation. As affine permutations require both CNOT and X gates to implement without ancillas — and moreover X can not be written in the form S · CZ · CNOT · H · CNOT · CZ · S — our projective decomposition reduces the equivalent 9-stage projective decomposition of [23] to 8 stages.

It can also be observed that with a minor modification, Algorithm 1 suffices to synthesize Clifford circuits with ancillas, including circuits for preparing stabilizer states. In particular, if $\Psi : \mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$ is an isometric Clifford path sum with $m \leq n$, the only modification needed is in the synthesis of $U|\vec{x}\rangle = |R(\vec{x})\rangle \otimes |f_x(\vec{x})\rangle$. If indeed $m \leq n$, then a Clifford circuit with ancillas exists and can be synthesized if and only if $\{R_i\} \cup \{(f_x)_i\}$ contains m linearly independent (row) vectors. This produces a 5-stage circuit of the form H · CNOT · X · CZ · S for the preparation of an arbitrary stabilizer state up to global phase. This stabilizer state decomposition was previously given in [26].

5 General synthesis

We now consider the more challenging problem of synthesizing a unitary circuit from an arbitrary path sum. Our method attempts to iteratively reduce the number of summed variables in a path sum by alternately applying generalized permutations and Hadamard gates to the symbolic state. Recall that a *(unitary) generalized permutation* is a permutation matrix whose nonzero entries are elements of $\mathbb{T} = \{z \in \mathbb{C} \mid |z| = 1\}$. The generalized permutations are generated by the gates

$$\Lambda_k(X) : |\vec{x}\rangle|y\rangle \mapsto |\vec{x}\rangle|y \oplus \prod_i x_i\rangle \quad \text{and} \quad \Lambda_k(R_Z(\theta)) : |\vec{x}\rangle \mapsto e^{2\pi i\theta \prod_i x_i}|\vec{x}\rangle.$$

Together with the Hadamard gate this forms an exactly universal set as it includes every single-qubit unitary along with the CNOT gate.

The following fact forms the basis of our synthesis algorithm. It gives a condition on a path sum which allows a summed variable to be eliminated by multiplication with a Hadamard gate.

Proposition 5.1. *Let* $\Psi : \mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$ *be a linear operator where*

$$\Psi|\vec{x}\rangle = \mathcal{N} \sum_{z \in \mathbb{Z}_2} \sum_{\vec{y} \in \mathbb{Z}_2^k} (-1)^{zQ(\vec{x},\vec{y})} e^{2\pi i P(\vec{x},\vec{y})} |z\rangle \otimes |f(\vec{x},\vec{y})\rangle.$$

Then $(\mathrm{H} \otimes \mathrm{I}_{n-1})\Psi|\vec{x}\rangle = \sqrt{2}\mathcal{N} \sum_{\vec{y} \in \mathbb{Z}_2^k} e^{2\pi i P(\vec{x},\vec{y})} |Q(\vec{x},\vec{y})\rangle \otimes |f(\vec{x},\vec{y})\rangle$.

Proof. By Equation (3), since $(\mathrm{H} \otimes \mathrm{I}_{n-1})\Psi \left|\vec{x}\right\rangle = \frac{1}{\sqrt{2}}\mathcal{N}\sum_z \sum_{\vec{y}} (-1)^{zQ(\vec{x},\vec{y})+zz'} e^{2\pi i P(\vec{x},\vec{y})} \left|z'\right\rangle \otimes \left|f(\vec{x},\vec{y})\right\rangle$ □

Note that Proposition 5.1 is essentially an inversion of the H gate, $\mathrm{H}^\dagger : \sum_z (-1)^{xz} \left|z\right\rangle \mapsto \left|x\right\rangle$. We say that a variable z is *reducible* in the path sum $\left|\Psi\right\rangle$ if $\left|\Psi\right\rangle$ is in the form of Proposition 5.1.

Definition 5.2 (Reducible)**.** A variable z is *reducible* in an expression of $\Psi : \mathbb{C}^{2^m} \to \mathbb{C}^{2^n}$ if, up to qubit reordering, it has the form

$$\Psi \left|\vec{x}\right\rangle = \mathcal{N} \sum_{z \in \mathbb{Z}_2} \sum_{\vec{y} \in \mathbb{Z}_2^k} (-1)^{zQ(\vec{x},\vec{y})} e^{2\pi i P(\vec{x},\vec{y})} \left|z\right\rangle \otimes \left|f(\vec{x},\vec{y})\right\rangle .$$

At a high level, our algorithm proceeds by attempting to synthesize a generalized permutation which will leave some path variable reducible. If the process terminates with remaining summed variables, or an unsynthesizeable ground term $e^{2\pi i P(\vec{x})} \left|f(\vec{x})\right\rangle$, the algorithm fails to produce a circuit. Algorithm 2 gives the high-level algorithm in pseudo-code.

Algorithm 2 General path sum synthesis algorithm

1. Normalize $\left|\Psi\right\rangle$ using Equations (2), (3) and (4)
2. For each remaining path variable y in $\left|\Psi\right\rangle$
   (a) If there exists a generalized permutation U such that y is reducible in $U^\dagger \left|\Psi\right\rangle$,
      i. $\left|\Psi\right\rangle \leftarrow (\mathrm{H} \otimes \mathrm{I}_{n-1})U^\dagger \left|\Psi\right\rangle$
      ii. Emit $U(\mathrm{H} \otimes \mathrm{I}_{n-1})$
      iii. Return to step 1
3. If path variables remain or Ψ is non-unitary, fail. Otherwise, synthesize and emit $\Psi^\dagger$.

Finding such a generalized permutation is highly non-trivial. Our method applies a series of symbolic simplifications, corresponding to $\Lambda_k(X)$ and $\Lambda_k(R_Z(\theta))$ gates, to the term $e^{2\pi i P(\vec{x},\vec{y})} \left|f(\vec{x},\vec{y})\right\rangle$. If these simplifications fail to leave any variable reducible, we fall back to an exponential-time procedure aimed at computing a substitution of the form in Equation (5) which will make some variable reducible. These heuristics are described in Appendix A.

6 Experiments

To test the performance and utility of our synthesis methods, we implemented Algorithms 1 and 2 in the FEYNMAN[1] software package. In this section, we briefly detail our investigations into applications to the optimization and decompilation of circuits, and to specification-based synthesis. All Clifford circuits synthesized have been checked for correctness using the method of [2]. For Algorithm 2, as the method of [2] often fails to verify circuits extracted using Equation (5), we instead validated correctness of our synthesis procedure by verifying the individual synthesis steps each on 1000 randomly generated unitary path sums extracted from Clifford+T circuits. Table 1 gives some statistics from experiments re-synthesizing random circuits.

Circuit optimization One of the key factors in phase folding optimizations [4, 25] is the placement of Hadamard gates. It was shown in [5] that the T-count in a Clifford+T circuit can be upper bounded by $O(hn^2)$, where h is the number of Hadamard layers in the circuit. As our Clifford synthesis algorithm produces circuits with just a single layer of Hadamard gates, it is natural to ask whether we can optimize T-count by reducing the number of Hadamard layers in Clifford+T circuits.

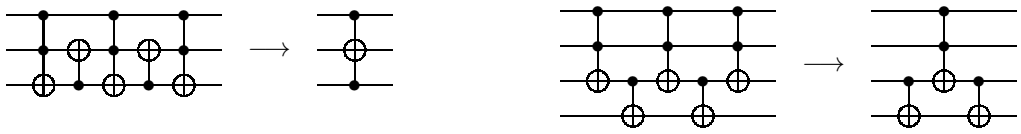
[1] Available at https://github.com/meamy/feynman.

	n	# gates	# circuits	avg. time (s)	avg. change (+/-)	success
Clifford	20	500	1000	0.137	+19.2%	–
	20	1000	1000	0.481	-12.9%	–
	50	500	1000	0.264	+90.7%	–
	50	1000	1000	1.518	+129.1%	–
Clifford+T	20	100	1000	0.010	+48.9%	99.9%
	20	200	1000	0.045	+93.7%	94.9%
	20	300	1000	0.097	+115.9%	74.7%
	50	100	1000	0.016	+33.5%	100.0%
	50	200	1000	0.044	+49.0%	100.0%
	50	300	1000	0.104	+79.4%	99.6%

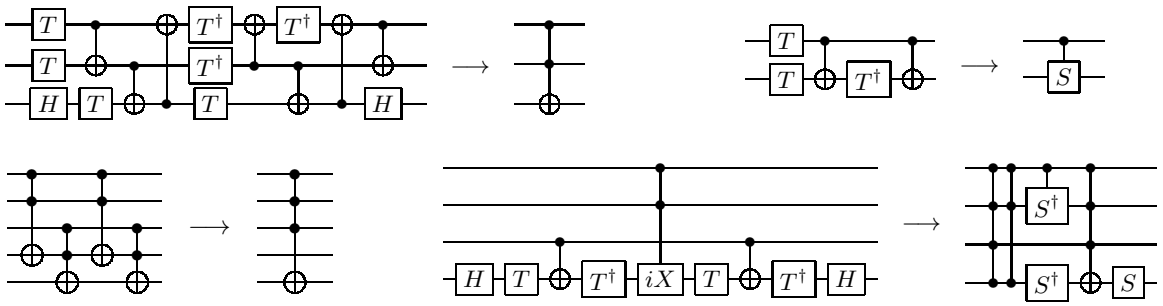
Table 1: Re-synthesis results for randomly generated circuits on n qubits. Avg. change gives the average percent increase (+) or decrease (-) in gate count.

We implemented a Clifford sub-circuit normalization method (the `-clifford` pass in FEYNOPT) using Algorithm 1 to re-synthesize simple greedily chosen Clifford sub-circuits. We tested the effect on T-count optimization by applying Clifford normalization together with phase folding and compared it against [18] on the benchmark set of [4]. In all but 4 benchmarks, the same T-count was achieved by normalizing greedily chosen Clifford sub-circuits and applying phase polynomial optimizations. In two of those cases, $\texttt{qcla-com}_7$ and $\texttt{csla-mux}_3$, our method produced lower T-count circuits — 94 (down from 95) and 60 (down from 62), respectively. For the other two cases, `ham15-med` and $\texttt{adder}_8$, our method produced worse results — 230 (up from 212) and 215 (up from 173), respectively.

More broadly, we might expect to be able to optimize a circuit by resynthesizing its simplified path sum using the general synthesis algorithm Algorithm 2. This is often effective when the path sum is simple, as in the resynthesized circuits corresponding to sub-circuits of the $\texttt{adder}_8$ benchmark, but as the path sum becomes increasingly complex extraction typically performs worse than human designs. We leave it as an avenue for future work to make symbolic synthesis practical for circuit optimization, and in particular to develop effective peephole optimization procedures.



Decompilation An interesting application of our symbolic synthesis algorithm is to the *decompilation* of quantum circuits. Classically, decompilation is the process of translating a program in a low-level language to equivalent high-level source code, typically used for reverse engineering or recompilation. As the gate set targeted by Algorithm 2 is quite high-level, in many cases it can be used to effectively decompile lower-level circuits. This decompilation can potentially help developers to examine the high-level structure of a low-level circuit, and also allow optimizations targeting higher level gate sets to be performed on circuits written over low-level gate sets, such as Clifford+T. Below we give some examples of standard circuits from the literature decompiled using Algorithm 2. The decompiler can be accessed with the `-decompile` option in FEYNOPT.



The bottom right circuit above is a relative phase Toffoli gate implementation taken from [6]. The utility of decompilation is apparant here, as both the fact that it implements a Toffoli up to phase and the exact form of the relative phase can be readily observed from the decompiled circuit.

Specification-based synthesis In [2] it was noted that path sums offer a convenient form of logical specification for many quantum computations, being very close to the "textbook" specification. Algorithm 2 gives a method of synthesizing a circuit directly from such a specification. Such specifications include not only classical reversible functions such as $|x_1x_2x_3\rangle \mapsto |x_1x_2(x_3 \oplus x_1x_2)\rangle$, which can be synthesized by existing reversible circuit synthesis methods, but also classical functions "in the phase," up to relative phases, or inside superpositions. We illustrate this by using Algorithm 2 to synthesize the quantum Fourier transform.

Recall that the n-qubit quantum fourier transform can be expressed as $QFT_n |\vec{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\vec{y} \in \mathbb{Z}_2^n} \omega_{2^n}^{\vec{x}\vec{y}} |\vec{y}\rangle$ where $\vec{x}\vec{y}$ is the integer product of $\vec{x}$ and $\vec{y}$. In the 3 qubit case, expanding the integer multiplication to a multilinear polynomial we have

$$QFT_3 |x_1x_2x_3\rangle = \frac{1}{\sqrt{2^3}} \sum_{y_1,y_2,y_3} \omega^{x_3y_3} i^{x_3y_2+x_2y_3} (-1)^{x_3y_1+x_2y_2+x_1y_3} |y_1y_2y_3\rangle$$

where y_1 is reducible, and in particular

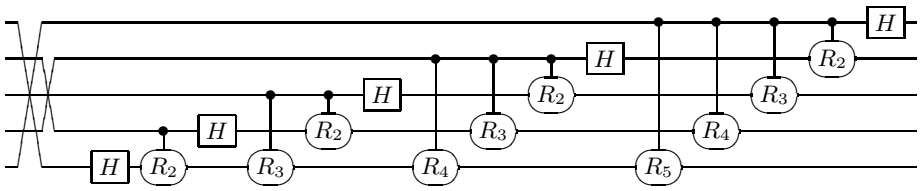
$$(\mathrm{H} \otimes \mathrm{I}_2) QFT_3 |x_1x_2x_3\rangle = \frac{1}{\sqrt{2^2}} \sum_{y_1,y_2} \omega^{x_3y_3} i^{x_3y_2+x_2y_3} (-1)^{x_2y_2+x_1y_3} |x_3y_2y_3\rangle .$$

While neither of y_2 or y_3 are reducible above, the $\omega^{x_3y_3}$ and $i^{x_3y_2}$ terms can be eliminated by applying controlled-$\mathrm{T}^\dagger$ and -$\mathrm{S}^\dagger$ gates, respectively, leaving y_2 reducible:

$$(\Lambda(\mathrm{S}^\dagger) \otimes \mathrm{I})(\mathrm{SWAP} \otimes \mathrm{I})(\mathrm{I} \otimes \Lambda(\mathrm{T}^\dagger))(\mathrm{SWAP} \otimes \mathrm{I})(\mathrm{H} \otimes \mathrm{I}_2) QFT_3 |x_1x_2x_3\rangle = \frac{1}{\sqrt{2^2}} \sum_{y_1,y_2} i^{x_2y_3} (-1)^{x_2y_2+x_1y_3} |x_3y_2y_3\rangle .$$

After eliminating y_2, the process repeats for y_1, leaving a final permutation to be synthesized.

A 5 qubit QFT circuit synthesized with our implementation is shown verbatim below, where $R_k := R_Z(1/2^k)$. We were able to synthesize instances on up to 50 qubits in just seconds on a desktop computer.



7 Conclusion

In this paper we looked at the problem of synthesis of unitary quantum circuits from symbolic expressions as sums-over-paths. We showed that we cannot hope to efficiently synthesize a circuit from a general path sum efficiently, as the problem of checking whether there the path sum represents a unitary transformation is itself **co-NP**-hard. A stronger result was given recently for the extraction of ZX-diagrams [15], though their work did not address the complexity of the potentially easier problem of unitarity testing. The problem of unitarity testing for ZX-diagrams is likewise believed to be intractable [30].

For the restricted case of Clifford operations, we showed that a circuit can be synthesized efficiently in the form C_1HC_2 for Hadamard-free Clifford circuits C_1 and C_2. For more general path sums we gave a heuristic based on symbolic manipulation and simplification of the sum. We experimentally validated our method, showing that most path sums corresponding to unitary transformations can in fact be synthesized. Moreover, our algorithm is capable of producing natural, high-level circuit designs for some path sums, including the quantum Fourier transform. It remains as a course of future work however to develop a complete synthesis algorithm, as well as to reduce the cost of synthesized circuits.

References

[1] S. Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A*, 461(2063):3473–3482, 2005.

[2] M. Amy. Towards large-scale functional verification of universal quantum circuits. In *Proceedings of the 15th International Conference on Quantum Physics and Logic*, QPL'18, pages 1–21, 2018.

[3] M. Amy, J. Chen, and N. J. Ross. A Finite Presentation of CNOT-Dihedral Operators. In *Proceedings of the 14th International Conference on Quantum Physics and Logic*, QPL'17, pages 84–97, 2017.

[4] 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*, 33(10):1476–1489, 2014.

[5] M. Amy and M. Mosca. T-count optimization and Reed-Muller codes. *IEEE Transactions on Information Theory*, 65(8):4771–4784, 2019.

[6] M. Amy and N. J. Ross. The phase/state duality in reversible circuit design. *Physical Review A*, 104:052602, 2021.

[7] M. Backens and A. Kissinger. ZH: A complete graphical calculus for quantum computations involving classical non-linearity. In *Proceedings of the 15th International Conference on Quantum Physics and Logic*, QPL'18, pages 23–42, 2018.

[8] M. Backens, H. Miller-Bakewell, G. de Felice, L. Lobski, and J. van de Wetering. There and back again: A circuit extraction tale. *Quantum*, 5:421, Mar. 2021.

[9] D. Bacon, W. van Dam, and A. Russell. Analyzing algebraic quantum circuits using exponential sums, 2008.

[10] X. Bian and P. Selinger. Generators and relations for $U_n(\mathbb{Z}[1/2, i])$. In *Proceedings of the 18th International Conference on Quantum Physics and Logic*, QPL'21, pages 145–164, 2021.

[11] S. Bravyi and D. L. Maslov. Hadamard-free circuits expose the structure of the Clifford group. *IEEE Transactions on Information Theory*, 67:4546–4563, 2021.

[12] B. Coecke and R. Duncan. Interacting quantum observables. In *Proceeds of the The 35th International Colloquium on Automata, Languages and Programming*, ICALP'08, pages 298–310, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[13] C. M. Dawson, A. P. Hines, D. Mortimer, H. L. Haselgrove, M. A. Nielsen, and T. J. Osborne. Quantum computing and polynomial equations over the finite field $\mathbb{Z}_2$. *Quantum Information and Computation*, 5(2):102–112, 2005.

[14] N. de Beaudrap and S. Herbert. Fast stabiliser simulation with quadratic form expansions. Available from arXiv:2109.08629, Sep. 2021.

[15] N. de Beaudrap, A. Kissinger, and J. van de Wetering. Circuit Extraction for ZX-diagrams can be #P-hard. Available from arXiv:2202.09194, 2022.

[16] J. Dehaene and B. De Moor. Clifford group, stabilizer states, and linear and quadratic operations over gf(2). *Phys. Rev. A*, 68:042318, Oct 2003.

[17] R. Duncan, A. Kissinger, S. Perdrix, and J. van de Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum*, 4:279, June 2020.

[18] A. Kissinger and J. van de Wetering. Reducing the number of non-clifford gates in quantum circuits. *Phys. Rev. A*, 102:022406, Aug 2020.

[19] D. E. Koh, M. D. Penney, and R. W. Spekkens. Computing quopit Clifford circuit amplitudes by the sum-over-paths technique. *Quantum Information and Computation*, 17(13&14):1081–1095, 2017.

[20] L. Lemonnier, J. van de Wetering, and A. Kissinger. Hypergraph simplification: Linking the path-sum approach to the ZH-calculus. In *Proceedings of the 17th International Conference on Quantum Physics and Logic*, QPL'20, 2020.

[21] S. M. Li, N. J. Ross, and P. Selinger. Generators and relations for the group $O_n(\mathbb{Z}[1/2])$. In *Proceedings of the 18th International Conference on Quantum Physics and Logic*, QPL'21, pages 210–264, 2021.

[22] J. Makary, N. J. Ross, and P. Selinger. Generators and relations for real stabilizer operators. In *Proceedings of the 18th International Conference on Quantum Physics and Logic*, QPL'21, pages 14–36, 2021.

[23] D. L. Maslov and M. Roetteler. Shorter stabilizer circuits via Bruhat decomposition and quantum circuit transformations. *IEEE Transactions on Information Theory*, 64:4729–4738, 2018.

[24] A. Montanaro. Quantum circuits and low-degree polynomials over $\mathbb{F}_2$. *Journal of Physics A: Mathematical and Theoretical*, 50(8):084002, 2017.

[25] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov. Automated Optimization of Large Quantum Circuits with Continuous Parameters. *npj Quantum Information*, 4(1):23, 2018.

[26] M. V. d. Nest. Classical simulation of quantum computation,the gottesman-knill theorem, and slightly beyond. *Quantum Information and Computation*, 10(3&4):0258–0271, 2010.

[27] T. Rudolph. Simple encoding of a quantum circuit amplitude as a matrix permanent. *Physical Review A*, 80:054302, Nov 2009.

[28] P. Selinger. Generators and relations for n-qubit Clifford operators. *Logical Methods in Computer Science*, Volume 11, Issue 2, 2015.

[29] G. S. Tseitin. On the complexity of derivation in propositional calculus. In *Automation of reasoning*, pages 466–483. Springer, 1983.

[30] J. van de Wetering, 2021. Private communication.

[31] R. Vilmart. The Structure of Sum-Over-Paths, its Consequences, and Completeness for Clifford. In *Foundations of Software Science and Computation Structures*, volume 12650 of *FoSSaCS'21*, pages 531–550, Luxembourg, Luxembourg, 2021.

A Finding generalized permutations

In this appendix we detail our method for finding a generalized permutation in step 2.(a) of Algorithm 2.

Compared to Clifford operators, simplification via Proposition 2.9 may not always leave the path sum in a reducible state. For instance, the path sum expression below, corresponding to a unitary transformation, is fully reduced with respect to Equations (2), (3) and (4):

$$\frac{1}{\sqrt{2^2}} \sum_{y_1,y_2} i^{x_2y_1 - x_2y_2} (-1)^{x_1y_1 + x_1y_2 + x_2y_1y_2} \left|y_1\right\rangle \left|y_2\right\rangle$$

However, neither y_1 nor y_2 are reducible due to the quadratic terms x_2y_1 and x_2y_2 in the exponent of i. At the moment, it is unclear how to proceed symbolically to find a generalized permutation that will make either path variable reducible in the above expression.

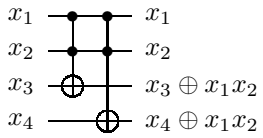
Our heuristic method of producing a generalized permutation proceeds in increasingly costly circuit stages in an attempt to synthesize as efficient circuits as possible. Rather than attempt to synthesize a distinct generalized permutation for every path variable y as described in Algorithm 2, we first apply a sequence of simplification stages generically to both reduce the redundant synthesis work, and produce simpler circuits in practice. The sequence of stages is given in Algorithm 3, and the individual synthesis steps are described in detail below.

Algorithm 3 Generalized permutation synthesis heuristic

1. Apply affine simplifications to the output state $|f(\vec{x},\vec{y})\rangle$
2. Apply non-linear simplifications to the phase $e^{2\pi i P(\vec{x},\vec{y})}$
3. Apply non-linear simplifications to the output state $|f'(\vec{x},\vec{y})\rangle$
4. Apply non-linear simplifications to the phase $e^{2\pi i P'(\vec{x},\vec{y})}$
5. If no path variable is reducible, attempt degree reduction on each variable

Affine simplifications As X and $CNOT$ gates are relatively inexpensive, the first stage of our generalize permutation synthesis attempts to simplify the output $|f(\vec{x},\vec{y})\rangle$ of the path sum as much as possible using only these affine transformations. In order to reduce the number of high-degree terms, which would otherwise require expensive multiply-controlled Toffoli gates, we perform affine simplifications on a *linearization* of f. Specifically, we write each $f_i(\vec{x},\vec{y})$ as a sparse vector $\vec{u_i} \in \mathbb{Z}_2^{2^{n+k}}$ using reverse lexociographic order for the encoding of monomials, then set $A = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix}^T$ and use Gaussian elimination to compute a sequence of $CNOT$ gates reducing A to echelon form. The example below illustrates our method.

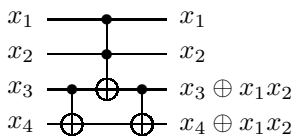
Example A.1. Consider the path sum $|x_1\rangle\,|x_2\rangle\,|x_3 \oplus x_1x_2\rangle\,|x_4 \oplus x_1x_2\rangle$. This could naturally by synthesized using two non-linear Toffolis to eliminate the x_1x_2 terms from the third and forth qubits. The resulting circuit is shown below:



Alternatively, we can write the output as a (sparse) linear system over all monomials in x_1, x_2, x_3, x_4 as shown below:

	x_1x_2	x_4	x_3	x_2	x_1
x_1	0	0	0	0	1
x_2	0	0	0	1	0
$x_3 \oplus x_1x_2$	1	0	1	0	0
$x_4 \oplus x_1x_2$	1	1	0	0	0

We use reverse lexicographic order so that reduction to echelon form will prioritize the number of high degree terms. Reducing this to echelon form results in a single $CNOT$ gate and reduces the state to $|x_1\rangle\,|x_2\rangle\,|x_3 \oplus x_1x_2\rangle\,|x_4 \oplus x_3\rangle$. Synthesizing this remaining transformation gives the overall circuit



Phase simplifications To reduce and simplify the number of terms in the phase $e^{2\pi i P(\vec{x},\vec{y})}$ of a path sum controlled R_Z gates with continuous parameters are used. In particular, given an n-dimensional path sum $e^{2\pi i\theta \prod_i x_i} |\vec{x}\rangle$, we can reduce the phase term by applying a $\Lambda_n(R_Z(-\theta))$ gate, since

$$\Lambda_n(R_Z(-\theta)) : e^{2\pi i\theta \prod_i x_i} |\vec{x}\rangle \mapsto |\vec{x}\rangle .$$

As the output of the path sum is in some state $|f(\vec{x},\vec{y})\rangle$, to apply the above rule we first need to apply a *change of frame* by setting $f_i(\vec{x},\vec{y}) = z_i$ and writing the phase polynomial $P(\vec{x},\vec{y})$ as $P'(\vec{x},\vec{y},\vec{z})$. This is achieved by, for each f_i, letting l be the *largest* (non-zero) degree term of f_i and substituting $l \leftarrow z_i \oplus l \oplus f_i(\vec{x},\vec{y})$ in the path sum.

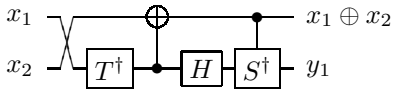
Example A.2. Consider the irreducible path sum $\frac{1}{\sqrt{2}} \sum_{y_1} \omega^{-x_1} i^{x_1 y_1 - x_2 y_1} (-1)^{x_1 x_2 y_1} |x_1 \oplus x_2\rangle |y_1\rangle$. Substituting $[x_2 \leftarrow z_1 \oplus x_1,\ y_1 \leftarrow z_2]$ gives the re-framed path sum

$$\omega^{-x_1} i^{-z_1 z_2} (-1)^{x_1 z_2} |z_1\rangle |z_2\rangle .$$

Applying a controlled S gate to eliminate the term $i^{-z_1 z_2}$ and rolling back the substitutions gives

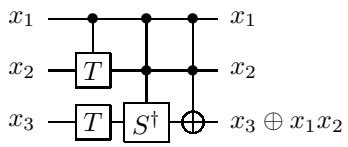
$$\omega^{-x_1} (-1)^{x_1 y_1} |x_1 \oplus x_2\rangle |y_1\rangle .$$

The variable y_1 is now reducible, so we can finish synthesis by applying a Hadamard to the second qubit, then synthesizing the final generalized permutation $|x_1 x_2\rangle \mapsto \omega^{-x_1} |x_1 \oplus x_2\rangle |x_1\rangle$. The resulting circuit is given below:



In our implementation, we apply phase simplifications both before and after non-linear simplifications in the state. This is so that we can effectively utilize high degree terms in the state to simplify high degree terms in the phase with phase gates on fewer qubits. The following example illustrates this effect.

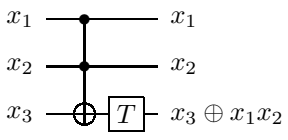
Example A.3. Consider the path sum $\omega^{x_3 + x_1 x_2} i^{-x_1 x_2 x_3} |x_1\rangle |x_2\rangle |x_3 \oplus x_1 x_2\rangle$. Eliminating the $x_1 x_2$ term in qubit 3 before simplifying the phase results in the following circuit:



However, by re-framing the sum with the substitution $[x_1 \leftarrow z_1, x_2 \leftarrow z_2, z_1 z_2 \leftarrow z_3 \oplus x_3]$ we find

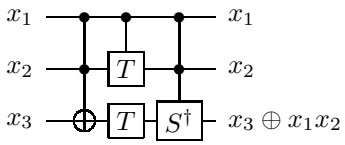
$$\omega^{x_3 + x_1 x_2} i^{-x_1 x_2 x_3} |x_1\rangle |x_2\rangle |x_3 \oplus x_1 x_2\rangle \equiv \omega^{z_3} |z_1\rangle |z_2\rangle |z_3\rangle$$

which can now be simplified with a single T gate. Note that the substitution is applied left to right, rather than as a simultaneous substitution. The resulting circuit is shown below:



The final substitution $z_1 z_2 \leftarrow z_3 \oplus x_3$ may seem counter-intuitive, as we could instead substitute $x_3 \leftarrow z_3 \oplus z_1 z_2$. We choose a monomial of maximal degree to substitute in order to avoid inadvertently increasing

the degreee of the phase polynomial. For instance, if the initial path sum was instead $\omega^{x_3} |x_1\rangle |x_2\rangle |x_3 \oplus x_1x_2\rangle$, substituting $x_3 \leftarrow z_3 \oplus z_1z_2$ results in a re-framed sum of $\omega^{z_3+z_1z_2} i^{-z_1z_2z_3} |z_1\rangle |z_2\rangle |z_3\rangle$ and the final circuit



By substituting the highest degree monomial instead, we avoid this issue and synthesize the simpler circuit placing the T gate to the left of the Toffoli.

Non-linear simplifications The non-linear simplification step of our synthesis algorithm reduces the number of non-linear terms in the output $|f(\vec{x}, \vec{y})\rangle$ by applying multiply-controlled Toffoli gates $\Lambda_k(X)$ via the rule

$$\Lambda_k(X) : |\vec{x}\rangle |f \oplus \prod_i x_i\rangle \mapsto |\vec{x}\rangle |f\rangle .$$

Our method for non-linear simplifications uses a naïve heuristic whereby a set of variables

$$V = \{v \mid f_i(\vec{x}, \vec{y}) = v \text{ for some } i\}$$

is computed. Any term in $f(\vec{x}, \vec{y})$ which is a product of variables contained in V is then eliminated with an appropriately controlled Toffoli gate.

This method is far from optimal, and in particular misses cases which can be factorized as a cascade of Toffoli gates. While better reversible synthesis methods exist, the lack of a known permutation to synthesize *a priori* in our case makes it difficult to apply such methods directly. An interesting avenue for future work would be to re-synthesize the permutation discovered through this process of simplification using state-of-the-art methods.

Degree reduction In many cases, the simplifications previously described fail to leave some path variable in a reducible position. When this happens, our last resort is to fall back to an exponential time procedure we call *degree reduction*. The idea of is to reduce the degree of the (non-Boolean) parts of a phase polynomial restricted to a particular variable, as these terms serve as roadblocks for reduction. This can in some cases be accomplished by applying variable substitutions in such a way as to cancel out terms involving a particular variable.

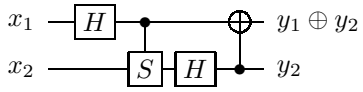
To illustrate degree reduction, recall the irreducible path sum expression from the beginning of this section,

$$\frac{1}{\sqrt{2^2}} \sum_{y_1,y_2} i^{x_2y_1 - x_2y_2} (-1)^{x_1y_1 + x_1y_2 + x_2y_1y_2} |y_1\rangle |y_2\rangle$$

The exponent of i cannot be directly reduced via simple phase simplifications, as both terms depend on x_2. However, we can *indirectly* eliminate one of these terms by applying Equation (5), substituting y_1 with $y_1 \oplus y_2$:

$$\begin{aligned}
&\frac{1}{\sqrt{2^2}} \sum_{y_1,y_2} i^{x_2y_1 - x_2y_2} (-1)^{x_1y_1 + x_1y_2 + x_2y_1y_2} |y_1\rangle |y_2\rangle \\
&\quad = \frac{1}{\sqrt{2^2}} \sum_{y_1,y_2} i^{x_2(y_1 \oplus y_2) - x_2y_2} (-1)^{x_1(y_1 \oplus y_2) + x_1y_2 + x_2(y_1 \oplus y_2)y_2} |y_1 \oplus y_2\rangle |y_2\rangle \qquad \text{by Equation (5)} \\
&\quad = \frac{1}{\sqrt{2^2}} \sum_{y_1,y_2} i^{x_2y_1} (-1)^{x_1y_1 + x_2y_2} |y_1 \oplus y_2\rangle |y_2\rangle
\end{aligned}$$

The final expression above can be simplified to $\frac{1}{\sqrt{2^2}}\sum_{y_1,y_2} i^{x_2y_1}(-1)^{x_1y_1+x_2y_2}\left|y_1\right\rangle\left|y_2\right\rangle$ by applying a $CNOT$ gate, which leaves y_2 in a reducible position. The resulting circuit is given below:



The above example is relatively easy to spot, but more complicated cases may require substitution of multiple variables, or even non-linear substitutions. Our heuristic revolves around computing a type of *cover* for the quotient $2(P/y)$, where y is the candidate for degree reduction.

Lemma A.4. *Let $P \in \mathbb{R}[y, x_1, \ldots, x_n]$ such that $4(P/y) \equiv 0 \mod 2$. If there exists a set $S \subseteq \{1, \ldots, n\}$ such that $4(P/x_i) \equiv 0 \mod 2$ for all $i \in S$ and*

$$\sum_{i\in S} 2(P/x_i) \equiv 2(P/y) \mod 2$$

then $2(P[x_i \leftarrow \overline{x_i \oplus y} \mid i \in S]/y) \equiv 0 \mod 2$

Proof. First recall that $\overline{x \oplus y} = x + y - 2xy$. Hence

$$2P[x_i \leftarrow \overline{x_i \oplus y} \mid i \in S] = 2P + \sum_{i\in S} 2y(P/x_i) + \sum_{i\in S} 4x_iy(P/x_i)$$

Taking the quotient by y we see

$$\begin{aligned} 2(P[x_i \leftarrow \overline{x_i \oplus y} \mid i \in S]/y) &= 2(P/y) + \sum_{i\in S} 2(P/x_i) + \sum_{i\in S} 4x_i(P/x_i) \\ &\equiv 2(P/y) + 2(P/y) \mod 2 \\ &\equiv 0 \mod 2 \end{aligned}$$

□

In the context of path sums, Lemma A.4 tells us that if $e^{2\pi iP(\vec{x},\vec{y})}$ can be written as $i^{y_iQ(\vec{x},\vec{y})}e^{2\pi iR(\vec{x},\vec{y})}$ for some i, and there exists a subset of path variables $\{y_j \mid j \neq i\}$ such that $e^{2\pi iP(\vec{x},\vec{y})} = i^{y_jQ_j(\vec{x},\vec{y})}e^{2\pi iR_j(\vec{x},\vec{y})}$ and $i^{\sum_j Q_j(\vec{x},\vec{y})} = i^{Q(\vec{x},\vec{y})}$, then the simultaneous substitution $y_j \leftarrow y_j \oplus y_i$ will eliminate the term $i^{y_iQ(\vec{x},\vec{y})}$. Additional simplifications in the state may then be further required to leave the path sum in a reducible state, as in the previous above.