Reducing the number of non-Clifford gates in quantum circuits

Aleks Kissinger[1,*] and John van de Wetering[2,†]
[1]*Department of Computer Science, University of Oxford, 15 Parks Rd, Oxford OX1 3QD, United Kingdom*
[2]*Radboud University, Tournooiveld 214, 6525EC Nijmegen, Netherlands*

(Received 6 May 2019; revised 3 July 2020; accepted 24 July 2020; published 11 August 2020)

We present a method for reducing the number of non-Clifford quantum gates, in particularly T-gates, in a circuit, an important task for efficiently implementing fault-tolerant quantum computations. This method matches or beats previous approaches to ancillae-free T-count reduction on the majority of our benchmark circuits, in some cases yielding up to 50% improvement. Our method begins by representing the quantum circuit as a ZX-diagram, a tensor networklike structure that can be transformed and simplified according to the rules of the ZX-calculus. We then extend a recent simplification strategy with a different ingredient, phase gadgetization, which we use to propagate non-Clifford phases through a ZX-diagram to find nonlocal cancellations. Our procedure extends unmodified to arbitrary phase angles and to parameter elimination for variational circuits. Finally, our optimization is self-checking, in the sense that the simplification strategy we propose is powerful enough to independently validate equality of the input circuit and the optimized output circuit. We have implemented the routines of this paper in the open-source library PyZX.

DOI: 10.1103/PhysRevA.102.022406

I. INTRODUCTION

Quantum circuits give a simple, universal language for describing quantum computations at a low level. When studying circuits it is often useful to distinguish between two kinds of primitive operations: Clifford gates and non-Clifford gates. Circuits consisting only of Clifford gates can be efficiently classically simulated [1], and can be implemented in a fault-tolerant manner with relative ease within many quantum error correcting codes [2,3]. However, for universal circuits at least one type of non-Clifford gate, such as the T gate, is needed. While techniques such as magic state distillation and injection allow for fault-tolerant implementation of T gates, they typically require a great deal more resources than Clifford gates [4]. Hence, efficient fault-tolerant quantum computation requires minimizing the number of non-Clifford gates within a circuit.

Existing methods for computing exact-optimal T-counts take exponential running time [5,6]. To date, the most successful scalable approaches to T-count minimization have been based on *phase polynomials*. Such methods rely on an efficient representation of circuits consisting of just CNOT and Z-phase gates. The first heuristic method for efficiently reducing T-count and T-depth using this representation, called *T-par*, was introduced in Ref. [7]. Their results were later improved upon in Refs. [8,9] by exploiting equivalences between phase polynomial optimization and other known problems.

Phase-polynomial methods share the limitation that they cannot deal directly with arbitrary quantum circuits. In particular, an arbitrary circuit will also contain Hadamard gates, which destroy the phase polynomial structure. Naïvely, one can simply cut the circuit into Hadamard-free sections and apply the optimization locally. This can be significantly improved by preprocessing to produce larger Hadamard-free sections: either by simple gate transformations [10,11] or introducing ancillae and classical control [9].

In this paper, we propose a different approach to reducing non-Clifford gate count based on the theoretical framework laid out in Ref. [12]. We first transform a circuit into a special kind of tensor network called a ZX-diagram [13,14]. This diagram is then subject to a collection of graphical transformation rules called the ZX-calculus [15]. By breaking the rigid circuit structure, ZX-diagrams are then subject to simplifications that have no obvious circuit analog.

It was noted in Ref. [12] that non-Clifford phases (i.e., angles which are not multiples of $\pi/2$) form an obstruction to the simplification. To overcome this issue, we introduce one crucial refinement to the simplification procedure: the *gadgetization* of non-Clifford phases. By splitting a node containing a phase into two parts consisting of the node itself, and a new *phase gadget*, phases can propagate nonlocally through a ZX-diagram and potentially cancel or combine with each other. In the case where there are no Hadamard gates in the circuit, these gadgets correspond to phase-parity terms in the representation of a phase polynomial, hence this nonlocal propagation can be seen as a generalisation of phase polynomial techniques to general circuits.

After diagrammatic simplification, we could, in principle, use a variation on the technique described in Ref. [12] to re-extract a quantum circuit from the ZX-diagram with fewer non-Clifford phases. However, we show here that we can sidestep the need for circuit extraction all together by exploiting the fact that our simplification method is parametric in non-Clifford phases. Rather than combining two phase gadgets

*aleks.kissinger@cs.ox.ac.uk
†john@vdwetering.name

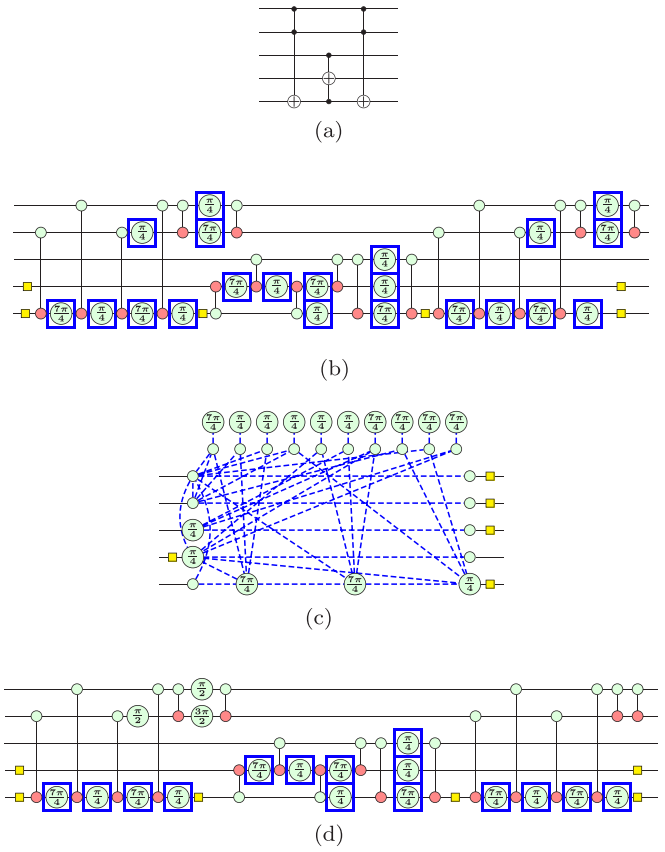


FIG. 1. Overview of the steps in our phase-teleportation scheme. (a) Original circuit. (b) The circuit expanded as a ZX diagram, with 21 T gates. (c) Simplified ZX-diagram. (d) 15 t gates remain after phase teleportation.

into one, we let the angle from one phase gadget jump onto the other one: $(\alpha_i, \alpha_j) \rightsquigarrow (\alpha_i + \alpha_j, 0)$. This does not have any effect on the graphical structure of the ZX-diagram, and hence performing this modification to the phases of the original circuit will result in a new circuit that reduces to the same ZX-diagram as before.

Hence, rather than re-extracting a circuit from a ZX-diagram, we use the diagram as a tool for discovering phases that can be shifted around nonlocally without changing the computed unitary. We call this technique *phase teleportation*. A pleasant property of phase teleportation, as opposed to the simplify-and-extract method, is that it leaves the structure of the quantum circuit completely intact, only changing the parameters. Hence, 2-qubit gate count is never increased and gates are always applied between the same pairs of qubits as before. As pointed out in Ref. [11], this could be advantageous when the circuit has been designed with limited qubit connectivity of the physical qubits in mind. Both optimisation routines, using either extraction or phase teleportation, are implemented in the open source Python library PyZX [16].

A high-level overview of our procedure is presented in Fig. 1. The four steps presented there are described in detail in Sec. II. We start with some circuit as in Fig. 1(a). By expanding the gates in terms of CNOT, H, and T gates, we can easily translate the circuit into a ZX-diagram [Fig. 1(b)]. We then apply the simplification procedure described in Sec. II D to obtain a reduced ZX-diagram [Fig. 1(c)]. Finally, we use the data about corresponding phases obtained from this simplification to perform *phase teleportation* in the original circuit to reduce T-count [Fig. 1(d)].

By leaving the circuit model we can sometimes "look around" obstructions such as Hadamard gates to find more optimizations. We see this translated in our results. In benchmark circuits with an abundance of Hadamard gates we can significantly outperform previous methods.

Our simplification routine can also validate equality of circuits. We do this by composing the adjoint of the optimised circuit with the original circuit and checking whether our simplification routine reduces the resulting ZX-diagram to the identity. While this method cannot detect errors in a circuit, the set of rewrite rules forms a certificate of equality when it does reduce a circuit to the identity. A general efficient circuit equality validation schema is unlikely to exist [17]. Nevertheless, our method is powerful enough to validate correctness of our optimised circuits as well as those produced in Ref. [11].

We target *ancilla-free* optimization and compare ourselves to the best known results for ancilla-free T-count reduction. When ancillas are allowed, the required amount of T gates can decrease [7]. We discuss the possibility of using our methods for ancillae-based optimizations in Sec. IV.

II. METHODS

In this section we will explain our main contributions in depth, namely how to do T-count optimization using the ZX-calculus. On a high level this proceeds in the following way:

(1) First we translate a quantum circuit into a ZX-diagram. See Sec. II A.

(2) We apply the diagrammatic simplification procedure *ZX-simplify* laid out in Secs. II B-II D.

(3) Finally, by keeping track of certain simplification steps, and how they affect phases in the original circuit, we will decrease the T-count of the circuit by means of *phase teleportation*. See Sec. II E.

Section II F explains our how our PyZX-produced circuit is combined with postprocessing and the TODD compiler.

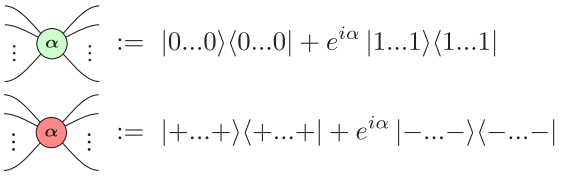
A. Background: the ZX-calculus

We will provide a brief overview of the ZX-calculus. For an in-depth reference see Ref. [18].

The ZX-calculus is a diagrammatic language similar to the familiar quantum circuit notation. A *ZX-diagram* (or simply *diagram*) consists of *wires* and *spiders*. Wires entering the diagram from the left are *inputs*; wires exiting to the right are *outputs*. Given two diagrams we can compose them by joining the outputs of the first to the inputs of the second, or form their tensor product by simply stacking the two diagrams.

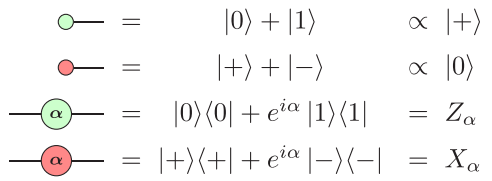
Spiders are linear operations which can have any number of input or output wires. There are two varieties: Z spiders

depicted as green dots and X spiders depicted as red dots:



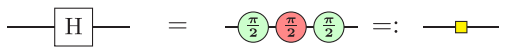
Note that if you are reading this document in monochrome or otherwise have difficulty distinguishing green and red, Z spiders will appear lightly shaded and X darkly shaded. The diagram as a whole corresponds to a linear map built from the spiders (and permutations) by the usual composition and tensor product of linear maps. As a special case, diagrams with no inputs represent (unnormalized) state preparations.

Example II.1. We can immediately write down some simple state preparations and unitaries in the ZX-calculus:



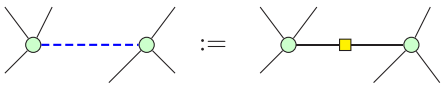
In particular, we have the Pauli matrices:

It will be convenient to introduce a symbol—a yellow square, - for the Hadamard gate. This is defined (up to a global phase) by the equation:



(1)

We will often use an alternative notation to simplify the diagrams, and replace a Hadamard between two spiders by a blue dashed edge, as illustrated below:



Both the blue edge notation and the Hadamard box can always be translated back into spiders when necessary. We will refer to the blue edge as a *Hadamard edge*.

Two diagrams are considered *equal* when one can be deformed to the other by moving the vertices around in the plane, bending, unbending, crossing, and uncrossing wires, as long as the connectivity and the order of the inputs and outputs is maintained. Equivalently, a ZX-diagram can be considered as a graphical depiction of a tensor network, as in, e.g., Ref. [19]. Then, just like any other tensor network, we can observe that the interpretation of a ZX-diagram is unaffected by deformation. As tensors, Z- and X-spiders can be written as follows:

$$\left(\ \alpha\ \right)^{j_1\ldots j_n}_{i_1\ldots i_m} = \begin{cases} 1 & \text{if } i_1 = \ldots = i_m = j_1 = \ldots = j_n = 0 \\ e^{i\alpha} & \text{if } i_1 = \ldots = i_m = j_1 = \ldots = j_n = 1 \\ 0 & \text{otherwise} \end{cases}$$

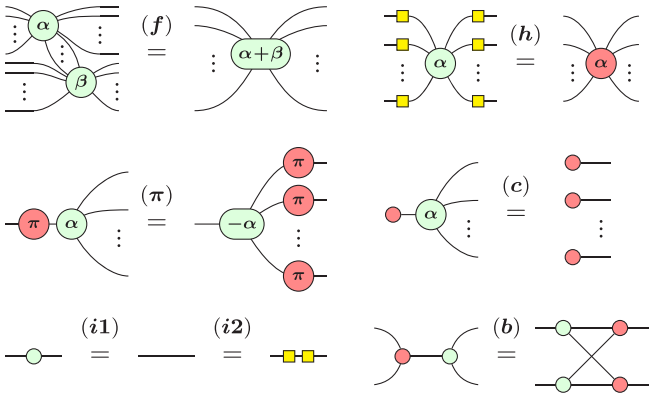


FIG. 2. A convenient presentation for the ZX-calculus. These rules hold for all $\alpha, \beta \in [0, 2\pi)$, and due to (**h**) and (**i2**) all rules also hold with the colors interchanged.

One can then define X-spiders as Z-spiders conjugated by Hadamard gates or define them explicitly as follows:

$$\left(\ \alpha\ \right)^{j_1\ldots j_n}_{i_1\ldots i_m} = \frac{1}{\sqrt{2}} \cdot \begin{cases} 1 + e^{i\alpha} & \text{if } \bigoplus_\alpha i_\alpha \oplus \bigoplus_\beta j_\beta = 0 \\ 1 - e^{i\alpha} & \text{if } \bigoplus_\alpha i_\alpha \oplus \bigoplus_\beta j_\beta = 1 \end{cases}$$

where i_α, j_β range over $\{0, 1\}$ and $\oplus$ is addition modulo 2.

In addition to simple deformations, ZX-diagrams satisfy a set of equations called the ZX-calculus. There exists several variations of the ZX-calculus. The set of rules we will use is presented in Fig. 2. Diagrams that can be transformed into each other using the rules of the ZX-calculus correspond to equal linear maps (up to normalisation). ZX-diagrams with arbitrary angles are expressive enough to represent any linear map [14]. It is often useful to consider *Clifford ZX-diagrams*, by analogy to Clifford circuits, where all angles are restricted to multiples of $\pi/2$. In that case, the rules given in Fig. 2 are *complete* with respect to linear map equality [15]. That is, if two Clifford ZX-diagrams describe the same linear map, one can be transformed into the other using the rules in Fig. 2. Extensions of the calculus exist that are complete for larger families of ZX-diagrams/maps, including *Clifford+T* ZX-diagrams [20], where phases are multiples of $\pi/4$, and arbitrary ZX-diagrams where any phase is allowed [21–23].

Quantum circuits can be translated into ZX-diagrams in a straightforward manner. We will take as our starting point circuits constructed from the following universal set of gates:

$$\text{CNOT} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{2}$$

$$Z_\alpha := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix},$$

$$H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Note that the CNOT gate and the H gate are Clifford, and hence our goal is to reduce the total number of gates of the form Z_α where $\alpha \neq k \cdot \frac{\pi}{2}$ for $k \in \mathbb{Z}$.

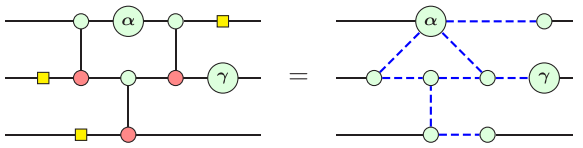
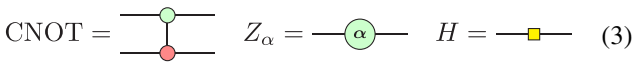


FIG. 3. A ZX-diagram that comes from a circuit, and its equivalent graph-like ZX-diagram.

This gate set admits a convenient representation in terms of spiders:

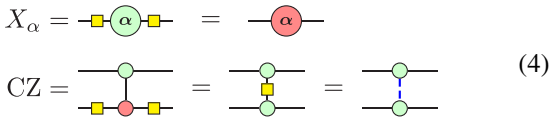


(3)

Note that for the CNOT gate, the green spider is the first (i.e., control) qubit and the red spider is the second (i.e., target) qubit.

The ZX-diagram representing the CNOT is actually only equal up to a scalar factor of $\sqrt{2}$. When converting a quantum circuit into a ZX-diagram we can save these global scalar factors separately so that the linear map represented by the ZX-diagram is equal to the circuit. However, for a ZX-diagram representing a unitary quantum circuit, the correct scalar factor can always be easily inferred, so we are warranted in ignoring this factor.

Other common gates can easily be expressed in terms of these gates. In particular, $S := Z_{\frac{\pi}{2}}$, $T := Z_{\frac{\pi}{4}}$ and



(4)

The first step of our simplification procedure is to transform the circuit into a *graphlike* ZX-diagram [12].

Definition II.2. A ZX-diagram is *graphlike* when

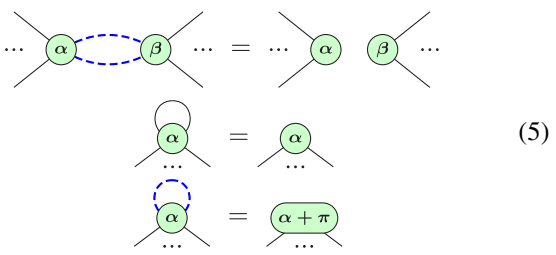
(1) All spiders are Z-spiders.

(2) Z-spiders are only connected via Hadamard edges.

(3) There are no parallel Hadamard edges or self-loops.

(4) Every input or output is connected to a Z-spider and every Z-spider is connected to at most one input or output.

See Fig. 3 for an example. In Ref. [12] it is shown that any ZX-diagram can efficiently be transformed into a graphlike ZX-diagram using the rules of the ZX-calculus. This transformation essentially amounts to turning all X spiders into Z spiders with the (**h**) rule, fusing as many spiders together as possible with (**f**), and removing parallel edges/self-loops with the following derived rules:



(5)

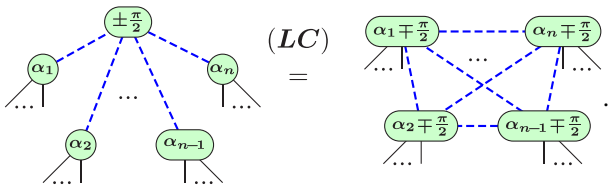
In particular, the number of non-Clifford phases in a diagram is never increased, and can actually be decreased by the (f) rule, as phases are added together. We call this *graphlike* because the resulting ZX-diagram is essentially an indirected, simple graph whose vertices are labeled by phase angles.

B. Clifford simplification of ZX-diagrams

A spider connected to an input or an output is called a *boundary spider*, whereas all other spiders are called *interior spiders*. If we interpret an N-qubit circuit as a ZX-diagram, there are precisely N inputs and N outputs, hence there are at most $2N$ boundary spiders. On the other hand, there will in general be a large number of interior spiders.

The main idea behind the first part of our simplification strategy is to remove as many interior *Clifford spiders*, i.e., spiders whose phase is a multiple of $\pi/2$, as possible. We do this by using two rewrite rules based on the graph-theoretic operations of *local complementation* and *pivoting*. For the proof of correctness of these rewrite rules we refer to Ref. [12].

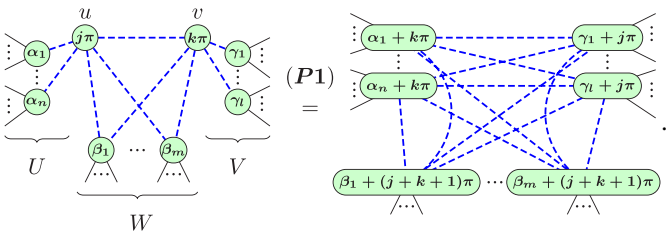
The first rule, based on local complementation, deletes a spider with a phase of $\pm\pi/2$ and introduces edges between all of its neighbours:



(6)

Since parallel edges vanish [cf., Eq. (5)], this can be seen as complementing the set of edges connecting the neighbours of the deleted vertex, hence the name.

The second rule deletes pairs of *Pauli spiders*, i.e., spiders whose phase is a multiple of π. For a pair of connected Pauli spiders u, v, we can split the neighbourhood of $\{u, v\}$ into three pieces: U the spiders only connected to u, V the spiders only connected to v, and W, the spiders connected to both. We can then delete the pair of spiders u, v provided we introduce complete bipartite graphs on (U, W), (V, W), and (U, V):



(7)

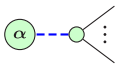
Again, thanks to Eq. (5), this can be seen as complementing the sets of edges present in the three bipartite graphs (U, W), (V, W), and (U, V).

Since the rules (**LC**) and (**P1**) both delete at least one spider, we can simply apply them repeatedly until no rule matches. This gives us a terminating procedure for simplifying our diagram. Note that we do not target the spiders in any specific order. Different orders of application will yield different diagrams (i.e., these rules are not *confluent*), but we always obtain the same amount of non-Clifford spiders at the end.

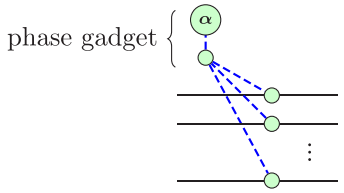
At this point, the simplification procedure in Ref. [12] employs a variation of (**P1**) to remove a few more Pauli spiders and terminates. In particular, nothing is done to eliminate *non-Clifford* spiders. This is the goal of the next two sections.

C. Phase gadgets

We first introduce a useful concept for ZX-diagrams: a *phase gadget*. A phase gadget is simply an arity-1 spider with angle α, connected via a Hadamard edge to a spider with no angle:



Phase gadgets are a useful tool for working with ZX-diagrams corresponding to unitaries. For example, the diagram

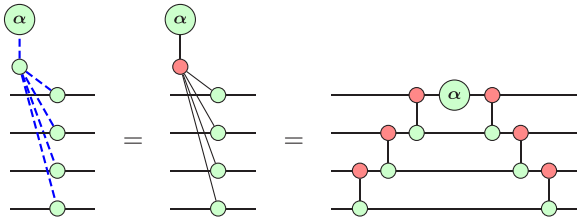


(8)

yields an n-qubit unitary U defined by

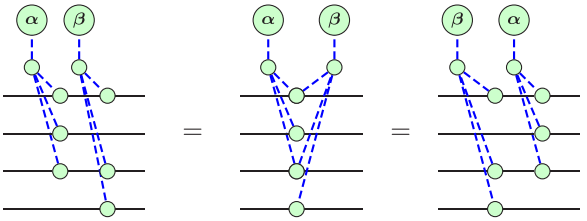
$$U :: |x_1, ..., x_n\rangle \mapsto e^{i\alpha(x_1 \oplus ... \oplus x_n)} |x_1, ..., x_n\rangle.$$

In fact, it is straightforward to show concretely (or in the ZX-calculus) that this unitary is equal to a ladder of CNOT gates, followed by a single phase gate, followed by the reverse ladder of CNOT gates. For example, on 4 qubits:



(9)

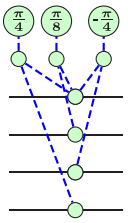
Since these gates are diagonal in the computational basis, they commute with each other. This also follows straightforwardly from the (**f**) rule:



Arbitrary diagonal unitaries, i.e., unitaries of the form

$$U :: |x_1, ..., x_n\rangle \mapsto e^{if(x_1,...,x_n)} |x_1, ..., x_n\rangle$$

for some $f : \{0, 1\}^n \to \mathbb{R}$, can easily be expressed in terms of phase gadgets. For example, for $f(x_1, x_2, x_3, x_4) = \frac{\pi}{4} x_1 \oplus x_4 + \frac{\pi}{8} x_1 \oplus x_2 - \frac{\pi}{4} x_1 \oplus x_3$ we get



In fact, the angles appearing in the phase gadgets correspond to the Fourier expansion of the semi-boolean function f. That is, any function $f : \{0, 1\}^n \to \mathbb{R}$ can be expressed as follows:

$$f(\vec{x}) = \alpha + \sum_{\vec{y}} \alpha_{\vec{y}} (x_1 y_1 \oplus \ldots \oplus x_n y_n), \tag{10}$$

where $\vec{x}, \vec{y} \in \{0, 1\}^n$ and $\alpha, \alpha_{\vec{y}} \in \mathbb{R}$. In the context of diagonal unitaries, α yields a global phase (which we ignore), and each $\alpha_{\vec{y}}$ corresponds to a phase gadget. A brief discussion of the form (10), and its relation to the usual Fourier transform of a semi-boolean function, can be found in the Appendix of Ref. [24]

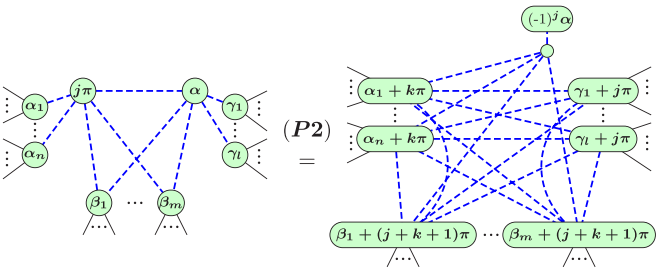
Phase polynomial techniques perform transformations on the function f in order to reduce the T-count needed to implement U (or some U' that is Clifford-equivalent to U). One of these methods is *TODD* [9], which we use as a postprocessing step for our algorithm (cf. Section II F). In the sequel, we will consider not just phase gadgets arising from unitaries such as Eq. (8), but phase gadgets appearing in arbitary graphlike ZX-diagrams. Hence, our simplification procedure can be seen as a generalization of phase polynomial techniques.

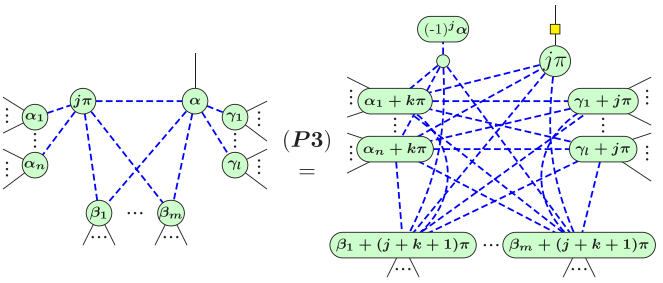
D. Full simplification of ZX-diagrams

In this section, we will introduce rules that reduce the number of non-Clifford spiders in the ZX-diagram, and hence the T-count in the resulting circuit.

First, it is worth noting that the (**P1**) rule from section II B was only able to remove an interior Pauli spider adjacent to another interior Pauli spider. We can now introduce two variations of this rule, (**P2**) and (**P3**), which together allow us to remove any remaining interior Pauli spider, at the cost of

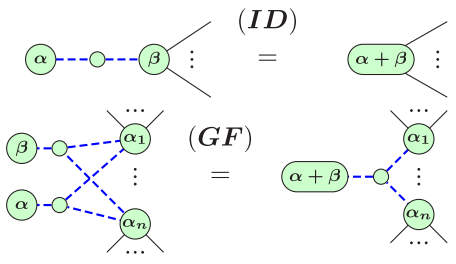
introducing a phase gadget:





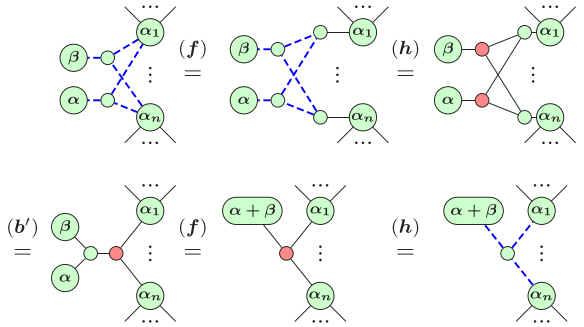
We apply (**P2**) when the interior Pauli spider is connected to any other interior spider, while (**P2**) is applied when it is connected to some boundary spider. Applying these rules to every remaining interior Pauli spider yields a diagram where every internal spider is either non-Clifford or part of a phase gadget. If the phase gadget is Clifford, then it can be removed by either (**P1**) or by two applications of (**LC**). Hence we can reduce to a case where all phase gadgets are non-Clifford.

We can now apply the following two rules, which both strictly decrease the number of non-Clifford spiders:



When a phase gadget is connected to exactly one other spider, its phase can be combined with the phase on that spider via (**ID**). This is essentially an application of the rules (**i1**) and (**i2**).

When two phase gadgets are connected to exactly the same set of spiders, they can be fused into one via the gadget-fusion rule (**GF**). This rule can be shown using the ZX-calculus:



where (**b′**) is the n-ary generalization of the rule (**b**), which follows from the other rules (see, e.g., [18], Sec. 9.4). For unitaries of the form (8), it corresponds to a well-known simplification used in phase-polynomial circuits, where two phases acting on the same parity of the input qubits can be summed together.

Each of the rewrite rules (**ID**) and (**GF**) removes a non-Clifford spider, and transforms another non-Clifford spider into a Clifford spider, which can be removed by one of the previous rules. We can now fully describe our simplification procedure for graphlike ZX-diagrams.

Algorithm II.3. ZX-simplify: Starting with a graph-like ZX-diagram, do the following:

(1) Apply (**LC**) until all interior proper Clifford vertices are removed.

(2) Apply (**P1**), (**P2**), and (**P3**) until all interior Pauli vertices are removed or transformed into phase gadgets.

(3) Remove all Clifford phase gadgets using (**LC**) and (**P1**).

(4) Apply (**ID**) and (**GF**) wherever possible. If any matches were found, go back to step 1, otherwise we are done.

This algorithm always terminates as every step either removes a spider or a phase gadget. In terms of complexity we see that if the original diagram had n spiders, that this algorithm takes at most n steps. Each step might need us to toggle the connectivity of all the neighbors of the involved spider. As this spider has at most n neighbors, this could involve n^2 operations on the diagram. The complexity of the algorithm is therefore bounded above by $O(n^3)$ elementary graph operations. In practice though, the ZX-diagrams resulting from quantum circuits will be quite sparse, and we tend to see a time scaling roughly between $O(n)$ and $O(n^2)$ on our benchmark circuits.

It will be useful to have a name for the diagrams produced by this simplification procedure.

Definition II.4. We say a graphlike ZX-diagram is in *reduced gadget form* when

(1) Every internal spider is a non-Clifford spider or part of a non-Clifford phase gadget.

(2) Every phase gadget has more than one target.

(3) No two phase gadgets have the same set of targets.

E. Phase teleportation

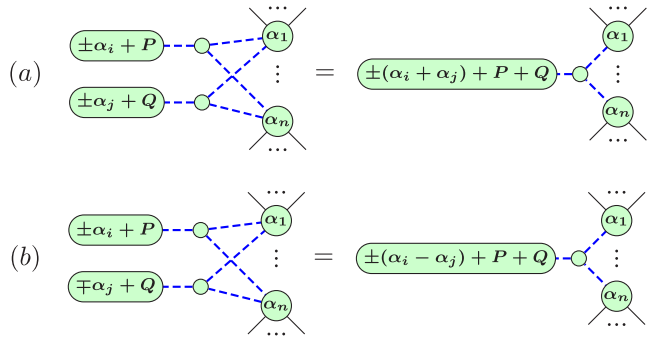
The simplification procedure described in the previous section produces a ZX-diagram that does not look like a circuit. In order to get a different, simplified circuit out, we

could apply (a variation of) the circuit extraction procedure of Ref. [12] as described in Ref. [25]. Alternatively, we can short-circuit the extraction using a trick we refer to as *phase teleportation*.

We begin by replacing every non-Clifford phase in our starting circuit C with a fresh variable name, $\alpha_1, \ldots, \alpha_n$, and storing the angles in a separate table $\tau : \{1, \ldots, n\} \to \mathbb{R}$.

We can then perform the simplification procedure described in the previous section *symbolically*. That is, we work on a ZX-diagram whose spiders are labeled not just with phase angles, but with polynomials over the variables $(\alpha_1, \ldots, \alpha_n)$.

Then, consider what happens when two variables are added together during the (**ID**) and (**GF**) rules. One of two things can occur: (a) the two variables have the same sign or (b) they have different signs:



Since none of our simplifications will copy any of the variables we started with, these are the only occurences of α_i and α_j in the ZX-diagram. Hence, in case (a), if we replace α_i with $\alpha_i + \alpha_j$ and α_j with 0, we get an equivalent diagram.

Put another way, in case (a), we can update our table τ by setting $\tau'(i) := \tau(i) + \tau(j)$, $\tau'(j) := 0$, and $\tau'(k) := \tau(k)$ for $k \notin \{i, j\}$. Then, (C, τ) and (C, τ') describe circuits which are provably equivalent by the rules of ZX-calculus. Case (b) is similar, except we should set $\tau'(i) := \tau(i) - \tau(j)$.

This observation yields the following algorithm:

Algorithm II.5. Phase teleportation: Starting with a circuit, do the following:

(1) Choose unique variables $\alpha_1, \ldots, \alpha_n$ for each non-Clifford phase and store the pair (C, τ), where C is the parametrized circuit and $\tau : \{1, \ldots, n\} \to \mathbb{R}$ assigns each variable to its phase.

(2) Interpret C as a ZX-diagram and run the *ZX-simplify* algorithm on the simplified diagram while doing the following:

> Whenever (**ID**) or (**GF**) are applied to a pair of vertices or phase gadgets containing variables α_i and α_j, respectively, update the phase table τ as described for cases (a) and (b) above.

(3) When *ZX-simplify* finishes, the pair (C, τ') describes an equivalent circuit.

Even though we do compute the reduced gadget form of the circuit C, the new circuit we output has the same structure as C itself, but with some of the phases changed. As a result, no new gates are introduced, but many non-Clifford phase gates will have their angles set to 0 or to multiples of $\pi/2$. Hence, running a dedicated gate minimizing circuit optimisation routine afterwards will often be much more effective.

F. Circuit optimisation and TODD

We now briefly describe a combined optimization routing consisting of first running the phase teleportation procedure, then doing some simple postprocessing, and finally applying the *TODD* algorithm described in Ref. [9].

The circuit postprocessing works by doing forward and backward passes through the circuit. During the forward pass, we commute 1-qubit gates as far forward as possible using standard gate commutation rules, canceling and combining gates whenever we can. We then take the adjoint of the circuit and repeat the process, and keep repeating the process until no more gates are removed.

We then apply the ancilla-free version of the TODD algorithm using the C++ tool Topt [26]. This tool is designed to optimize CNOT+Phase circuits, so we first cut our circuit into Hadamard-free chunks. Then, before running Topt on each chunk, we again use standard gate commutation laws to pull as many gates as possible from neighboring chunks into the current one. Since Topt is nondeterministic, we run it multiple times and we take the best result. Running Topt on each chunk in this manner then yields the T-counts reported in the last column of Table I .

III. RESULTS

In this section we benchmark the algorithms described in the previous section against a suite of benchmark circuits. The goal is reducing the total number of non-Clifford gates, but for all of our benchmark circuits, those gates are all $T := Z_{\pi/4}$, so from hence forth, we will simply refer to the number of non-Clifford gates as the T-count.

For our benchmarks, we have used all of the Clifford+T benchmark circuits from Refs. [7,11] (minus some of the larger members of the `gf(2^n)-mult` family). See Table I. These benchmark circuits were also used in Refs. [8,9] and include components that are likely to be of interest to quantum algorithms, such as adders or Grover oracles. Of these 36 benchmark circuits, we are at or improving upon the best previously known ancilla-free T-count for 26 circuits ($\sim$72%), and we improve on 6 ($\sim$ 17%). If we apply some simple postprocessing afterwards and feed the resulting circuit into the TODD phase polynomial optimiser [9], we improve on the state of the art for 20 circuits ($\sim$ 56%). These two methods seem to complement each other well in the ancilla-free regime, obtaining significantly better numbers than either of the two methods alone, and matching or beating all other methods for every circuit tested.

For 20 of the 36 circuits, we exactly match the best previously known result, which is interesting, since the methods we use are quite different in nature from previous methods. The circuits where PyZX seems to do considerably better are ones that contain many Hadamard gates. The fact that PyZX achieves improvements when there are many Hadamard gates is as expected, as most other successful methods employ a dedicated phase-polynomial optimiszer [7–9,11] that is hampered by the existence of Hadamard gates. On the other hand, the only circuits where phase polynomial techniques significantly outperform our methods are in the `gf(2^n)-mult` family. After some simple preprocessing, these circuits have

TABLE I. Benchmark circuits from [27] and [28]. The columns n and T contain the amount of qubits and T gates in the original circuit. *Best* is the previous best-known ancilla-free T-count for that circuit and *Method* specifies which method was used: RM_m and RM_r refer to the *maximum* and *recursive* Reed-Muller decoder of Ref. [8], *T-par* is Ref. [7], *TODD* is Ref. [9], and *NRSCM* refers to Ref. [11]. *PyZX* and *PyZX+TODD* specify the T-counts produced by, respectively, our method, and our method combined with TODD. Numbers shown in bold are better than previous best, and italics are worse.

Circuit	n	T	Best	Method	PyZX	PyZX +TODD
adder$_8$	**24**	**399**	**213**	**RM$_m$**	**173**	**167**
Adder8	23	266	56	NRSCM	56	56
Adder16	47	602	120	NRSCM	120	120
Adder32	95	1274	248	NRSCM	248	248
Adder64	191	2618	504	NRSCM	504	504
barenco-tof3	5	28	16	T-par	16	16
barenco-tof4	7	56	28	T-par	28	28
barenco-tof5	9	84	40	T-par	40	40
barenco-tof10	19	224	100	T-par	100	100
tof$_3$	5	21	15	T-par	15	15
tof$_4$	7	35	23	T-par	23	23
tof$_5$	9	49	31	T-par	31	31
tof$_{10}$	19	119	71	T-par	71	71
csla-mux$_3$	*15*	*70*	*58*	*RM$_r$*	*62*	**45**
csum-mux$_9$	*30*	*196*	*76*	*RM$_r$*	*84*	**72**
cycle17$_3$	**35**	**4739**	**1944**	**RM$_m$**	**1797**	**1797**
gf(2^4)-mult	*12*	*112*	*56*	*TODD*	*68*	**52**
gf(2^5)-mult	*15*	*175*	*90*	*TODD*	*115*	**86**
gf(2^6)-mult	*18*	*252*	*132*	*TODD*	*150*	**122**
gf(2^7)-mult	*21*	*343*	*185*	*TODD*	*217*	**173**
gf(2^8)-mult	*24*	*448*	*216*	*TODD*	*264*	**214**
ham15-low	17	161	97	T-par	97	97
ham15-med	**17**	**574**	**230**	**T-par**	**212**	**212**
ham15-high	20	2457	1019	T-par	1019	**1013**
hwb$_6$	7	105	75	T-par	75	**72**
hwb$_8$	**12**	**5887**	**3531**	**RM$_{m\&r}$**	**3517**	**3501**
mod-mult-55	*9*	*49*	*28*	*TODD*	*35*	**20**
mod-red-21	11	119	73	T-par	73	73
mod5$_4$	**5**	**28**	**16**	**T-par**	**8**	**7**
nth-prime$_6$	**9**	**567**	**400**	**RM$_{m\&r}$**	**279**	**279**
nth-prime$_8$	*12*	*6671*	*4045*	*RM$_{m\&r}$*	*4047*	**3958**
qcla-adder$_{10}$	36	589	162	T-par	162	**158**
qcla-com$_7$	*24*	*203*	*94*	*RM$_m$*	*95*	**91**
qcla-mod$_7$	26	413	235[a]	NRSCM	237	**216**
rc-adder$_6$	14	77	47	RM$_{m\&r}$	47	47
vbe-adder$_3$	10	70	24	T-par	24	24

[a] An error in a previously reported T-count.

almost no Hadamard gates, hence they are very well-suited to phase polynomial techniques.

It should be noted that while the circuits of Table I are all written in the Clifford+T gate set, our optimization routine is agnostic to the values of the non-Clifford phases. We have also tested our routine on the quantum Fourier transform circuits of Ref. [11] that include more general non-Clifford phases, and in each case found that our non-Clifford gate count exactly matched their results.

The optimization routines are implemented in the open source Python library PyZX [29]. All circuit optimizations were run on a consumer laptop. Our method took a few seconds to run for most circuits, with some of the bigger ones taking up to a few minutes. We tested the correctness of the optimization by generating the matrix of the original and the optimised circuit for thousands of small randomly generated circuits and checking equality, in addition to doing the same for the smaller benchmark circuits.

We can also use the ZX-calculus for verification of equality [30]. For all benchmark circuits, we composed the original circuit with the adjoint of the optimized one, and then ran our simplification routine on this circuit. In every case, the resulting circuit was reduced to the identity. Since the set of rewrites needed to do this reduction is different from the ones used to produce the original optimized circuit, this gives extra confidence that our optimization routine is implemented correctly, as it is very unlikely that an error in our rewrites would cancel itself in this way. With the same validation scheme we have also verified correctness of all the optimized benchmark circuits of Ref. [11], except for `qcla-mod`$_7$, which was then shown to be incorrect using the FEYNMAN tool [31].

IV. CONCLUSION

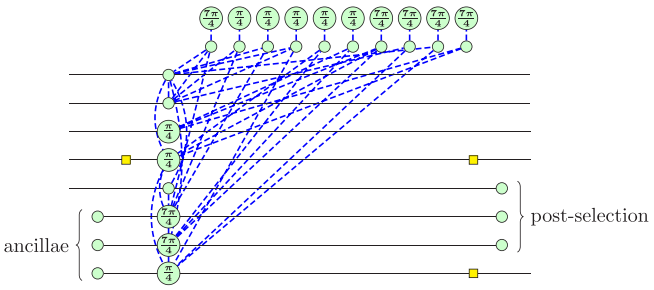
We have implemented a quantum circuit optimization routine that uses ZX diagrams to go beyond the rigid structure of the circuit model. This routine matches or beats the previous best-known T-count for the majority of the benchmark circuits we have tested. We have, furthermore, shown that combining our routine with the TODD compiler [9] achieves T-counts that are better than either of these methods separately. Finally, our simplification routine is powerful enough to validate the correctness of our optimized circuits.

Notably, our simplification is completely parametric in non-Clifford phase angles: phase teleportation treats all non-Clifford angles as free parameters which are simply combined or eliminated. As a consequence our simplification procedure generalizes from concrete circuits to parametrized circuits, where the analog of T-count reduction is elimination of redundant free parameters. This could potentially yield significant improvements in the performance of hybrid classical/quantum algorithms relying on parametrized circuits, such as the quantum variational eigensolver [32].

Our routine can be improved or made more versatile in a variety of ways. Our method currently does not affect the amount of CNOT or Hadamard gates in the circuit. This is because we only use the simplified ZX-diagram to track cancellation of phase gates, instead of using it to extract a new circuit directly. While direct circuit extraction is possible, at this stage such a circuit often contains more Clifford gates than we started out with. In future work, we aim to improve our circuit extraction methods to produce better circuits directly from the ZX-diagrams.

We currently only consider ancilla-free optimization, whereas it has been shown [7,9] that allowing additional ancillae can greatly decrease the required T-count. A promising approach to introducing ancillae into our simplification procedure is by treating the reduced ZX-diagram as a phase polynomial circuit where every non-input corresponds to a

new ancilla in the $|+\rangle$ state and every non-output corresponds to projecting (i.e., post-selecting) onto $\langle+|$. For example, by "unfusing spiders" in the skeleton ZX-diagram depicted in Fig. 1(c), we can obtain a post-selected circuit of the form



The middle part of the diagram can be described as a phase polynomial (cf., Sec. II C), and hence can be further reduced by powerful phase polynomial techniques such as in Refs. [8,9]. In order to obtain a computation that is deterministically realizable, it will be necessary to replace postselected nodes with measurements and classical corrections, which may be possible with relatively low overhead using techniques such as those in Refs. [9,33,34]. While this can be done straightforwardly for several small examples, we leave development of an efficient algorithm for extracting deterministic circuits with ancillae and classical control as a topic of future work.

While the ZX-calculus forms a powerful language for reasoning about low-level gate sets (e.g., Clifford+T), it can only reason about Toffoli and CCZ gates in an indirect manner, by first translating those gates into Clifford+T representations. The *ZH-calculus* [35] in contrast, makes it straightforward to reason about mid-level quantum gates such as Toffoli and CCZ gates. It then stands to reason that we can achieve further optimizations for circuits defined in terms of these mid-level gates (such as adders and classical oracles), by first doing simplifications in the ZH-calculus, then translating the diagram into a Clifford+T gate set, and doing further simplifications in the ZX-calculus.

One final open question concerns the power of our circuit equality validation scheme, using the ZX-calculus simplifier. We have already noted that this scheme seems to be powerful enough to validate any optimization made by our simplification routine or the one found in Ref. [11]. It is then an interesting question to explore the exact power (and limitations) of this scheme.

Note added: Recently, Zhang and Chen reported nearly identical numbers to those in Table I, using an independent approach based on Pauli rotations [36]. It is a topic of ongoing research as to why these seemingly quite different methods produce the same T-counts.

ACKNOWLEDGMENTS

The authors are supported in part by AFOSR Grant No. FA2386-18-1-4028. We thank E. Campbell and L. Heyfron for useful discussions and for help running the Topt tool as well as M. Amy for checking various optimized circuits in the Feynman verifier [31]. We furthermore thank Q. Wang and N. de Beaudrap for interesting discussions on the ZX calculus, phase gadgets, and circuit optimization. Finally, we thank W. Zeng and the Unitary Fund, as well as Cambridge Quantum Computing, for their support of the PyZX project.

[1] S. Aaronson and D. Gottesman, Improved simulation of stabilizer circuits, Phys. Rev. A **70**, 052328 (2004).

[2] R. Raussendorf and J. Harrington, Fault-Tolerant Quantum Computation with High Threshold in Two Dimensions, Phys. Rev. Lett. **98**, 190504 (2007).

[3] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, Surface code quantum computing by lattice surgery, New J. Phys. **14**, 123011 (2012).

[4] E. T. Campbell, B. M. Terhal, and C. Vuillot, Roads towards fault-tolerant universal quantum computation, Nature (London) **549**, 172 (2017).

[5] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, 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 **32**, 818 (2013).

[6] O. Di Matteo and M. Mosca, Parallelizing quantum circuit synthesis, Quant. Sci. Technol. **1**, 015003 (2016).

[7] 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**, 1476 (2014).

[8] M. Amy and M. Mosca, T-count optimization and Reed-Muller codes, Trans. Inf. Theory **65**, 4771 (2019).

[9] L. E. Heyfron and E. T. Campbell, An efficient quantum compiler that reduces T count, Quantum Science and Technology **4**, 015004 (2018).

[10] N. Abdessaied, M. Soeken, and R. Drechsler, Quantum circuit optimization by Hadamard gate reduction, in *International Conference on Reversible Computation* (Springer, Berlin, 2014), pp. 149–162.

[11] 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**, 23 (2018).

[12] R. Duncan, A. Kissinger, S. Pedrix, and J. van de Wetering, Graph-theoretic simplification of quantum circuits with the ZX calculus, Quantum **4**, 279 (2020).

[13] B. Coecke and R. Duncan, Interacting quantum observables, in *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science (Oxford University, Oxford, 2008).

[14] B. Coecke and R. Duncan, Interacting quantum observables: categorical algebra and diagrammatics, New J. Phys. **13**, 043016 (2011).

[15] M. Backens, The ZX calculus is complete for stabilizer quantum mechanics, New J. Phys. **16**, 093021 (2014).

[16] A. Kissinger and J. van de Wetering, PyZX: Large scale automated diagrammatic reasoning, in *Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019*, edited by B. Coecke and M. Leifer, Electronic Proceedings in Theoretical Computer Science Vol. 318 (Open Publishing Association, 2020), pp. 229–241.

[17] A. D. Bookatz, QMA-complete problems, Quant Info. Comput. **14**, 2014 (2014).

[18] B. Coecke and A. Kissinger, *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning* (Cambridge University Press, Cambridge, 2017).

[19] R. Penrose, Applications of negative dimensional tensors, in *Combinatorial Mathematics and its Applications* (Academic Press, New York, 1971), pp. 221–244.

[20] E. Jeandel, S. Perdrix, and R. Vilmart, A complete axiomatisation of the ZX calculus for Clifford+T quantum mechanics, in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18* (ACM, New York, NY, 2018), pp. 559–568.

[21] A. Hadzihasanovic, K. F. Ng, and Q. Wang, Two complete axiomatisations of pure-state qubit quantum computing, in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18* (ACM, New York, NY, USA, 2018) pp. 502–511.

[22] E. Jeandel, S. Perdrix, and R. Vilmart, Diagrammatic reasoning beyond Clifford+T quantum mechanics, in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18* (ACM, New York, NY, 2018), pp. 569–578.

[23] R. Vilmart, A near-minimal axiomatisation of zx-calculus for pure qubit quantum mechanics, in *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)* (IEEE, Canada, 2019), pp. 1–10.

[24] M. Amy, P. Azimzadeh, and M. Mosca, On the controlled-NOT complexity of controlled-NOT–phase circuits, Quant. Sci.e Technol. **4**, 015002 (2018).

[25] M. Backens, H. Miller-Bakewell, G. de Felice, L. Lobski, and J. van de Wetering, There and back again: A circuit extraction tale, arXiv:2003.01664 (2020).

[26] L. Heyfron, Topt circuit optimiser, https://github.com/Luke-Heyfron/TOpt.

[27] M. Amy, T-par: A quantum circuit optimizer based on sum-over-paths representations, https://github.com/meamy/t-par.

[28] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov, optimizer: Benchmark quantum circuits before and after optimization, https://github.com/njross/optimizer.

[29] A. Kissinger and J. van de Wetering, PyZX: A circuit optimisation tool based on the ZX-caculus, http://github.com/Quantomatic/pyzx.

[30] N. Chancellor, A. Kissinger, J. Roffe, S. Zohren, and D. Horsman, Graphical Structures for Design and Verification of Quantum Error Correction, arXiv:1611.08012 (2016).

[31] M. Amy, Towards large-scale functional verification of universal quantum circuits, in *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018*, edited by P. Selinger and G. Chiribella, Electronic Proceedings in Theoretical Computer Science Vol. 287 (Open Publishing Association, 2019), pp. 1–21.

[32] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).

[33] R. Duncan and S. Perdrix, Rewriting measurement-based quantum computations with generalised flow, in *International Colloquium on Automata, Languages, and Programming* (Springer, Berlin, 2010), pp. 285–296.

[34] A. Kissinger and J. van de Wetering, Universal MBQC with generalised parity-phase interactions and Pauli measurements, Quantum **3**, 134 (2019).

[35] 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, Halifax, Canada, 3-7th June 2018*, edited by P. Selinger and G. Chiribella, Electronic Proceedings in Theoretical Computer Science Vol. 287 (Open Publishing Association, 2019), pp. 23–42.

[36] F. Zhang and J. Chen, Optimizing T gates in Clifford+T circuit as $\pi/4$ rotations around Paulis, arXiv:1903.12456 (2019) .