Polynomial-Time T-Depth Optimization of Clifford+T Circuits Via Matroid Partitioning

Matthew Amy, Dmitri Maslov, and Michele Mosca

***Abstract*—Most work in quantum circuit optimization has been performed in isolation from the results of quantum fault-tolerance. Here we present a polynomial-time algorithm for optimizing quantum circuits that takes the actual implementation of fault-tolerant logical gates into consideration. Our algorithm resynthesizes quantum circuits composed of Clifford group and T gates, the latter being typically the most costly gate in fault-tolerant models, e.g., those based on the Steane or surface codes, with the purpose of minimizing both T-count and T-depth. A major feature of the algorithm is the ability to resynthesize circuits with ancillae at effectively no additional cost, allowing space-time trade-offs to be easily explored. The tested benchmarks show up to 65.7% reduction in T-count and up to 87.6% reduction in T-depth without ancillae, or 99.7% reduction in T-depth using ancillae.**

***Index Terms*—Circuit optimization, circuit synthesis, Clifford+T, matroid partitioning, quantum circuits, sum over paths.**

I. Introduction

QUANTUM computation has the potential to efficiently solve important computational problems, including integer factorization [35] and quantum simulation [24], for which there are no known efficient classical algorithms. However, even with recent advances in quantum information processing technologies [6], [7], [9], [32], the prospects of scalable quantum computing without some systematic way of mitigating physical errors and noise are few.

The active fields of quantum error correction and fault-tolerance provide such tools for constructing scalable quantum computers. By combining physical qubits through the use of error correcting codes and providing fault-tolerant encoded gates, larger computations can be achieved with high fidelity—by concatenating codes, or in topological codes by increasing code distance—provided the physical operations achieve a certain threshold fidelity. With recent improvements to fault-tolerant thresholds [5], [17], [18], scalable quantum computation is becoming more and more viable, resulting in a growing need for efficient automated design tools targeting fault-tolerant quantum computers.

Quantum circuit synthesis and optimization is particularly important, given the prevalence of the circuit model of quantum computation, but previous work has been largely isolated from the unique concerns of fault-tolerance. While at the physical level multiqubit gates such as the controlled-NOT (CNOT) are generally the hardest to perform, most of the common quantum error-correcting codes admit efficient CNOT encodings. Moreover, for fault-tolerant models based on (double even, self-dual) CSS codes, e.g., the Steane code, as well as the promising surface codes, the Clifford group can be implemented as logical gates with little cost [17], [19].

For universal quantum computing, however, at least one nonClifford group gate is needed, which typically requires large ancilla factories and gate teleportation to implement fault-tolerantly [8], [20]. As the nonClifford T gate has known constructions in most of the common error correction schemes, the standard universal fault-tolerant gate set is taken to be "Clifford + T." Given the high cost of the fault tolerant implementations of the T gate [1], [17], exceeding the cost of Clifford group gates by as much as a factor of a hundred or more, it has recently been proposed that efficient circuits should minimize both the number of T gates and the maximum number of T gates in any circuit path [2], [16]—we call these metrics a circuit's T-count and T-depth, respectively. Indeed, Fowler [16] shows how to perform fault-tolerant computations in time proportional to one round of measurement per round of simultaneous T gates, and as a result the T-depth directly determines a circuit's runtime. Likewise, reducing the number of T gates reduces the number of ancilla states that require preparation, vastly reducing circuit volume and at the same time increasing fidelity. While the primary purpose of our work is to optimize T-depth (circuit runtime), our algorithm also provides significant reductions to T-count (circuit volume).

Some recent work has been done concerning minimization of T-depth [2], [34], though these previous results focus on finding small optimal two- and three-qubit circuits [2], classes of circuits that can be parallelized to T-depth 1 by adding ancillae [2], [34]. By contrast, we report a scalable

Manuscript received December 13, 2013; revised April 8, 2014; accepted May 26, 2014. Date of current version September 16, 2014. This work was supported by the Intelligence Advanced Research Projects Activity through the Department of Interior National Business Center under Contract DllPC20l66. The work of D. Maslov was supported by the National Science Foundation, and the work of M. Mosca was supported by Canada's NSERC, MPrime, CIFAR, and CFI. The work of IQC and Perimeter Institute was supported in part by the Government of Canada and in part by the Province of Ontario. This paper was recommended by Associate Editor R. Drechsler.

M. Amy is with the Department of Computer Science, University of Toronto, Toronto, ON M5S 2J7, Canada (e-mail: matt.e.amy@gmail.com).

D. Maslov is with the National Science Foundation, Arlington, VA 22230 USA (e-mail: dmitri.maslov@gmail.com).

M. Mosca is with the Institute for Quantum Computing, also with the Department of Combinatorics & Optimization, University of Waterloo, Waterloo, ON, Canada, and also with the Perimeter Institute for Theoretical Physics, Waterloo, ON N2L 3G1, Canada (e-mail: michele.mosca@uwaterloo.ca).

Digital Object Identifier 10.1109/TCAD.2014.2341953

automated tool for the optimization of T-depth that functions with or without ancillae, and is not limited to a few qubits or a specific class of circuits. In particular, we present a polynomial-time algorithm for optimizing both the T-depth and T-count of quantum circuits composed of Clifford group and T gates. The algorithm also makes automatic use of ancillae to optimize T-depth, with the addition of ancillae typically decreasing the runtime of our software implementation. Our experiments show on average 62.3% reduction in T-depth and 41.4% reduction in T-count without adding any ancillae, using the available benchmarks. When the use of ancillae is allowed, the average T-depth reduction is demonstrated to be as high as 80.2% (the more ancillae are allowed the more parallelization becomes possible, in some cases reducing T-depth by as much as 99.7%).

The rest of this paper is structured as follows. Section II reviews some background on quantum and reversible computation, and introduces the notations we will use. Sections III and IV describe the algorithmic core—a procedure that optimally parallelizes the T gates in a circuit composed of CNOT and T gates by performing matroid partitioning. Section V develops a heuristic extending the optimal $\{\text{CNOT}, T\}$ core to a universal gate set, while Section VI describes the final algorithm. Section VII reports our experimental results, and Section VIII concludes the paper.

II. Preliminaries

We begin by reviewing some basic facts about quantum and reversible circuits necessary for this paper.

In the classical circuit model, the state of a system of n bits is represented as a binary string of length n, with classical gates corresponding to operators that map length-n binary strings to length-m binary strings. More precisely, length-n binary strings are vectors of $\mathbb{F}_2^n$, where $\mathbb{F}_2$ is the two-element finite field with addition corresponding to Boolean exclusive-OR (EXOR, $\oplus$) and multiplication corresponding to Boolean AND ($\wedge$). We then represent classical gates as operators $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$, and we typically refer to f as a (classical) function. For brevity, if $m = 1$ we call f Boolean.

The quantum circuit model, one of the prominent models of quantum computation [30], generalizes the classical circuit model to deal with quantum effects. In particular, it describes the state space of a system of n qubits as a vector in a 2^n-dimensional complex vector space $\mathcal{H}$ spanned by the (classical) n bit states. By convention, we refer to the classical states as the standard or computational basis of $\mathcal{H}$ and write them in Dirac notation [12]: $|x\rangle, x \in \mathbb{F}_2^n$.

In contrast to the classical circuit model, quantum gates are restricted to a subset of all operators on $\mathcal{H}$—specifically, quantum gates are linear operators $U : \mathcal{H} \to \mathcal{H}$ that preserve the L_2 norm. Such operators U satisfy $U^\dagger U = UU^\dagger = I$, where $U^\dagger$ denotes the adjoint of U, and are known as unitary [30]. Given that unitary operators are invertible, we see that the subset of quantum transformations that permute the computational basis states are exactly the set of invertible classical transformations—we call such functions reversible, with the intuition that any computation performed by reversible functions can be undone or reversed. The Toffoli gate

$$\Lambda_2(X) : |x\rangle|y\rangle|z\rangle \mapsto |x\rangle|y\rangle|z \oplus (x \wedge y)\rangle$$

is an example of a reversible function.

We can also have classical/quantum computations that use ancillae—being bits/qubits that can be initialized to the $0/|0\rangle$ or $1/|1\rangle$ state and act as a temporary register. Without loss of generality, we require that all ancillae are initialized in the $0/|0\rangle$ state. In the case of a circuit with n bits, m of which are data bits (i.e., $n - m$ is the number of ancillae), we describe the state space as some subspace V of $\mathbb{F}_2^n$ with dimension m. We will typically use n to represent the total number of bits in a system, and m to refer to the number of data bits.

While all reversible classical gates are linear as operators over $\mathcal{H}$, they need not be linear as operators over $\mathbb{F}_2^n$. In particular, we call $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ linear if $f(x \oplus y) = f(x) \oplus f(y)$. For instance, the controlled-NOT gate

$$\text{CNOT} : |x\rangle|y\rangle \mapsto |x\rangle|x \oplus y\rangle$$

is both linear over $\mathbb{F}_2^n$ and reversible, while the Toffoli gate is nonlinear. It is a known result that linear reversible functions are exactly those that can be computed by a circuit consisting of only CNOT gates [31].

Throughout this paper we will also be interested in linear Boolean functions and their relation to linear reversible functions. For convenience, we refer to the set of n-ary linear Boolean functions $\mathbb{F}_2^n \to \mathbb{F}_2$ as the dual vector space $(\mathbb{F}_2^n)^*$ of $\mathbb{F}_2^n$, and note that a linear Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be represented as a row vector over $\mathbb{F}_2^n$—i.e., x^T for some $x \in \mathbb{F}_2^n$. Furthermore, for a set of linear Boolean functions $S \subseteq (\mathbb{F}_2^n)^*$, we define rank($S$) as the maximum number of independent (row) vectors in S, or equivalently the dimension of the subspace V^* spanned by S.

As this paper is concerned with the optimization of quantum circuits, we also define some quantum gates commonly used in fault tolerant models. In particular, we define the T and Hadamard gates

$$T : |x\rangle \mapsto e^{\frac{i\pi}{4}x}|x\rangle, \quad H : |x\rangle \mapsto \frac{|0\rangle + (-1)^x|1\rangle}{\sqrt{2}}.$$

We will show that circuits over the gate set $\{\text{CNOT}, T\}$ implement linear reversible functions with discrete phases corresponding to the eighth roots of unity. A side result of this paper is a proof that $\{\text{CNOT}, T\}$ circuits can be simulated on a classical computer in polynomial time. If this set is further extended with the Hadamard gate we achieve a gate set that is universal for quantum computation [13]. The universal $\{H, \text{CNOT}, T\}$ gate set is typically called the "Clifford + T" gate set as it extends the *Clifford group* $\mathcal{C}_n$, defined as all n qubit circuits over $\{H, P := T^2, \text{CNOT}\}$, with the T gate. Moreover, $\{H, \text{CNOT}, T\}$ is a minimal generating set for the Clifford + T gate set.

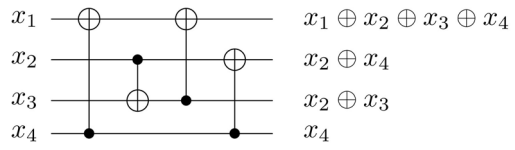


Fig. 1. Linear reversible circuit computing the functions $x_1 \oplus x_2 \oplus x_3 \oplus x_4$, $x_2 \oplus x_4$, and $x_2 \oplus x_3$.

Since quantum gates are commonly defined by unitary matrices, we provide the equivalent matrix definitions below

$$H := \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix},\ \text{CNOT} := \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}$$

$$T := \begin{pmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{pmatrix}.$$

A. Computable Sets of Linear Boolean Functions

While every linear reversible function f of arity n can be written as n linear Boolean functions

$$f(x_1, x_2, \ldots, x_n) = (f_1(x_1, \ldots, x_n), \ldots,\ f_n(x_1, \ldots, x_n))$$

a set of linear Boolean functions may not in general define a reversible function. For instance, $f(x_1, x_2, x_3) = (x_1, x_2, x_1 \oplus x_2)$ is irreversible since the input value x_3 is effectively lost. It turns out that a set of n linear Boolean functions describes the outputs of an n-ary linear reversible function if, and only if, their rank is equal to the dimension of the input space.

Lemma 1: Given a subspace V of $\mathbb{F}_2^n$ and a set of linear Boolean functions $S = \{f_1, f_2, \ldots, f_n\} \subseteq V^*$, the linear function $f{:}V \to \mathbb{F}_2^n$ defined as

$$f(x_1, x_2, \ldots, x_n) = (f_1(x_1, \ldots, x_n), \ldots,\ f_n(x_1, \ldots, x_n))$$

is reversible if, and only if, $\text{rank}(S) = \dim(V)$.

Proof: Following the rank-nullity theorem, $\text{rank}(f) = \dim(V) - \dim(\text{Ker}(f))$, and the fact that f is reversible if, and only if, f is one-to-one by noting that the (row) vectors of S form the transformation matrix of f. ■

Since the unitary quantum circuit model is reversible, a set of linear Boolean functions S can only be computed simultaneously (i.e., there exists a quantum circuit implementing the transformation $|x_1x_2\cdots x_n\rangle \mapsto |y_1y_2\cdots y_n\rangle$, where for each $f \in S$, $f(x_1, x_2, \ldots, x_n) = y_i$ for some i) if it defines a reversible function. We call such a set (reversibly) computable—as we will be concerned strictly with reversible computations, we frequently omit the qualifier reversible.

We will also want to know whether a set S of linear Boolean functions is computable on $n > |S|$ qubits, as in Fig. 1. Since every n-ary linear reversible function can be written as n linear Boolean functions, clearly S is computable if, and only if, there exists some computable size n superset S' of S—specifically, $S \subseteq S' \subseteq V^*$ such that $|S'| = n$ and $\text{rank}(S') = \dim(V)$. However, directly checking whether such a set exists would be costly, and so we instead establish a necessary and sufficient condition similar to Lemma 1.

Lemma 2: Given a subspace V of $\mathbb{F}_2^n$ and a set of linear Boolean functions $S \subseteq V^*$, there exists a linear reversible function computing S if, and only if

$$\dim(V) - \text{rank}\,(S) \leq n - |S|. \tag{1}$$

Proof: Suppose there exists a superset S' of S with cardinality n and $\text{rank}(S') = \dim(V)$. Then

$$\begin{aligned} n - |S| = |S' \setminus S| &\geq \text{rank}(S') - \text{rank}(S)\\ &= \dim(V) - \text{rank}(S).\end{aligned}$$

Suppose instead that $\dim(V) - \text{rank}(S) \leq n - |S|$. Then any maximal linearly independent subset of S can be extended to a complete basis of V^* by adding $\dim(V) - \text{rank}(S)$ vectors. As a result there exists a rank $\dim(V)$ superset S' of S with cardinality $|S'| = |S| + \dim(V) - \text{rank}(S) \leq n$. ■

We note that when $\dim(V) = n$, inequality (1) implies $|S| = \text{rank}(S)$, i.e., S is linearly independent.

III. {CNOT, T} Circuits

We first consider circuits over the gate set {CNOT, T, $P{:}= T^2$, $Z{:}= T^4$, $T^\dagger{:}= T^7$, $P^\dagger{:}= T^6$}, as they have a particular property that will be crucial to synthesizing low T-depth circuits. Note that all even powers of the T gate are all Clifford gates while all odd powers lie outside the Clifford group, and that no T^k gate requires more than a single nonClifford T gate to implement. This is an essential observation for practical considerations. We usually omit the extraneous gates and refer to this gate set by the generating set {CNOT, T}.

It can be observed that since $\text{CNOT}|x\rangle|y\rangle = |x\rangle|x \oplus y\rangle$ and $T|x\rangle = e^{\frac{i\pi x}{4}}|x\rangle$ for $x, y \in \mathbb{F}_2$, a {CNOT, T} circuit can be described as computing a linear reversible function on the input basis state, with an added phase that is some power of $\omega{:}= e^{i\pi/4}$. Stated more precisely in [2, Lemma 2],

Lemma 3: A unitary $U \in U(2^n)$ is exactly implementable by an n-qubit circuit over {CNOT, T} if, and only if

$$U|x_1x_2\cdots x_n\rangle = \omega^{p(x_1,x_2,\ldots,x_n)}|g(x_1, x_2, \ldots, x_n)\rangle$$

where $x_1, x_2, \ldots, x_n \in \mathbb{F}_2$ and

$$p(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{l} c_i \cdot f_i(x_1, x_2, \ldots, x_n)$$

for some linear reversible function $g \in \mathbb{F}_2^n \to \mathbb{F}_2^n$ and linear Boolean functions $f_1, f_2, \ldots, f_l \in (\mathbb{F}_2^n)^*$ with coefficients $c_1, c_2, \ldots, c_l \in \mathbb{Z}_8$.

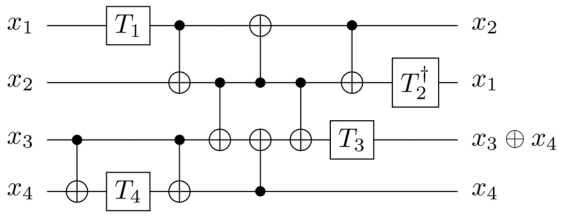
As a result, we can fully characterize any unitary implementable by a {CNOT, T} circuit with a set $S \subseteq \mathbb{Z}_8 \times (\mathbb{F}_2^n)^*$ of linear Boolean functions together with coefficients in $\mathbb{Z}_8$, and a linear reversible output function $g{:}\mathbb{F}_2^n \to \mathbb{F}_2^n$. The unitary can then be recovered with the interpretation

$$\begin{aligned} &U_{\langle S,g\rangle}{:}|x_1x_2\cdots x_n\rangle \mapsto \omega^{p(x_1,x_2,\ldots,x_n)}|g(x_1, x_2, \cdots x_n)\rangle\\ &p(x_1, x_2, \ldots, x_n) = \textstyle\sum_{(c,f)\in S} c \cdot f(x_1, x_2, \ldots, x_n).\end{aligned}$$

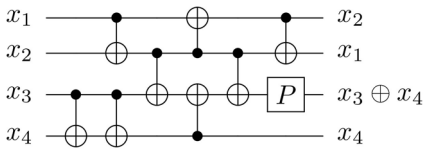
Moreover, given a {CNOT, T} circuit S and g are efficiently computable, taking time linear in the number of qubits and gates. In computing S it also becomes apparent when phases

from T gates, possibly physically separated within the circuit, accumulate and can be applied by a single gate, such as $P : |x\rangle \mapsto \omega^{2x}|x\rangle$. Our experiments show that a large number of T gates can be removed in this way.

Example 1: Consider the following circuit.



We can compute S and g by tracking the effect of CNOT gates on the state, and taking note of the state when a phase gate is applied. For instance, in the circuit above we see that T_1 and $T_2^\dagger$ (indices are used to mark different T gates within the circuit) are both applied to qubits with state $|x_1\rangle$, resulting in a cumulative phase of $\omega^{x_1+7x_1} = \omega^{8x_1} = 1$. As such, both gates can be removed. Likewise, T_3 and T_4 are both applied to qubits in the state $|x_3 \oplus x_4\rangle$, giving a cumulative phase of $\omega^{2(x_3 \oplus x_4)}$—this pair of T gates can thus be replaced with a single P gate. The result is an optimized circuit:



which may be optimized further by rewriting the linear reversible section. Note that the outputs of the circuit specify g, i.e., $g(x_1, x_2, x_3, x_4) = (x_2, x_1, x_3 \oplus x_4, x_4)$.

Once S and g have been computed, the proof of Lemma 3 [2] gives a constructive method for synthesizing a circuit implementing $U_{\langle S,g\rangle}$. However, this naïve method of resynthesis may end up with worse T-depth than the original circuit, despite the possibly reduced T-count.

We can instead recall from Section II that if a subset $A \subseteq S$ is reversibly computable (i.e., if the linear Boolean functions in A are reversibly computable), we can construct a linear reversible function with outputs simultaneously computing the functions in A; in this case, $|A|$ of the necessary phase factors could then be applied in parallel. Synthesis thus proceeds by partitioning S into computable subsets, then for each partition $A \subseteq S$ first compute a (reversible) superset of A with a stage of CNOT gates—many efficient algorithms exist [4], [26], [31] that can decompose a linear reversible function into CNOT gates. Next we apply the relevant phase gates in parallel to add the phase $\omega^{\sum_{(c,f)\in A} c \cdot f(x_1,x_2,\ldots,x_n)}$, and finally uncompute by reversing the CNOT stage. Given that at most one T gate is used to implement any integer power of T, every partition will have a T-depth of at most 1.

As a result, any unitary U implementable over $\{\text{CNOT}, T\}$ can be implemented in T-depth k where k is the minimum number of sets partitioning[1] $S_1 = \{(c, f) \in S \mid c \equiv 1 \mod 2\}$ into computable subsets, as elements in $S_0 = S \setminus S_1$ do not require T gates to implement. In fact, we can trivially see that given a specific set S and output g, k is the minimal T-depth, as any layer of T gates in a circuit implementing $U_{\langle S,g\rangle}$ corresponds to a computable subset of S.

It is important to note that while this method reduces and maximally parallelizes the T gates, it will not necessarily find the optimal T-count and by extension T-depth. Specifically, given a $\{\text{CNOT}, T\}$ circuit with phase defined by the set S there may be some distinct S' that defines an equivalent computation using fewer T gates. Consider, for instance, the set $S_\emptyset = \left\{(1, f) \mid f \in (\mathbb{F}_2^4)^*\right\}$. We note that the integer sum $\sum_{f\in(\mathbb{F}_2^4)^*} f(x_1, x_2, x_3, x_4)$ is equal to 8 for any nonzero $(x_1, x_2, x_3, x_4) \in \mathbb{F}_2^4$, and 0 for $(0, 0, 0, 0)$. Since $\omega^8 = \omega^0 = 1$, $S_\emptyset$ computes the trivial phase on every input and is therefore equivalent to the empty set $\emptyset$. In this case, all $|S_\emptyset| = 15$ T gates can be removed.

It turns out, through a brute force search, that for $n < 4$ no two sets S, S' define equivalent circuits with distinct T-counts. As a result, we obtain a proof that the doubly controlled-Z gate requires seven T gates to implement over $\{\text{CNOT}, T\}$ with any number of ancillae. For $n \geq 4$ however, the problem of minimizing T-count in $\{\text{CNOT}, T\}$ circuits reduces to a minimization problem over multilinear polynomials in mixed arithmetic; moreover, since every such polynomial defines the global phase for some $\{\text{CNOT}, T\}$ circuit, the two problems are in fact equivalent. To the authors' knowledge no bounds on the complexity of this problem have yet been found.

A. Parallelizing $\Lambda_2(Z)$

To illustrate $\{\text{CNOT}, T\}$ resynthesis, consider the circuit in Fig. 2, implementing the doubly controlled-Z gate

$$\Lambda_2(Z) : |x_1x_2x_3\rangle \mapsto \omega^{4(x_1 \wedge x_2 \wedge x_3)}|x_1x_2x_3\rangle$$

over $\{\text{CNOT}, T\}$. We track the effect of each CNOT gate on the state of the qubits, as annotated in the circuit. When a T or $T^\dagger$ gate is applied, we add a term in the exponent of ω corresponding to the state of the target qubit with a coefficient of 1 or 7, respectively. The resulting transformation is $\Lambda_2(Z) : |x_1x_2x_3\rangle \mapsto \omega^{p(x_1,x_2,x_3)}|x_1x_2x_3\rangle$, where

$$p(x_1, x_2, x_3) = x_1 + x_2 + x_3 + 7(x_1 \oplus x_2) + 7(x_1 \oplus x_3) + 7(x_2 \oplus x_3) + (x_1 \oplus x_2 \oplus x_3).$$

In fact, since $2(x \wedge y) = x + y - x \oplus y = x + y + 7(x \oplus y) \mod 8$, we see that $\omega^{p(x_1,x_2,x_3)} = \omega^{4(x_1 \wedge x_2 \wedge x_3)}$ as expected.

The T-stages in Fig. 2 also correspond to a partition of S, specifically

$$\left\{ \begin{matrix} \{x_1, x_2\}, \{7(x_1 \oplus x_3), 7(x_2 \oplus x_3)\}, \\ \{x_1 \oplus x_2 \oplus x_3\}, \{7(x_1 \oplus x_2), x_3\} \end{matrix} \right\}.$$

It can easily be verified that for each subset S in this partition, $\dim(V) - \text{rank}(S) \leq n - |S|$ (i.e., S is computable by

[1]For the remainder of this paper we do not separate S_0 and S_1 to simplify the presentation, though our algorithm can be trivially modified to partition S_0 and S_1 separately.

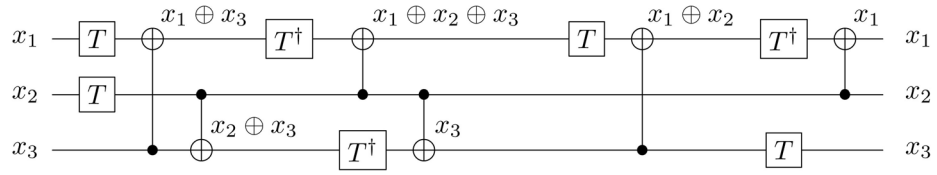


Fig. 2. {CNOT, T} circuit implementing the doubly controlled Z gate.

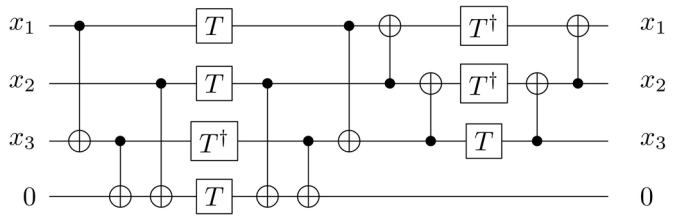


Fig. 3. T-Depth 2 implementation of Fig. 2 with one ancilla.

Lemma 2), since $\dim(V) = n = 3$ and each subset is linearly independent. By contrast, the partition

$$\left\{ \begin{matrix} \{x_3\}, \{7(x_1 \oplus x_3), 7(x_2 \oplus x_3), 7(x_1 \oplus x_2)\}, \\ \{x_1 \oplus x_2 \oplus x_3\}, \{x_1, x_2\} \end{matrix} \right\}$$

could not have been synthesized as the set $\{7(x_1 \oplus x_3), 7(x_2 \oplus x_3), 7(x_1 \oplus x_2)\}$ has rank 2, thus the mapping $|x_1x_2x_3\rangle \mapsto |(x_1 \oplus x_3)(x_2 \oplus x_3)(x_1 \oplus x_2)\rangle$ is not reversible.

While we have not yet described how to find a minimal computable partition algorithmically, we can observe that the T-depth 3 partition

$$\left\{ \begin{matrix} \{x_1, x_2, 7(x_1 \oplus x_3)\}, \{x_1 \oplus x_2 \oplus x_3\}, \\ \{x_3, 7(x_1 \oplus x_2), 7(x_2 \oplus x_3)\} \end{matrix} \right\}$$

is computable since each subset satisfies (1), and is also minimal. If, however, we added an ancilla when resynthesizing Fig. 2, we can use the extra qubit to generate a smaller partition. In particular, we know $|x_1x_2x_3\rangle \mapsto |x_1x_2(x_1 \oplus x_3)\rangle$ is reversible so $|x_1x_2x_3\rangle|0\rangle \mapsto |x_1x_2(x_1 \oplus x_3)\rangle|0\rangle$ is as well. We can then compute the value $x_1 \oplus x_2 \oplus x_3$ into the ancilla with 2 CNOT gates and apply 4 T gates (one for each qubit) at the same time. To connect this intuitive idea with inequality (1), we observe that $n = 4$, $\dim(V) = 3$ since the input $|x_1x_2x_3\rangle|0\rangle$ spans a space of dimension 3, and

$$\text{rank}\left(\{x_1, x_2, 7(x_1 \oplus x_3), x_1 \oplus x_2 \oplus x_3\}\right) = 3$$

so $\dim(V) - \text{rank}(S) = 0 \leq 0 = n - |S|$. Fig. 3 shows the resulting circuit, implementing $\Lambda_2(Z)$ in T-depth 2.

IV. Matroids

We now turn our attention to the problem of determining a minimal partition of a set of linear Boolean functions into computable sets. Due to the connection between the computability condition (1) on sets of linear Boolean functions and linear independence, we are able to phrase the problem as a matroid partitioning problem. To do so, we first introduce the concept of a matroid [39], an algebraic structure that generalizes the idea of linear independence in vector spaces.

Definition 1: A finite matroid is a pair (S, I) where S is a finite set and I is a set of subsets of S such that:

1) $\emptyset \in I$;
2) for all $A, B \subset S$, if $A \in I$ and $B \subset A$, then $B \in I$;
3) for all $A, B \in I$, if $|A| > |B|$, then there exists some $a \in A$ such that $B \cup \{a\} \in I$.

It turns out that a set of linear Boolean functions, together with an independence relation defined by the equality (1), forms a matroid:

Lemma 4: For any subspace V of $\mathbb{F}_2^n$ with dimension m and set of linear Boolean functions $S = \{f_1, f_2, \ldots, f_k\} \subseteq V^*$, let I denote the set

$$\{A \subseteq S \mid m - \text{rank}\,(A) \leq n - |A|\}.$$

The pair (S, I) is a finite matroid.

Proof: We verify that (S, I) satisfies all three conditions of Definition 1.

1) $m - \text{rank}(\emptyset) \leq n - |\emptyset|$ is trivially true since $m \leq n$. Thus $\emptyset \in I$.
2) Suppose $A, B \subset S$, where $A \in I$ and $B \subset A$. Since $A = B \cup (A \setminus B)$ we see that

$$\begin{aligned} \text{rank}(A) &\leq \text{rank}(B) + \text{rank}(A \setminus B) \\ &\leq \text{rank}(B) + (|A| - |B|) \end{aligned}$$

and so $\text{rank}(A) - \text{rank}(B) \leq |A| - |B|$. Since $m - \text{rank}(A) \leq n - |A|$, we see that

$$m \leq n + \text{rank}(A) - |A| \leq n + \text{rank}(B) - |B|$$

and thus $m - \text{rank}(B) \leq n - |B|$.
3) Suppose $A, B \in I$ and $|A| > |B|$. If $\text{rank}(A) \leq \text{rank}(B)$, then

$$m - \text{rank}(B) \leq m - \text{rank}(A) \leq n - |A| < n - |B|$$

and so $m - \text{rank}(B \cup \{s\}) \leq n - |B \cup \{s\}|$ for any $s \in A$. Otherwise, $\text{rank}(A) > \text{rank}(B)$ and we can let A' and B' be maximal linearly independent subsets of A and B, respectively. Since $A' \not\subseteq \text{span}(B')$, for any $s \in A \setminus \text{span}(B')$, $B' \cup \{s\}$ is linearly independent. Then

$$\begin{aligned} m - \text{rank}(B' \cup \{s\}) &= m - \text{rank}(B \cup \{s\}) \\ &= m - \text{rank}(B) - 1 \\ &\leq n - |B| - 1 \\ &= n - |B \cup \{s\}|. \end{aligned}$$

■

With Lemmas 2 and 4 we see that the problem of finding a minimal partition of the phase factors in a {CNOT, T} circuit reduces to the more general matroid partitioning problem.

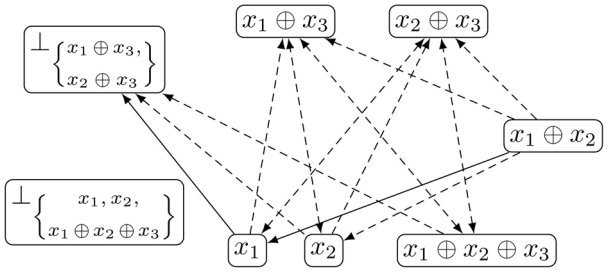


Fig. 4. Directed graph G_s constructed when adding $x_1 \oplus x_2$ to the minimal partition $\{\{x_1 \oplus x_3, x_2 \oplus x_3\}, \{x_1, x_2, x_1 \oplus x_2 \oplus x_3\}\}$. A minimum length path is shown in solid lines, resulting in the new partition $\{\{x_1 \oplus x_3, x_2 \oplus x_3, x_1\}, \{x_1 \oplus x_2, x_2, x_1 \oplus x_2 \oplus x_3\}\}$.

A. Matroid Partitioning

The matroid partitioning problem [15] can be defined as follows.

Definition 2 (Matroid partitioning): Given a matroid (S, I), find a partition $\{A_1, A_2, \ldots, A_k\}$ of S such that $A_i \in I$ for each $1 \leq i \leq k$ and for any other partition $\{A'_1, A'_2, \ldots, A'_{k'}\}$ into independent subsets, $k' \geq k$.

Matroid partitioning can, perhaps surprisingly, be solved in polynomial time, given an independence oracle for the matroid [15]. As a result, given an oracle for (1), the T gates in a $\{\text{CNOT}, T\}$ circuit can be optimally partitioned efficiently. Since the condition in Lemma 2 can be checked by using Gaussian elimination to compute the matrix rank in $O(n^3)$ time,[2] we thus see that a minimal partition can be computed in polynomial time. The rest of this section describes an algorithm for computing such a minimal partition.

The algorithm we use for solving the matroid partitioning problem is based on an algorithm due to Edmonds [15]. Given a matroid (S, I) and a minimal (matroid) partition P of $S' \subset S$, we take an element $s \in S \setminus S'$ not already partitioned and construct a minimal partition of $S' \cup \{s\}$. To create the new partition, we construct a directed graph G_s containing a vertex u for every $u \in S' \cup \{s\}$ as well as a vertex $\perp_p$ for every subset $p \in P$. The edges of G_s represent changes to the partition that are invariant under the property of each subset being independent. In particular, for any $u, v \in S' \cup \{s\}$ there is a directed edge $v \to u$ in G_s if, and only if, u is contained in some subset $p \in P$ and $(p \setminus \{u\}) \cup \{v\} \in I$, i.e., v can be added to p if we remove u. Additionally, given a subset $p \in P$ and element $u \in S' \cup \{s\}$, there exists an edge $u \to \perp_p$ if, and only if, $p \cup \{u\} \in I$. A path from s to $\perp_p$ for some subset p gives a set of updates to P that produce a valid partition P' of $S' \cup \{s\}$. Likewise, if there is no such path, there is no partition of size $|P|$ partitioning $S' \cup \{s\}$ (see [15] for a proof), and so a new subset $\{s\}$ is added to P. Fig. 4 shows the full graph G_s for one iteration when computing a minimal partition for $\Lambda_2(Z)$ (Fig. 2).

Rather than generating the graph G_s explicitly for each element s, we try to construct a path from s to some $\perp_p$ breadth-first (Algorithm 1). The time complexity of breadth-first search is $O(|E| + |V|)$ for a graph with edge set E and vertices V. We can note that there are $|S'| + |P| + 1$ vertices and at most $|S'|^2 + |P| \cdot (|S'| + 1) + (|S'| + |P|)$ edges in G_s, as well as the fact that $|P| \leq |S'|$. Since each edge requires a single test for independence in $O(n^3)$ time, the breadth first search requires time in $O(|S'|^2 \cdot n^3 + |S'|)$.

[2]In practice we reduce this to $O(m^2 n)$ by storing vectors in V^* as length $\dim(V) = m$ vectors. The $O(n^3)$ bound is used for simplicity.

Algorithm 1 Matroid Partitioning Algorithm

```
function PARTITION(s, P, (S, I))
    /* I denotes the independence oracle,
       P is a minimal partition */
    Create path queue Q, Q.enqueue(s → ∅)
    Unmark each element of S, mark s
    while Q nonempty do
        t := Q.dequeue()
        for each A ∈ P do
            Set A' := A ∪ {head(t)}
            if A' ∈ I then
                Set A := A'
                for each u → v in path t do
                    Replace u with v in its current partition
                end for
            else
                for each unmarked u ∈ A do
                    if A' \ {u} ∈ I then
                        Q.enqueue(u → t)
                        Mark u
                    end if
                end for
            end if
        end for
    end while
    If no path was found, set P := P ∪ {s}
end function
```

Algorithm 1 details the algorithm for adding an element s to a partition P of matroid (S, I)—the full algorithm follows by iteratively adding each element of the ground set to the initially empty partition, and correctness follows from the property that if P is minimal for (S, I), then the new partition P' is minimal for $(S \cup \{s\}, I)$. Since adding a single element to a partition of i elements takes $O\left((2i)^2 \cdot n^3 + 2i\right)$ time, and $\sum_i^{|S|} i^2 = \frac{|S|^3}{3} + \frac{|S|^2}{2} + \frac{|S|}{6}$, we see that Algorithm 1 can be used to partition the full set S in $O(|S|^3 \cdot n^3)$ time.

V. Extending to Universal Gate Set

In the previous sections we described a method for resynthesizing $\{\text{CNOT}, T\}$ circuits that removes redundant T-gates by computing the total phase and parallelizing the phase gates through matroid partitioning. However, the usefulness of such an algorithm on its own is marred by the fact that $\{\text{CNOT}, T\}$ circuits are a restricted class of quantum circuits—in particular, they do not create superpositions or interference between basis states. To apply the optimization procedure to more complex quantum circuits, we extend these ideas to deal with the universal gate set $\{H, \text{CNOT}, T\}$.

We recall that a Hadamard gate H has the effect

$$H : |x_1\rangle \mapsto \frac{1}{\sqrt{2}} \sum_{x_2 \in \mathbb{F}_2} \omega^{4(x_1 \wedge x_2)} |x_2\rangle$$

on a basis state $x_1 \in \mathbb{F}_2$. We call x_2 a path variable, following in the tradition of similar representations of quantum circuits [11], [33], called sum over paths representations. We note that the state $|x_1\rangle$ is no longer directly available, and instead we have a fresh state $|x_2\rangle$.

Circuits over $\{H, \text{CNOT}, T\}$ can then be described by a phase polynomial and set of linear Boolean outputs, similar to Lemma 3.

Lemma 5: If a unitary $U \in U(2^n)$ is implementable by an n-qubit circuit over $\{H, \text{CNOT}, T\}$ with k Hadamard gates, then

$$U|x_1 x_2 \cdots x_n\rangle = \frac{1}{\sqrt{2^k}} \sum_{x_{n+1},\ldots,x_{n+k} \in \mathbb{F}_2} \omega^{p(x_1,\ldots,x_{n+k})} |y_1 y_2 \cdots y_n\rangle$$

where $x_1, x_2, \ldots, x_n \in \mathbb{F}_2$, $y_i = h_i(x_1, x_2, \ldots, x_{n+k})$ and

$$p(x_1, x_2, \ldots, x_{n+k}) = \sum_{i=1}^{l} c_i \cdot f_i(x_1, \ldots, x_{n+k}) + 4 \sum_{i=1}^{k} x_{n+i} \wedge g_i(x_1, \ldots, x_{n+k})$$

for some linear Boolean functions h_i, f_i, g_i and coefficients $c_i \in \mathbb{Z}_8$. The k path variables $x_{n+1}, \ldots, x_{n+k}$ result from the application of Hadamard gates.

Proof: We sketch a proof by induction. The base cases follow directly from the definitions of H, CNOT, and T, while the inductive step is immediate from linearity of unitaries

$$U \frac{1}{\sqrt{2^k}} \sum_{x_{n+1},\ldots,x_{n+k} \in \mathbb{F}_2} \omega^{p(x_1,x_2,\ldots,x_{n+k})} |y_1 y_2 \ldots y_n\rangle = \frac{1}{\sqrt{2^k}} \sum_{x_{n+1},\ldots,x_{n+k} \in \mathbb{F}_2} \omega^{p(x_1,x_2,\ldots,x_{n+k})} U|y_1 y_2 \ldots y_n\rangle.$$

■

It can be noted that unlike Lemma 3, the converse is not true—some computations of the form in Lemma 5 do not define unitary transformations.

Synthesis of a circuit based on the above representation is more challenging. In particular, Hadamard gates change the state space by adding a new path variable, resulting in some subspace of $\mathbb{F}_2^{n+k}$, possibly with greater dimension if the previous state was linearly dependent with the other qubits (e.g., a zero initialized ancilla). Each linear Boolean function is likewise computable only in some of the possible state spaces. To resynthesize we then need to apply Hadamard gates in such a way as to be able to pick up each phase factor and end in the state space span $(\{y_1, y_2, \ldots, y_n\})$.

Since the application of a Hadamard gate changes the state space, the state of each qubit must be chosen so that the qubits span a suitable space afterwards. As an illustration consider the transformation

$$|x_1 x_2\rangle \mapsto \frac{1}{\sqrt{2}} \sum_{x_3 \in \mathbb{F}_2} \omega^{4(x_3 \wedge x_2)} |(x_1 \oplus x_2) x_3\rangle.$$

We could achieve the correct phase by applying a Hadamard gate on the second qubit first, but the resulting state would be $|x_1 x_3\rangle$, from which we cannot directly construct the output state $|(x_1 \oplus x_2) x_3\rangle$. The simplest way to choose a suitable state for each qubit before applying a Hadamard gate is to use the qubit's state in the original circuit, though there may be other ways of computing such states. During resynthesis we then transform the state to match the state in the original circuit before a particular Hadamard gate is applied.

In this sense, the Hadamard gates are fixed by the original circuit and the resynthesis process needs to place the remaining terms of the phase (i.e., $c_i \cdot f_i(x_1, x_2, \ldots, x_{n+k})$ in between them. One approach is to use the greedy nature of Algorithm 1 to maintain a partition of those functions that are computable in the current state space of the circuit. Specifically, we iterate through the Hadamard gates and for each one we partition any elements that are in the new state space—this step relies on the fact that Algorithm 1 is greedy to avoid having to repartition the elements that were already in the old state space. For any block in the partition containing functions that will not be computable after the next Hadamard gate, we remove the block and synthesize a $\{\text{CNOT}, T\}$ circuit applying those phases. A more detailed description is given in Section VI.

While this method is heuristic, we note that the partition is always minimal for the set of currently computable functions. In particular, Algorithm 1 produces a minimal partition when given a minimal partition, and removing blocks from a partition does not affect minimality—given a subset P' of a minimal partition P, if there existed a partition P_0 of the elements in P' such that $|P_0| < |P'|$, then $P_0 \cup P \setminus P'$ is a partition of the elements in P into fewer sets.

One problem arises in that the dimension of the state space may increase in the next subcircuit (e.g., if the Hadamard is applied to an ancilla qubit). In this case, the independence condition (1) of the matroid changes, and previous partitions may now be invalid under the new inequality. However, as a trivial consequence of the fact that the dimension increases by at most 1, a partition that is no longer independent can be modified to satisfy it by removing exactly one linearly dependent element. Furthermore, if all partitions are modified to satisfy the new independence condition in this way, the new partition is minimal and the elements that were removed can be repartitioned by Algorithm 1.

Lemma 6: Given a subspace V of $\mathbb{F}_2^n$ with $\dim(V) = m$ and a set of linear Boolean functions $S \subseteq V^*$, let

$$I_i = \{A \subseteq S \mid i - \text{rank}\,(A) \leq n - |A|\}.$$

If P is a minimal partition of (S, I_m), then the partition P' defined by removing one linearly dependent element from every $A \in P$ if $A \notin I_{m+1}$ is a minimal partition of (S', I_{m+1}), where S' is the set of elements partitioned by P'.

Proof: Suppose there exists some partition P_0 of (S', I_{m+1}) with $|P_0| < |P'|$. We then see that one element of $S \setminus S'$ can be added to any $A \in P_0$ to give a set in I_m. In particular, consider any $A \in P_0$. Since $m + 1 - \text{rank}(A) \leq n - |A|$ we see that

$$m - \text{rank}(A) \leq n - |A| - 1 = n - |A \cup \{s\}|$$

$$
\begin{aligned}
&[\![X_i]\!]\langle S, Q, H\rangle = \langle S, (g_1, \ldots, g_{i-1}, \neg g_i, \ldots, g_n), H\rangle \\
&[\![Z_i]\!]\langle S, Q, H\rangle = \langle S \uplus \{(4, g_i)\}, Q, H\rangle \\
&[\![Y_i]\!]\langle S, Q, H\rangle = \langle S \uplus \{(4, g_i)\}, Q', H\rangle \qquad \text{where } Q' = (g_1, \ldots, g_{i-1}, \neg g_i, \ldots, g_n) \\
&[\![P_i]\!]\langle S, Q, H\rangle) = \langle S \uplus \{(2, g_i)\}, Q, H\rangle \\
&[\![P_i^\dagger]\!]\langle S, Q, H\rangle) = \langle S \uplus \{(6, g_i)\}, Q, H\rangle \\
&[\![H_i]\!]\langle S, Q, (h_1, h_2, \ldots, h_j)\rangle) = \langle S, Q', H'\rangle \qquad \text{where } Q' = (g_1, \ldots, g_{i-1}, (0, x_{j+i}), \ldots, g_n), \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad H' = (h_1, h_2, \ldots, h_j, \{Q_I = Q, Q_O = Q'\}) \\
&[\![\text{CNOT}_{(i,j)}]\!]\langle S, Q, H\rangle) = \langle S, Q', H\rangle \qquad \text{where } Q' = (g_1, \ldots, g_{j-1}, g_j \oplus g_i, \ldots, g_n) \\
&[\![T_i]\!]\langle S, Q, H\rangle) = \langle S \uplus \{(1, g_i)\}, Q, H\rangle \\
&[\![T_i^\dagger]\!]\langle S, Q, H\rangle) = \langle S \uplus \{(7, g_i)\}, Q, H\rangle
\end{aligned}
$$

Fig. 5. Semantic function $[\![\cdot]\!]$. U_i denotes the gate U applied to qubit i and $\text{CNOT}_{(i,j)}$ specifies i as the control qubit and j as the target. For states $(b, f), (b', f') \in \mathcal{S}$, $\neg(b, f)$ denotes $(1 \oplus b, f)$ and $(b, f) \oplus (b', f')$ denotes $(b \oplus b', f \oplus f')$. We define $S \uplus T$ as the union of S and T where any f such that $(c_1, f) \in S$, $(c_2, f) \in T$ is given coefficient $c_1 + c_2 \mod 8$.

for any $s \in S \setminus S'$. We also note that $n - |A \cup \{s\}| \geq 0$ as required, since any subset of S has rank at most m, so for any $A \in I_{m+1}$, $m + 1 - \text{rank}(A) > 0$ and thus $|A| < n$. Therefore, for any $A \in P_0$, $s \in S \setminus S'$ we have $A \cup \{s\} \in I_m$.

Next we note that for any $T \subseteq S$ there exists a partition of (T, I_m) with size at most $|T| - 1$. This is a simple result of the fact that $m < n$, as any subset $A \subseteq T$ of size 2 has rank at least 1, so

$$m - \text{rank}(A) \leq m - 1 \leq n - 2 = n - |A|.$$

Additionally, any size 1 subset of T is trivially independent under I_m.

Thus, since we can add one element to every partition in P_0, and we can partition the remaining $|S \setminus S'| - |P_0|$ elements into at most $|S \setminus S'| - |P_0| - 1$ partitions, we see that there exists a partition of (S, I_m) with size at most

$$|P_0| + (|S \setminus S'| - |P_0| - 1) = |S \setminus S'| - 1.$$

Since $|S \setminus S'| - 1 < |P|$ we obtain a contradiction. ∎

VI. T-par Algorithm

In the last section we described a heuristic for optimizing T-count and depth over the gate set $\{H, \text{CNOT}, T\}$. In this section, we present the concrete algorithm, T-par , and enlarge the gate set to include the Pauli gates

$$X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \; Y := \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}, \; Z := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

We ignore the irrelevant global phase i in $Y = iXZ$, though it can be recovered by applying $XPXP = iI$ to any qubit. We also note that the algorithm can be generalized to include arbitrary phase gates $R_z(\theta)$. The Appendix gives a demonstration of the T-par algorithm on a simple circuit.

Recall that in the computational basis, the input space of a circuit with n qubits, $n - m$ ancillae and k Hadamard gates is a dimension m subspace V of $\mathbb{F}_2^{n+k}$. We represent the state of a qubit as a vector in the dual space of $\mathbb{F}_2^{n+k}$, $\mathbb{F}_2^{(n+k)*}$, along with a Boolean value b, called the parity, which is used to record bit flips. Specifically, we note that $X{:}|x\rangle \mapsto |1 \oplus x\rangle$, so we can model bit flips with a single parity bit. We denote the set of states $\mathbb{F}_2 \times \mathbb{F}_2^{(n+k)*}$ as $\mathcal{S}$.

Given a Clifford + T circuit C, written as a sequence of gates over $\{X, Y, Z, P, P^\dagger, H, \text{CNOT}, T, T^\dagger\}$, we first compute a triple $\langle S, Q, H\rangle \in \mathcal{D}$ representing C. $S = \{(c, f) \mid c \in \mathbb{Z}_8, f \in \mathcal{S}\}$ stores the T^k phase factors as linear Boolean functions with parity and multiplicity, $Q = (g_1, g_2, \ldots, g_n) \in \mathcal{S}^n$ tracks the state of each qubit, and $H = (h_1, h_2, \ldots, h_k)$ gives a sequence of Hadamard gates where each entry h_i stores the input and output states, $h_i.Q_I$ and $h_i.Q_O$ respectively. We define the initial state of the circuit as $Q_0 = ((0, x_1), (0, x_2), \ldots, (0, x_m), (0, 0), \ldots, (0, 0))$, which is understood as the state $|x_1 x_2 \cdots x_m\rangle|0\rangle^{\otimes n-m}$. To compute $\langle S, Q, H\rangle$, we use the function $[\![U]\!]{:}\mathcal{D} \to \mathcal{D}$ (Fig. 5) to sequentially update the triple $\langle S, Q, H\rangle$ for each gate U in the circuit, starting from the initial value $\langle \emptyset, Q_0, \emptyset\rangle$.

The T-par algorithm (Algorithm 2) proceeds as follows: after computing $\langle S, Q, H\rangle$, we resynthesize C by iterating through the Hadamard gates in H while updating a partition P of the functions of S that are computable in the current subcircuit. In particular, we divide S into S_P and S_{-P}, where S_P are the already partitioned elements and S_{-P} are those not already partitioned. For a given $h_i = \{Q_I, O_O\}$, for every $(c, f) \in S_{-P}$ we check whether $f \in \text{span}(Q_I)$. If so, we add (c, f) to the current partition using Algorithm 1 with the independence relation $A \subseteq S \in I$ if, and only if, $\text{rank}(Q_I) - \text{rank}(A) \leq n - |A|$. After partitioning, we update S_P and S_{-P} accordingly. The tests for inclusion of each function f in $\text{span}(h_i.Q_I)$ requires $O\left(|S_{-P}| \cdot (n + k)^3\right)$ time, and we can loosely bound the partitioning time as the time to partition the entire set S using Algorithm 1, $O(\left(|S|^3 \cdot (n + k)^3\right)$; since $|S_{-P}| \leq |S|$, the entire step takes $O\left(|S|^3 \cdot (n + k)^3\right)$ time. A tighter analysis is possible, though we omit it as the algorithm is heuristic in nature.

We next iterate through P and for each block $A \in P$, if $f \notin \text{span}(Q_O)$ for some $(c, f) \in A$ we remove A from the partition and synthesize a circuit computing the relevant phase factors. While we defer the discussion of the synthesis procedure for now, we remark that it requires $O\left((n + k)^3\right)$ time. Otherwise, if A is no longer an independent set under

Algorithm 2 T-Parallelization Algorithm

function T-PAR (Clifford + T circuit C)
  C': $= \emptyset$
  $\langle S, Q, H\rangle$: $= [\![C_{|C|}]\!] \cdots [\![C_1]\!] \langle \emptyset, Q_0, \emptyset\rangle$
  Set S_P: $= \emptyset$; S_{-P}: $= S$; P: $= \emptyset$
  for each $1 \leq i \leq k$ **do**
    I: $= \{A \subseteq S \mid \text{rank}(h_i.Q_I) - \text{rank}(A) \leq n - |A|\}$
    for each $(c, f) \in S_{-P}$ **do**
      if $f \in \text{span}(h_i.Q_I)$ **then**
        P: =Partition($(c, f), P, (S_P, I)$)
        S_P: $= S_P \cup \{(c, f)\}$; S_{-P}: $= S_{-P} \setminus \{(c, f)\}$
      end if
    end for
    for each $A \in P$ **do**
      if $i = k$ or $\exists (c, f) \in A$ s.t. $f \notin \text{span}(h_i.Q_O)$ **then**
        Append(C', Synthesize($A, h_i.Q_I, h_i.Q_I$))
        P: $= P \setminus A$
      else if $\text{rank}(h_i.Q_O) - \text{rank}(A) > n - |A|$ **then**
        Choose $(c, f) \in A$ such that
          $\text{rank}(A) = \text{rank}(A \setminus \{(c, f)\})$
        A: $= A \setminus \{(c, f)\}$
        S_P: $= S_P \setminus \{(c, f)\}$; S_{-P}: $= S_{-P} \cup \{(c, f)\}$
      end if
    end for
    Append(C', Synthesize($\emptyset, h_i.Q_I, h_i.Q_O$))
  end for
  return C'
end function
function SYNTHESIZE(A, Q_I, Q_O)
  /* Synthesize a circuit implementing
   U:$|Q_I\rangle \mapsto \omega^{\sum_{(c,(b,f))\in A} c\cdot(b \oplus f(x_1, x_2, \ldots, x_{n+k}))} |Q_O\rangle$ */
  Compute $A' \supseteq A$ s.t. $\text{rank}(A') = \text{rank}(Q_I)$, $|A'| = n$
  Synthesize {CNOT, X} circuit C_1:$|Q_I\rangle \mapsto \left|A'\right\rangle$
  Synthesize $\{Z, P, T\}$ circuit
    C_2:$\left|A'\right\rangle \mapsto \omega^{\sum_{(c,b,f)\in A'} c\cdot(b \oplus f(x_1, x_2, \ldots, x_{n+k}))} \left|A'\right\rangle$
  Synthesize {CNOT, X, H} circuit C_3:$\left|A'\right\rangle \mapsto |Q_O\rangle$
  Return $C_1 C_2 C_3$
end function

the new independence relation ($A \subseteq S \in I$ if, and only if, $\text{rank}(Q_O) - \text{rank}(A) \leq n - |A|$) we remove a linearly dependent element from A and update S_P and S_{-P} so that the deleted element will be repartitioned on the next iteration. As rank(A) and a linearly dependent element can both be found with one application of Gaussian elimination, this step also requires $O\left((n+k)^3\right)$ time, and so the entire loop takes $O\left(|P| \cdot (n+k)^3\right)$ time.

Finally, we synthesize a circuit transforming the current state space Q_I into Q_O, requiring $O\left((n+k)^3\right)$ time, and repeat the process for the next Hadamard. The entire algorithm, shown in Algorithm 2, thus runs in time

$$O\left(|C| \cdot n + k \cdot (n+k)^3 \cdot \left(|S|^3 + |P| + 1\right)\right).$$

As $|C| \cdot n$ is in most cases negligible compared to the $k \cdot (n+k)^3 \cdot |S|^3$ factor, and $|P| \leq |S|$, we describe the runtime as simply $O\left(k \cdot |S|^3 \cdot (n+k)^3\right)$. Moreover, it should be noted that if no repartitioning is done, the runtime is bounded by $O\left(|S|^3 \cdot (n+k)^3\right)$, as each element is partitioned exactly once, rather than the worst case of k times in general.

A. Synthesizing Partitions

We now describe the general synthesis procedure, SYNTHESIZE(A, Q_I, Q_O), from Algorithm 2. The procedure synthesizes a circuit with inputs Q_I and outputs Q_O applying the phases given by a computable partition A of linear Boolean functions.

The algorithm proceeds by first extending A with $n - |A|$ linear Boolean functions to form a set A' with rank equal to rank(Q_I)—this is accomplished by using row operations to reduce A to a subset of Q_I, then adding the vectors in $Q_I \setminus A$. Next we synthesize a circuit computing A' by reducing Q_I and A' to the same basis using Gaussian elimination in $O((n+k)^3)$ time, where addition of two rows corresponds to the application of a CNOT gate, and parity changes correspond to X gates. The circuit reducing Q_I to this basis is applied forwards, while the circuit reducing A' is applied in reverse, giving a circuit mapping $|Q_I\rangle \mapsto \left|A'\right\rangle$. The phase factors are applied by constructing a combination of T, P, and Z gates, corresponding to the relevant coefficients, then the circuit mapping $|Q_I\rangle$ to $\left|A'\right\rangle$ is reversed to compute $|Q_I\rangle$. In the case when $Q_O \neq Q_I$, the corresponding Hadamard gate is applied to compute the output $|Q_O\rangle$.

As alluded to earlier, we see that the synthesis procedure has time complexity $O\left((n+k)^3\right)$, as it requires a constant number (three) of applications of Gaussian elimination. Moreover, the T-depth of the resulting circuit is 1.

VII. RESULTS

We implemented Algorithm 2 in C++[3] and used it to optimize various quantum circuits, mostly arithmetic and reversible ones, from the literature. Individual circuits were written in the standard fault-tolerant universal gate set $\{X, Y, Z, H, P, P^\dagger, \text{CNOT}, T, T^\dagger\}$, using the decompositions found in [2], where applicable, and the Nielsen-Chuang implementation of $\Lambda_k(X)$ gates [30, p. 184]. As most arithmetic circuits are dominated by Toffoli gates, we used the lowest T-depth Toffoli, without ancillae, known [2]. Our implementation additionally removes adjacent Hadamard gates during parsing.

Results are reported in Tables I and II. They were generated in Debian Linux running on a quad-core 64-bit Intel Core i7 2.40 GHz processor and 8 GB RAM. Table I gives gate counts for the circuits before and after optimization. Table II shows T-depths before and after optimization using either[4] 0, n, or unboundedly many ancillae, where n denotes the original number of qubits in the circuit. Circuit T-depths before optimization were computed by parallelizing entire Toffoli gates when possible.

[3]C++ Source code available at http://code.google.com/p/tpar/

[4]In order to give an example of the trade-off between number of ancillae and the T-depth for some nonconstant number of ancillae, we arbitrarily chose to illustrate results with n ancillae. Our implementation allows any other computable value.

TABLE I
T-COUNT BENCHMARKS. WE REPORT THE GATE COUNTS AFTER OPTIMIZING CIRCUITS WITH *T*-PAR, USING NO EXTRA ANCILLAE. *N* SPECIFIES THE NUMBER OF QUBITS. x_C REPORTS THE NUMBER OF CNOT GATES, x_T REPORTS THE NUMBER OF *T* GATES, AND x_U REPORTS THE NUMBER OF OTHER GATES. x' DENOTES THE NUMBER OF GATES AFTER OPTIMIZATION

Benchmark	N	x_C	x_T	x_g	x'_C	x'_T	x'_g	Time (s)	T-count reduction (%)
Grover_5 [21][5]	9	140	140	125	317	52	79	0.001	62.9
Mod 5_4 [27]	5	32	28	9	48	16	12	0.000	42.9
VBE-Adder_3 [38]	10	80	70	20	114	24	23	0.001	65.7
CSLA-MUX_3 [37]	15	90	70	20	425	62	21	0.001	11.4
CSUM-MUX_9 [37]	30	196	196	84	405	112	94	0.005	42.9
QCLA-Com_7 [14]	24	215	203	73	496	95	108	0.003	53.2
QCLA-Mod_7 [14]	26	441	413	132	1,150	249	216	0.008	39.7
QCLA-Adder_{10} [14]	36	273	238	86	737	162	73	0.018	31.9
Adder_8 [36]	24	466	399	126	885	215	180	0.007	46.1
RC-Adder_6 [10]	14	104	77	30	230	63	41	0.001	18.2
Mod-Red_{21} [25]	11	121	119	58	290	73	70	0.001	38.7
Mod-Mult_{55} [25]	9	55	49	36	161	37	33	0.000	24.5
Mod-Adder_{1024} [27]	28	2,005	1,995	570	3,540	1,011	632	0.099	49.3
$\text{Mod-Adder}_{1048576}$ [27]	58	17,310	17,290	4,940	27,378	7,298	5,542	7.290	57.8
$\Lambda_3(X)$ – [3]	5	28	28	8	54	16	12	0.000	42.9
– [30]	5	21	21	6	41	15	9	0.000	28.6
$\Lambda_4(X)$ – [3]	7	56	56	16	90	28	23	0.000	50.0
– [30]	7	35	35	10	63	23	16	0.000	34.3
$\Lambda_5(X)$ – [3]	9	84	84	24	132	40	34	0.001	52.4
– [30]	9	49	49	14	94	31	23	0.000	36.7
$\Lambda_{10}(X)$ – [3]	19	224	224	64	328	100	89	0.004	55.4
– [30]	19	119	119	34	232	71	58	0.002	40.3
GF(2^4)-Mult [29]	12	115	112	32	324	68	27	0.001	39.3
GF(2^5)-Mult [29]	15	179	175	50	494	115	35	0.004	34.3
GF(2^6)-Mult [29]	18	257	252	72	649	150	43	0.008	40.5
GF(2^7)-Mult [29]	21	349	343	98	992	217	36	0.031	36.7
GF(2^8)-Mult [29]	24	468	448	128	1,256	264	40	0.052	41.1
GF(2^9)-Mult [29]	27	575	567	162	1,701	351	44	0.110	38.1
GF(2^{10})-Mult [29]	30	709	700	200	2,176	410	69	0.227	41.4
GF(2^{16})-Mult [29]	48	1,856	1,792	512	6,592	1,040	82	5.079	42.0
GF(2^{32})-Mult [29]	96	7,291	7,168	2,048	33,269	4,128	166	602.577	42.4
GF(2^{64})-Mult [29]	192	28,860	28,672	8,192	180,892	16,448	334	95,447.466	42.6
Average									41.4
Maximum									65.7

The runtimes show that the resynthesis algorithm scales well to large circuits, the largest tested circuit having 192 qubits, 28 672 *T* gates and 8192 Hadamard gates. This stands in contrast to most previous efforts to optimize quantum circuits, which have generally been limited in usefulness to a few qubits and a small number of gates. While our Hadamard gate heuristic adds significant complexity to the algorithm, on average the runtime is actually faster than our previous heuristic of resynthesizing individual {CNOT, *T*} subcircuits, as the greater *T*-count reductions reduce the number of calls to the partitioning algorithm. Thus *T*-par appears to be an efficient and effective heuristic algorithm for the large-scale optimization of fault-tolerant circuits.

All the tested benchmarks show significant reductions in terms of both *T*-count and *T*-depth when no ancillae are added, with average reductions of 41.4% and 62.3% respectively. The algorithm is particularly effective in cases where adjacent Toffoli gates share either controls or targets, as many of the phases cancel—each of the GF(2^m) multipliers share this structure, and as a result show large reductions in *T*-count and *T*-depth. In fact, the *T*-par algorithm will parallelize any GF(2^m) multiplication circuit to *T*-depth 2—by comparison, the minimum *T*-depth achievable using a *T*-depth 1 Toffoli implementation [34] is $12(m-1)$. Those circuits that mix controls and targets between adjacent Toffoli gates are less affected by the optimization, e.g., CSUM-MUX_9, as the Hadamard gates create barriers to *T* parallelization.

We also tested our algorithm's ability to make use of ancillae to optimize *T*-depth (Table II). For each of the benchmark circuits, we applied our algorithm with double the original number of qubits (i.e., *n* extra ancillae where the original circuit used *n* qubits), as well as with unboundedly many qubits. It can be noted that our algorithm usually decreases in running time when ancillae are added, as the addition of ancillae decreases the rate at which partitions have to be modified or created. The algorithm is thus very flexible, and the experimental data illustrates a great potential for exploring space-time trade-off in quantum circuits.

Although *T*-par provides significant reductions to both *T*-count and *T*-depth, in all experiments the number of CNOT gates was increased. While the focus of our work was on the optimization of the more costly *T* gates, there exist algorithms, e.g., [26], [31], that provide better theoretical bounds than Gaussian elimination based linear reversible synthesis. In particular, [31] provides an algorithm

[5]Grover's [21] search is performed with four iterations using the oracle $f(\vec{x}) = \neg x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4 \wedge \neg x_5$.

TABLE II
T-DEPTH BENCHMARKS. WE REPORT THE *T*-DEPTH AFTER NO OPTIMIZATION (ORIGINAL) AND AFTER OPTIMIZATION WITH 0 (SEE TABLE I), *n*, OR UNBOUNDED ANCILLAE

Benchmark	T-depth original	T-depth 0 ancilla	Red. (%)	T-depth N ancilla	Time (s)	Red. (%)	T-depth unbounded ancilla	Time (s)	Red. (%)
Grover_5 [21]	60	18	70.0	14	0.002	76.7	14	0.001	76.7
Mod 5_4 [27]	12	6	50.0	3	0.000	75.0	3	0.000	75.0
VBE-Adder_3 [38]	24	9	62.5	5	0.000	79.2	5	0.000	79.2
CSLA-MUX_3 [37]	21	8	61.9	4	0.004	81.0	4	0.001	81.0
CSUM-MUX_9 [37]	18	9	50.0	4	0.003	77.8	3	0.005	83.3
QCLA-Com_7 [14]	27	11	55.6	7	0.003	74.1	7	0.004	74.1
QCLA-Mod_7 [14]	57	29	49.1	14	0.008	75.4	14	0.010	75.4
QCLA-Adder_{10} [14]	24	11	54.2	6	0.005	75.0	6	0.006	75.0
Adder_8 [36]	69	30	56.5	15	0.007	78.3	15	0.008	78.3
RC-Adder_6 [10]	33	22	33.3	11	0.002	66.7	11	0.001	66.7
Mod-Red_{21} [25]	48	25	47.9	15	0.002	68.8	15	0.011	68.8
Mod-Mult_{55} [25]	15	7	53.3	4	0.000	73.3	4	0.005	73.3
Mod-Adder_{1024} [27]	831	250	69.9	222	0.150	73.3	222	0.112	73.3
$\text{Mod-Adder}_{1048576}$ [27]	7,113	1,910	73.1	1,860	11.863	73.9	1,860	10.586	73.9
$\Lambda_3(X)$ – [3]	12	8	33.3	4	0.001	66.7	4	0.000	66.7
– [30]	9	6	33.3	3	0.000	66.7	3	0.001	66.7
$\Lambda_4(X)$ – [3]	24	13	45.8	8	0.001	66.7	8	0.002	66.7
– [30]	15	9	40.0	5	0.001	66.7	5	0.000	66.7
$\Lambda_5(X)$ – [3]	36	18	50.0	12	0.001	66.7	12	0.004	66.7
– [30]	21	12	42.9	7	0.001	66.7	7	0.001	66.7
$\Lambda_{10}(X)$ – [3]	96	43	55.2	32	0.005	66.7	32	0.032	66.7
– [30]	51	27	47.1	17	0.003	66.7	17	0.008	66.7
$\text{GF}(2^4)$-Mult [29]	36	6	83.3	4	0.001	88.9	2	0.001	94.4
$\text{GF}(2^5)$-Mult [29]	48	9	81.3	5	0.002	89.6	2	0.002	95.8
$\text{GF}(2^6)$-Mult [29]	60	9	85.0	5	0.005	91.7	2	0.003	96.7
$\text{GF}(2^7)$-Mult [29]	72	12	83.3	7	0.026	90.3	2	0.004	97.2
$\text{GF}(2^8)$-Mult [29]	84	13	84.5	7	0.035	91.7	2	0.006	97.6
$\text{GF}(2^9)$-Mult [29]	96	15	84.4	7	0.058	92.7	2	0.010	97.9
$\text{GF}(2^{10})$-Mult [29]	108	16	85.2	7	0.157	93.5	2	0.012	98.1
$\text{GF}(2^{16})$-Mult [29]	180	24	86.7	12	3.128	93.3	2	0.061	98.9
$\text{GF}(2^{32})$-Mult [29]	372	47	87.4	23	644.189	93.8	2	1.246	99.5
$\text{GF}(2^{64})$-Mult [29]	756	94	87.6	44	127,287.329	94.2	2	78.641	99.7
Average			62.3			78.5			80.2
Maximum			87.6			94.2			99.7

to synthesize linear reversible circuits with $\Theta(n^2/\log(n))$ gates, and [26] reports an O(n)-depth algorithm in linear nearest-neighbor architectures—we recently implemented the optimal Patel *et al.* [31] algorithm in T-par , but this did not make a significant difference to the CNOT counts. In the future we would like to explore advanced linear reversible synthesis algorithms to reduce CNOT counts, as well as apply post-optimization by stabilizer circuit optimization algorithms [23]. Another possibility for reducing CNOT gate counts is by assigning weights to sets of linear Boolean functions corresponding to their CNOT complexity, then finding low weight partitions.

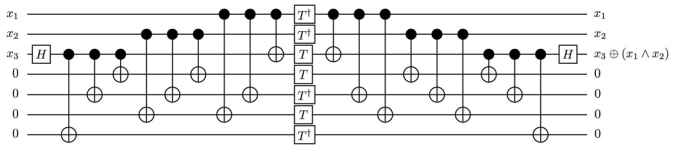


Fig. 6. T-Depth 1 implementation of the Toffoli gate.

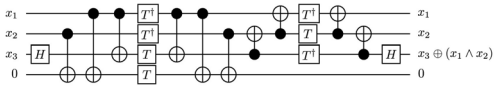


Fig. 7. T-Depth 2 implementation of the Toffoli gate.

A. Applications

As an important application, our experiments include instances of the multiple control Toffoli gates, $\Lambda_k(X)$, which are widely used in the construction of reversible circuits. We report the results using two different implementations—the Barenco *et al.* [3] implementation using $k-2$ ancillae with arbitrary initial states, and the Nielsen-Chuang implementation using $k-2$ ancillae initialized in the state $|0\rangle$ [30]. In both constructions the ancillae are returned to their initial state. Our optimization of the Barenco *et al.* version reduces the T-count from $7(4k-8)$ to $3(4k-8)+4$ and T-depth from $3(4k-8)$ to $4k-8$ with unbounded ancillae, in the instances we tried. Likewise, our optimization of the Nielsen-Chuang implementation reduced the T-count from $7(2k-3)$ to $4(2k-3)+3$ and T-depth from $3(2k-3)$ to $2k-3$. These formulas in fact hold for every $k \geq 3$, a consequence of the simple structure of the circuits. As the two versions use $4k-8$ and $2k-3$ sequential Toffoli gates, respectively, we note that T-par parallelizes each Toffoli to T-depth 1 when sufficiently many ancillae are available. Moreover, the reductions in T-count can be observed to correspond directly to each shared target or control—in this way, T-par achieves the same T-count and

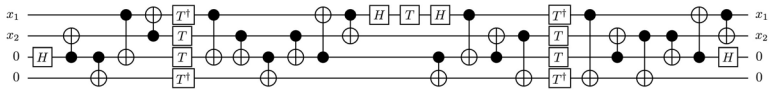


Fig. 8. T-Depth 3 implementation of the controlled-T gate (CNOT stages optimized by templates [28]).

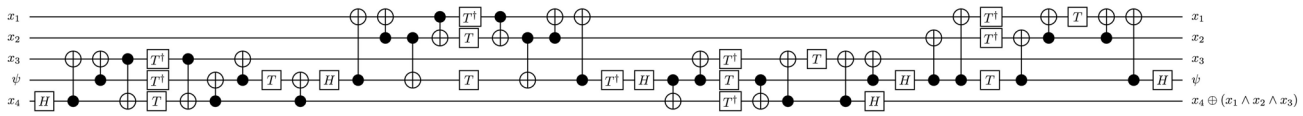


Fig. 9. Optimized implementation of the $\Lambda_3(X)$ gate (CNOT stages optimized by templates [28]).

depth reductions as the controlled gate construction reported in [34], but applies to more general cases and does not require implementing controls with the nonHermitian and unintuitive $\Lambda_2(\pm iX)$ gates.

We also note that our algorithm reproduces many of the previous results regarding the optimization of T-depth. In particular, Fig. 6 shows the circuit produced by running T-par on an implementation of the Toffoli gate. The circuit mirrors the T-depth 1 Toffoli reported in [34]. Moreover, the full range of T-depths possible with different numbers of ancillae can be observed, as seen in Fig. 7. We also show a resynthesized controlled-T gate [2] using one ancilla to reduce the T-depth from 5 to 3 (Fig. 8), and a resynthesized Barenco *et al.* implementation of the $\Lambda_3(X)$ gate using no ancillae (Fig. 9).

VIII. Conclusion

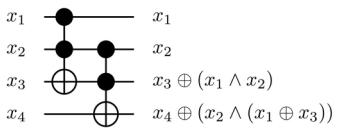
We have described an algorithm for resynthesizing Clifford + T circuits with reduced T-count and depth. The algorithm uses a circuit representation based on linear Boolean functions, allowing T gates to be combined and then parallelized through the use of matroid partitioning algorithms. The algorithm has worst case runtime that is cubic in the number of T gates, qubits, and Hadamard gates, though our experiments show that the algorithm is fast for circuits of considerable size.

Our benchmarks (Tables I and II) show that large gains can be obtained in reducing the T-count and T-depth of quantum circuits. In some cases, T-count was reduced by as much as 65.7%, while the T-depth could be reduced by up to 87.6% without ancillae. Furthermore, the benchmarks illustrate that ancillae can be used to parallelize T gates further, and given the runtimes reported the algorithm can be seen to provide substantial flexibility in exploring the trade-off between ancilla usage and T-depth. In the most extreme case we were able to reduce the T-depth of GF(2^m)-Mult circuits from $12(m-1)$ to a constant of 2, using unbounded ancillae. While the benchmarks were all arithmetic or otherwise reversible operations, such operations typically require the majority of the resources in circuits for quantum algorithms of interest [22].

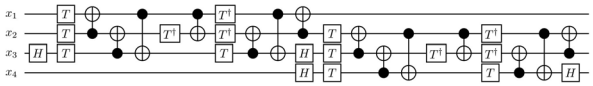
We close by noting that as a consequence of the T-par algorithm, reducing the number of terms in the mixed arithmetic polynomials describing the phase corresponds directly to reducing the T-complexity of quantum circuits; in fact, it was observed that minimization of T-count in $\{\text{CNOT}, T\}$ circuits is equivalent to minimizing the number of odd coefficients in the phase. A natural avenue of future work is then to develop methods by which the phase polynomials themselves can be optimized for T-count and depth. This paper also represents the first instance, to the authors' knowledge, of the use of sum over paths style representations in quantum circuit synthesis and optimization—we hope this will inspire others to find new applications of such representations to synthesis and optimization of quantum circuits. Efficient synthesis of linear reversible circuits is another important direction that would directly contribute to improving the results of this paper.

Appendix
Parallelizing $(\Lambda_2(X) \otimes I)(I \otimes \Lambda_2(X))$

In this section we illustrate the workings of the T-par algorithm using the following circuit:



We first expand this circuit by using the T-depth 3 Toffoli gate implementation [2]



Next we compute $\langle S, Q, H\rangle$ by applying the function $[\![\cdot]\!]$ to each gate in sequence, starting with $\langle \emptyset, (x_1, x_2, x_3, x_4), \emptyset\rangle$.

The result is

$$S = \left\{ \begin{array}{c} x_1, 2x_2, x_5, 7(x_1 \oplus x_2), 7(x_1 \oplus x_5), \\ 7(x_2 \oplus x_5), (x_1 \oplus x_2 \oplus x_5), x_6, x_7, 7(x_2 \oplus x_6), \\ 7(x_2 \oplus x_7), 7(x_6 \oplus x_7), (x_2 \oplus x_6 \oplus x_7) \end{array} \right\}$$

$$Q = (x_1, x_2, x_6, x_8)$$

$$H = (h_1, h_2, h_3, h_4) \quad \text{where}$$

$$h_1 = \{Q_I = (x_1, x_2, x_3, x_4), Q_O = (x_1, x_2, x_5, x_4)\}$$

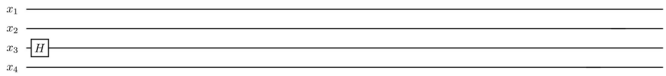
$$h_2 = \{Q_I = (x_1, x_2, x_5, x_4), Q_O = (x_1, x_2, x_6, x_4)\}$$

$$h_3 = \{Q_I = (x_1, x_2, x_6, x_4), Q_O = (x_1, x_2, x_6, x_7)\}$$

$$h_4 = \{Q_I = (x_1, x_2, x_6, x_7), Q_O = (x_1, x_2, x_6, x_8)\}.$$

Starting with h_1, we see that the terms $x_1, 2x_2, 7(x_1 \oplus x_2)$ are computable, so we partition them into blocks satisfying

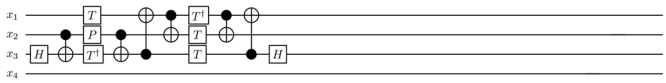
$4 - \text{rank}(A) \leq 4 - |A|$, giving $P = \{\{x_1, 2x_2\}, \{7(x_1 \oplus x_2)\}\}$. As neither partition will become uncomputable after h_1, we simply apply the first Hadamard gate:



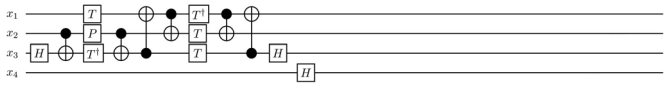
At the second Hadamard gate, the path variable x_5 is available, so $x_5, 7(x_1 \oplus x_5), 7(x_2 \oplus x_5), x_1 \oplus x_2 \oplus x_5$ are now computable. We add them to the partition to get

$$P = \left\{ \begin{matrix} \{x_1, 2x_2, 7(x_2 \oplus x_5)\}, \{7(x_1 \oplus x_2)\} \\ \{x_1 \oplus x_2 \oplus x_5, 7(x_1 \oplus x_5), x_5\} \end{matrix} \right\}.$$

We trivially see that $x_2 \oplus x_5 \notin \text{span}(\{x_1, x_2, x_6, x_4\})$ and likewise neither is x_5, so we synthesize a circuit computing the partitions $\{x_1, 2x_2, 7(x_2 \oplus x_5)\}$ and $\{x_1 \oplus x_2 \oplus x_5, 7(x_1 \oplus x_5), x_5\}$, while allowing $\{7(x_1 \oplus x_2)\}$ to move past the second Hadamard gate.



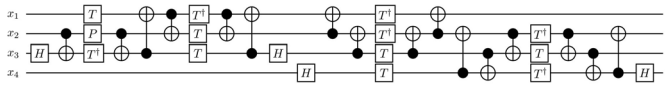
Again, x_6 is now available, so we insert the newly computable terms $x_6, 7(x_2 \oplus x_6)$ into the current partition $P = \{\{7(x_1 \oplus x_2)\}\}$ to get $P = \{\{7(x_1 \oplus x_2), x_6, 7(x_2 \oplus x_6)\}\}$. As the single partition block will still be computable after applying h_3, we apply the next Hadamard gate:



We now add the last of the terms $x_7, 7(x_6 \oplus x_7)$, and $x_2 \oplus x_6 \oplus x_7$ to the partition, giving

$$\left\{ \begin{matrix} \{7(x_1 \oplus x_2), x_6, 7(x_2 \oplus x_6), x_7\}, \\ \{7x_2 \oplus x_7), 7(x_6 \oplus x_7), x_2 \oplus x_6 \oplus x_7\} \end{matrix} \right\}.$$

As both partitions contain the value x_7 which will be uncomputable after h_4, we apply the remaining partitions, followed by the last Hadamard gate:



The final circuit, shown above, reduces the original circuit by two T gates (from 14 to 12) and by two levels of T-depth (from 6 to 4).

Note that the partitions used in the above are minimal, but are not the partitions Algorithm 1 actually produces. Instead, these partitions have been created to best demonstrate the algorithm. Additionally, for simplicity, we described states without parity, as there are no bit flip gates in this example.

ACKNOWLEDGMENT

The authors would like to thank M. Rötteler and B. Cunningham for many helpful discussions, as well as the anonymous reviewers for their comments.

REFERENCES

[1] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," *Quantum Inf. Comput.*, vol. 6, no. 2, pp. 97–165, 2006.

[2] M. Amy, D. Maslov, M. Mosca, and M. Rötteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 6, pp. 818–830, Jun. 2013.

[3] A. Barenco *et al.*, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, pp. 3457–3467, Mar. 1995.

[4] T. Beth and M. Rötteler, "Quantum algorithms: Applicable algebra and quantum physics," in *Quantum Information* (Springer Tracts in Modern Physics), vol. 173. Berlin, Germany: Springer, 2001, pp. 96–150.

[5] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, "Strong resilience of topological codes to depolarization," *Phys. Rev. X*, vol. 2, Apr. 2012, Art. ID 021004.

[6] J. W. Britton *et al.*, "Engineered 2D Ising interactions in a trapped-ion quantum simulator with hundreds of spins," *Nature*, vol. 484, pp. 489–492, Apr. 2012.

[7] K. R. Brown *et al.*, "Single-qubit-gate error below 10^{-4} in a trapped ion," *Phys. Rev. A*, vol. 84, Sep. 2011, Art. ID 030303.

[8] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal Clifford gates and noisy ancillas," *Phys. Rev. A*, vol. 71, Feb. 2005, Art. ID 022316.

[9] J. M. Chow *et al.*, "Universal quantum gate set approaching fault-tolerant thresholds with superconducting qubits," *Phys. Rev. Lett.*, vol. 109, Aug. 2012, Art. ID 060501.

[10] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. Petrie Moulton. (2004, Oct.). *A new quantum ripple-carry addition circuit*, ArXiv e-prints [Online]. Available: http://arxiv.org/abs/quant-ph/0410184

[11] C. M. Dawson *et al.*, "Quantum computing and polynomial equations over the finite field $\mathbb{Z}_2$," *Quantum Inf. Comput.*, vol. 5, pp. 102–112, Mar. 2005.

[12] P. A. M. Dirac, "A new notation for quantum mechanics," in *Proc. Cambridge Philos. Soc.*, vol. 35, no. 3, pp. 416–418, 1939.

[13] D. P. DiVincenzo, "Two-bit gates are universal for quantum computation," *Phys. Rev. A*, vol. 51, no. 2, pp. 1015–1022, Feb. 1995.

[14] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," *Quantum Inf. Comput.*, vol. 6, pp. 351–369, Jul. 2006.

[15] J. Edmonds, "Minimum partition of a matroid into independent subsets," *J. Res. Nat. Bureau Standards*, vol. 69B, pp. 67–72, Jan. 1965.

[16] A. G. Fowler. (2012). *Time-optimal quantum computation*, ArXiv e-prints [Online]. Available: http://arxiv.org/abs/1210.4626

[17] A. G. Fowler, A. M. Stephens, and P. Groszkowski, "High-threshold universal quantum computation on the surface code," *Phys. Rev. A*, vol. 80, Nov. 2009, Art. ID 052312.

[18] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, "Toward practical classical processing for the surface code," *Phys. Rev. Lett.*, vol. 108, 2012, Art. ID 180501.

[19] D. Gottesman, "Theory of fault-tolerant quantum computation," *Phys. Rev. A*, vol. 57, pp. 127–137, Jan. 1998.

[20] D. Gottesman and I. L. Chuang, "Quantum teleportation is a universal computational primitive," *Nature*, vol. 402, no. 6760, pp. 390–393, 1999.

[21] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. Symp. Theory Comput.*, 1996, pp. 212–219.

[22] (2013). *IARPA Quantum Computer Science Program* [Online]. Available: http://www.iarpa.gov/Programs/sso/QCS/qcs.html

[23] V. Kliuchnikov and D. Maslov, "Optimization of Clifford circuits," *Phys. Rev. A*, vol. 88, Nov. 2013, Art. ID 052307.

[24] S. Lloyd, "Universal quantum simulators," *Science*, vol. 273, no. 5278, pp. 1073–1078, 1996.

[25] I. L. Markov and M. Saeedi, "Constant-optimized quantum circuits for modular multiplication and exponentiation," *Quantum Inf. Comput.*, vol. 12, nos. 5–6, pp. 361–394, 2012.

[26] D. Maslov, "Linear depth stabilizer and quantum Fourier transformation circuits with no auxiliary qubits in finite-neighbor quantum architectures," *Phys. Rev. A*, vol. 76, Nov. 2007, Art. ID 052310.

[27] D. Maslov. (2013, Oct.). *Reversible Logic Synthesis Benchmarks Page* [Online]. Available: http://webhome.cs.uvic.ca/dmaslov/

[28] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, "Quantum circuit simplification and level compaction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 3, pp. 436–444, Mar. 2008.

[29] D. Maslov, J. Mathew, D. Cheung, and D. K. Pradhan, "An $O(m^2)$-depth quantum algorithm for the elliptic curve discrete logarithm problem over GF(2^m)," *Quantum Inf. Comput.*, vol. 9, no. 7, pp. 610–621, 2009.

[30] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[31] K. N. Patel, I. L. Markov, and J. P. Hayes, "Optimal synthesis of linear reversible circuits," *Quantum Inf. Comput.*, vol. 8, no. 3, pp. 282–294, 2008.
[32] C. Rigetti *et al.*, "Superconducting qubit in a waveguide cavity with a coherence time approaching 0.1 ms," *Phys. Rev. B*, vol. 86, Sep. 2012, Art. ID 100506.
[33] T. Rudolph, "Simple encoding of a quantum circuit amplitude as a matrix permanent," *Phys. Rev. A*, vol. 80, Nov. 2009, Art. ID 054302.
[34] P. Selinger, "Quantum circuits of *T*-depth one," *Phys. Rev. A*, vol. 87, Apr. 2013, Art. ID 042302.
[35] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. Found. Comput. Sci.*, Santa Fe, NM, USA, 1994, pp. 124–134.
[36] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," *Quantum Inf. Comput.*, vol. 10, no. 9, pp. 872–890, 2010.
[37] R. Van Meter and K. M. Itoh, "Fast quantum modular exponentiation," *Phys. Rev. A*, vol. 71, May. 2005, Art. ID 052320.
[38] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Phys. Rev. A*, vol. 54, pp. 147–153, Jul. 1996.
[39] H. Whitney, "On the abstract properties of linear dependence," *Amer. J. Math.*, vol. 57, no. 3, pp. 509–533, 1935.

Matthew Amy was born in Toronto, ON, Canada, in 1989. He received the B.Math. degree in computer science with a minor in pure mathematics from the University of Waterloo, Waterloo, ON, Canada, in 2011, where he received the M.Math. degree in computer science and quantum information in 2013. He is currently pursuing the Doctorate in computer science at the University of Toronto, Toronto, ON, Canada.

His current research interests include programming languages, formal methods, quantum circuit synthesis, and optimization.

Mr. Amy held an NSERC Undergraduate Student Research Award in formal verification at the University of Toronto while doing undergraduation, and is continuing that work in the Ph.D. degree.

Dmitri Maslov received the M.Sc. degree in mathematics from Lomonosov's Moscow State University, Moscow, Russia, in 2000, and the M.Sc. and Ph.D. degrees in computer science from the University of New Brunswick, Fredericton, NB, Canada, in 2002 and 2003, respectively.

He is a Program Director with the Division of Computing and Communication Foundations, Directorate for Computer and Information Science and Engineering, National Science Foundation, Arlington, VA, USA. His current research interests include quantum circuits and architectures, quantum algorithms, quantum information processing, logic synthesis, and reversible logic.

Michele Mosca received the Doctorate in mathematics from the University of Oxford, Oxford, U.K., in 1999, on the topic of Quantum Computer Algorithms.

He returned to Waterloo in 1999 as a faculty member. He is a Co-Founder and Deputy Director of the Institute for Quantum Computing, University of Waterloo, Waterloo, ON, Canada, a Professor with the Department of Combinatorics and Optimization of the Faculty of Mathematics, and a Founding Member of the Waterloo Perimeter Institute for Theoretical Physics, Waterloo, ON, Canada. His current research interests include quantum algorithms and complexity, tools for optimizing the implementation of quantum circuits, and the development of cryptographic tools that will be safe against quantum technologies. His work is published widely in top journals, and he has co-authored the respected textbook *An Introduction to Quantum Computing* (OUP).

Dr. Mosca has won numerous academic awards and honors, including the 2010 Canada's Top 40 Under 40, the Premier's Research Excellence Award in 2000–2005, the Fellow of the Canadian Institute for Advanced Research, since 2010, the Canada Research Chair in Quantum Computation in 2002–2012, the University Research Chair at the University of Waterloo, since 2012, and the Queen Elizabeth II Diamond Jubilee Medal in 2013.