# The phase/state duality in reversible circuit design

Matthew Amy* and Neil J. Ross†
*Department of Mathematics and Statistics*
*Dalhousie University, Halifax, Canada*

The reversible implementation of classical functions accounts for the bulk of most known quantum algorithms. As a result, a number of reversible circuit constructions over the Clifford+$T$ gate set have been developed in recent years which use both the state and phase spaces, or $X$ and $Z$ bases, to reduce circuit costs beyond what is possible at the strictly classical level. We study and generalize two particular classes of these constructions: relative phase circuits, including Giles and Selinger's multiply-controlled $iX$ gates and Maslov's 4 qubit Toffoli gate, and measurement-assisted circuits, including Jones' Toffoli gate and Gidney's temporary logical-AND. In doing so, we introduce general methods for implementing classical functions up to phase and for measurement-assisted termination of temporary values. We then apply these techniques to find novel $T$-count efficient constructions of some classical functions in space-constrained regimes, notably multiply-controlled Toffoli gates and temporary products.

## I. INTRODUCTION

The reversible implementation of classical functions on a quantum computer is crucial to many quantum algorithms, including Grover's search algorithm [1] and Shor's factoring algorithm [2]. In such algorithms, oracles for classical subroutines account for the bulk of the total circuit volume. As a result, the optimization of quantum circuits for classical reversible functions is central to the resource-efficient implementation of quantum algorithms.

Two complexity measures in the design of quantum circuits are the circuit *time* or *depth* and *space* or *width*. The former corresponds roughly to the number of gates that appear in the circuit and is sometimes weighted to account for the fact that some gates are more costly than others, while the latter is given by the number of qubits used in the the circuit. Changes to these costs are amplified in fault-tolerant contexts; each additional logical qubit requires a large number of physical qubits, while longer computations require more error correction, further increasing the physical footprint of a quantum algorithm.

Standard techniques for synthesizing reversible circuits can lead to massive space overheads, as they rely on ancillary qubits to hold intermediate values. This overhead can be often be mitigated by *uncomputing* intermediate values once they are no longer useful, at the expense of extra gates. This space-time trade-off is explored at the level of reversible circuit synthesis through pebble games [3].

Over the past decade, significant effort has been devoted to efficiently implementing classical functions over the Clifford+$T$ gate set, motivated by the fact that Clifford+$T$ gates are well-suited for fault-tolerant quantum computation [4]. In this context the number of $T$ gates in a circuit, its *$T$-count*, often dominates the cost in time due to the difficulty of implementing the $T$ gate in a fault-tolerant manner. These recent efforts resulted in a variety of optimized Clifford+$T$ implementations of reversible gates, many of which leverage the *phase space*, or the $X$ basis, to go beyond optimizations possibly purely in the *state space*, or $Z$ basis. By these we mean information encoded in either the phase of a quantum state or the computational basis state, respectively. As information in the phase and state can be freely exchanged and independently operated on, we refer to this as the *phase/state duality*.

In the present work we study further applications of the phase/state duality to reversible circuit design, generalizing several recent constructions:

- the multiply-controlled $iX$ gates of [5] and [6],
- the measurement-assisted Toffoli of [7],
- the relative phase Toffoli-4 of [8], and
- the temporary logical-AND of [9].

We then apply these methods to introduce new circuit designs over the Clifford+$T$ gate set which improve the cost primarily of space-constrained implementations of oracles for classical functions.

Table I gives an overview of the novel circuits we give, as well as the best-known constructions when possible. While many of these implement classical functions up to relative phase, all constructions with relative phases *in the controls and/or target* can

* matt.amy@dal.ca
† neil.jr.ross@dal.ca

| Gate | Ancillary state | $T$-count | Valid | Notes | Ref. |
|---|---|---|---|---|---|
| $U_{f\cdot g}$ | $\|00\rangle$ | $2\tau(U_f)+\tau(U_g)+8$ | – | | 1 |
| $U_{f\cdot g}$ | – | $2\tau(U_f)+2\tau(U_g)+4$ | – | Relative phase in the controls | 4 |
| $U_{f\cdot g}$ | – | $2\tau(U_f)+\tau(U_g)+4$ | – | Relative phase in the controls & target | 5 |
| $\Lambda_k(X)$ | $\|z\rangle$ | $16(k-1)$ | $k\geq 6$ | Prior art | [8] |
| $\Lambda_k(X^{\bullet})$ | $\|z\rangle$ | $8(k-2)+4$ | $k\geq 2$ | Relative phase in the controls & ancilla | 2 |
| $\Lambda_k(X)$ | $\|z\rangle$ | $16(k-2)$ | $k\geq 4$ | | 3 |
| $\Lambda_k(X)$ | $\|0\rangle$ | $16(k-3)$ or $16(k-3)+4$ | $k\geq 4$ | Measurement-assisted | |
| $\Lambda_k(iX)$ | – | $16(k-2)+4$ | $k\geq 6$ | Prior art; Relative phase in the controls | [5, 8] |
| $\Lambda_k(iX)$ | – | $16(k-3)+4$ | $k\geq 4$ | Relative phase in the controls | 7 |
| $\Lambda_k(X^{\bullet})$ | – | $16(k-4)+4$ | $k\geq 5$ | Relative phase in the controls | 9 |
| $\Lambda_k(X^{\star})$ | – | $8(k-2)$ | $k\geq 3$ | Relative phase in the controls & target | 8 |
| $\Lambda_k(X^{\star})$ | $\|0\rangle^{\otimes m}$ | $4m+8(k-m-2)$ | $k\geq 5$ | Relative phase in the controls & target | |
| $U_{f_k}$ | $\|z\rangle$ | $8(k-1)$ | $k\geq 2$ | | |
| $U_{f_k}$ | – | $4(k-1)$ | $k\geq 2$ | Relative phase in the controls & target | 6 |
| 3-AND | $\|0\rangle$ | 8 | – | Prior art; Relative phase in the controls | [8] |
| 3-AND$^{\dagger}$ | – | 3 or 4 | – | Relative phase; Measurement-assisted | 14 |
| $k$-AND | $\|0\rangle$ | $16(k-3)+4$ | $k\geq 4$ | – | 12 |
| $k$-AND$^{\dagger}$ | – | 0 or $16(k-4)+4$ | $k\geq 6$ | Measurement-assisted | 13 |
| $k$-AND | $\|0\rangle$ | $8(k-2)$ | $k\geq 3$ | Relative phase in the controls | 10 |
| $k$-AND$^{\dagger}$ | – | $8(k-4)$ or $8(k-4)+4$ | $k\geq 4$ | Relative phase; Measurement-assisted | 11 |

TABLE I. $T$ count scaling for various reversible functions and gates. $\tau(U_f)$ and $\tau(U_g)$ give the $T$-counts of implementations of $U_f$ and $U_g$, respectively. References to explicit circuits are given where possible.

be used as drop-in replacements for matched compute/uncompute pairs.

Our contributions include an ancilla-free $k$-control Toffoli up to a relative phase in both the controls and the target with $T$-count $8(k-2)$, improving the best known construction by a factor of roughly 50%. We also show that if this gate is used to initialize a temporary product of $k$ bits as in [9], it can be terminated with the aid of measurement and classical control with at most $8(k-4)+4$ $T$ gates. Combined, these constructions give a method of temporarily instantiating a logical product of $k$ bits with total $T$-count at most $16(k-3)+4$ and no ancillas, besides the one used to store the product. Previous techniques require $32(k-2)+8$ $T$ gates, a reduction of over 50% compared to the state of the art.

We also give novel space-constrained constructions for the $k$-control Toffoli gate with reduced $T$-count and for the efficient multiplication of classical oracles up to phase. We additionally show that there exist classes of Boolean functions of degree $k$ which can be implemented up to relative phase and without ancillas using $4(k-1)$ $T$ gates. These constructions match or improve on the $T$-count of the best known generic method [10] which uses $O(k)$ ancillas, and have potential applications to automated and LUT-based [11] synthesis of reversible circuits. More broadly, these constructions show that there exist functions for which existing techniques are not able to reduce the $T$-count through the addition of ancillas.

## II. BACKGROUND

### A. Quantum oracles

It is well-known that, in the presence of ancillas, the gate set

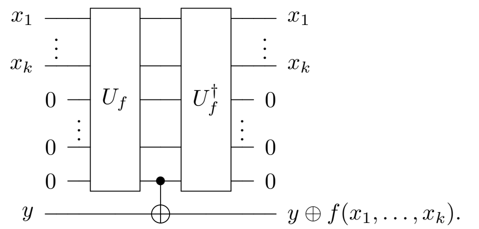
$$\{X, \Lambda_1(X), \Lambda_2(X)\},$$

consisting of the NOT, controlled-NOT, and Toffoli gates, is *universal for classical computing* [12]. That is, for any Boolean (or classical) function $f:\mathbb{Z}_2^n\to\mathbb{Z}_2^m$, there exists a circuit over the gate set $\{X, \Lambda_1(X), \Lambda_2(X)\}$ which implements a unitary $U_f$ whose action on the computational basis is described by

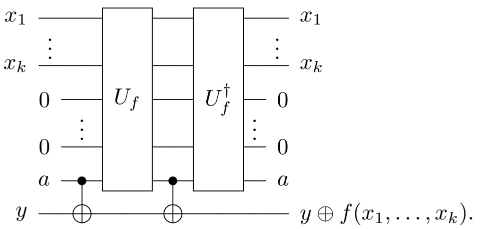
$$|\mathbf{x}\rangle\,|0\cdots 0\rangle\,|y\rangle \mapsto |\mathbf{x}\rangle\,|g_1(\mathbf{x})\cdots g_k(\mathbf{x})\rangle\,|y\oplus f(\mathbf{x})\rangle$$

where $\mathbf{x}=x_1x_2,\ldots,x_n$ and the $g_i$ are some Boolean functions $g_i:\mathbb{Z}_2^n\to\mathbb{Z}_2$. The unitary $U_f$ is an *oracle* for $f$. The qubits beginning in the $|0\rangle$ state are *clean* ancillas. The values $g_i(x)$ used in the process of computing $f$ are *temporary values* and are often referred to as *garbage*.

To reclaim the space used for temporary values, the final result can be copied on an additional ancilla, and the circuit for $U_f$ can be run in reverse. This *uncomputes* the temporary values that are no longer needed, thereby *cleaning up* the garbage. This technique, colloquially known as the *Bennett trick*, is shown below:

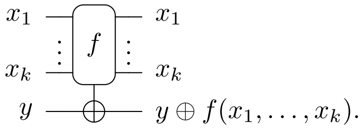


Note that, in the circuit above, the target of $U_f$ is in the $|0\rangle$ state. One can relax this requirement at the cost of an extra $\Lambda_1(X)$ gate:



In this case we say that the (uninitialized) ancilla is *dirty*.

We use rounded boxes to denote oracles which do not leave any garbage and do not modify their inputs:



### B. Generalized permutations

A *(unitary) generalized permutation matrix* is a permutation matrix whose nonzero entries are elements of $\mathbb{T} = \{z \in \mathbb{C} \mid |z| = 1\}$, the group of complex numbers of unit length. Every generalized permutation matrix $U$ can be factored as the product of a permutation matrix $P$ and a diagonal matrix $D$, i.e. $U = PD$. Note that, since $D' = PDP^\dagger$ is also diagonal, $U$ can alternatively be factored as

$$U = PD = PDP^\dagger P = D'P.$$

We will sometimes leverage this kind of quasi-commutation throughout the remainder of this paper. Restricting the nonzero entries of generalized permutation matrices to $m$-th roots of unity yields the *generalized symmetric group* $\mathcal{S}(m, n)$.

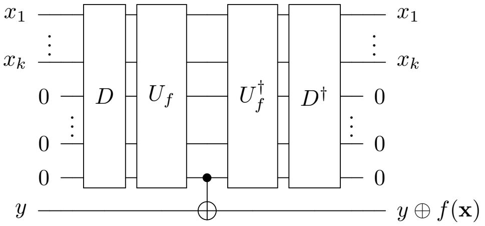
### C. Relative phases

Generalized permutations occur in quantum computing as *relative-phase* implementations of classical functions. In particular, a generalized permutation $\widetilde{U_f}$ acting on the computational basis as

$$\widetilde{U_f} : |\mathbf{x}\rangle\,|0\cdots0\rangle\,|y\rangle \mapsto e^{ig(\mathbf{x},y)}\,|\mathbf{x}\rangle\,|0\cdots0\rangle\,|y \oplus f(\mathbf{x})\rangle$$

is called a *relative-phase* implementation or oracle for $f$ and $e^{ig(\mathbf{x},y)}$ is called the *phase*. If $g(\mathbf{x}, y) = g(\mathbf{x}, y')$ for all $y' \in \mathbb{Z}_2$, we say that the phase *depends only on the controls*. Otherwise, we say that the phase *depends on the controls and the target*.

It can be observed [12] that a relative phase implementation suffices to compute any temporary value in a reversible circuit or oracle. For example, the circuit below uses the Bennett trick and a relative phase implementation $\widetilde{U_f} = U_f D$ of $f$ where $D$ is some diagonal unitary to construct a *phase-free* oracle for $f$.



The correctness of the circuit can be established through the quasi-commutation noted above. Indeed, we have $U_f D = D' U_f$ for some diagonal matrix $D'$. The diagonal gates can thus be moved inwards and cancelled, since diagonal matrices commute with controls.

More generally, an oracle $U_f$ in some compute/uncompute pair $U_f^\dagger U U_f$ may be implemented up to a relative phase on qubit $i$ whenever the internal computation $U$ is globally constant on the *state* space of qubit $i$. In particular, $U$ is globally constant on the state space of the first qubit if

$$U(|x_1\rangle \otimes |x_2 \cdots x_n\rangle) = e^{ig(\mathbf{x})}\,|x_1\rangle \otimes U_{x_1}\,|x_2 \cdots x_n\rangle$$

for any $\mathbf{x} \in \mathbb{Z}_2^n$. In practice, this accounts for the vast majority of cases where a temporary value is computed and later uncomputed. Additional discussion can be found in Appendix A.

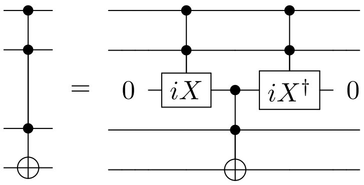
### D. Phase space optimizations

The observation that the phase space can be used to optimize reversible circuits through the use of generalized permutations dates back to Norman Margolus [13]. Margolus noted that the Toffoli gate can be implemented up to a phase with just 3 two-qubit gates, rather than the otherwise minimal 5. DiVincenzo and Smolin [13] found similar optimizations using relative phases, albeit with a stern warning that this "is often a dangerous thing to do." This idea was explored further by Barenco *et al.* [12], noting that implementing up to phase is generally a safe thing to do *as long as only classical computations are performed before uncomputing the phase.*

The idea of using the phase space to optimize reversible circuits experienced a recent resurgence, in part due to Peter Selinger's relative phase Clifford+$T$ implementation of the Toffoli gate. In particular, Selinger introduced the doubly-controlled $iX$ gate

$$\Lambda_2(iX) : |x_1\rangle\,|x_2\rangle\,|y\rangle \mapsto i^{x_1x_2}\,|x_1\rangle\,|x_2\rangle\,|y \oplus (x_1x_2)\rangle\,,$$

which can be implemented with only 4 $T$ gates (see Figure 1) as opposed to the optimal 7 $T$ gates needed to implement the Toffoli gate on the nose in the absence of ancillas and measurements [14]. Since the erroneous phase $i^{x_1x_2}$ is irrelevant to computations in the state space, the doubly controlled $iX$ gate can be used interchangeably with a Toffoli gate to compute a temporary logical AND of two bits. When this temporary value is later uncomputed, the extraneous phase is also uncomputed, as below:



Automated methods were later developed which achieve the same $T$-counts by identifying the redundant $i^{x_1x_2}$ phase terms when regular Toffoli gates are used instead [15, 16], mitigating the need for explicit relative phase constructions.

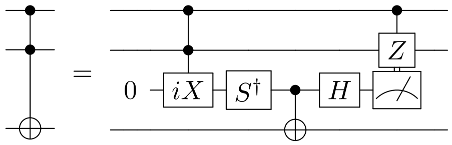
Cody Jones [7] used the $iX$ gate to implement a full Toffoli gate using only 4 $T$ gates, a measurement and a classically controlled Clifford correction. The main insights were (i) that the phase $i^{x_1x_2}$ could be corrected with a single $S^\dagger$ gate if the target of the $\Lambda_2(iX)$ gate is in the $|0\rangle$ state, and (ii) that the state $|(x_1x_2)\rangle$ can be uncomputed with a measurement and classically-controlled Clifford corrections. Explicitly, with a single clean ancilla, the product $|(x_1x_2)\rangle$ can first be computed using a $\Lambda_2(iX)$ gate:

$$\Lambda_2(iX)\,|x_1\rangle\,|x_2\rangle\,|0\rangle = i^{x_1x_2}\,|x_1\rangle\,|x_2\rangle\,|(x_1x_2)\rangle\,.$$

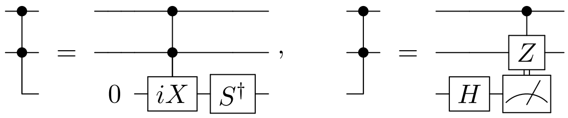
The $i^{x_1x_2}$ phase can then be immediately corrected by applying an $S^\dagger$ gate to the ancilla. And the product $|(x_1x_2)\rangle$ can be copied into the target register using a $\Lambda(X)$ gate and uncomputed from the ancilla. Rather than uncomputing the state $|(x_1x_2)\rangle$, Jones noted that it can be traded for a phase via a Hadamard gate, since:

$$H\,|(x_1x_2)\rangle = \frac{1}{\sqrt{2}}\sum_{z\in\mathbb{Z}_2}(-1)^{x_1x_2z}\,|z\rangle\,.$$

While correcting this phase with a doubly-controlled $Z$ gate would require 7 $T$ gates, the ancilla can be measured first and then the resulting phase — 1 if the measurement result is 0 or $(-1)^{x_1x_2}$ otherwise — can be subsequently corrected. In the case of a measured value of 1, a classically-controlled $\Lambda_1(Z)$ gate is all that is needed to correct to the phase. The resulting circuit is shown below.



As Jones's circuit involves an ancilla, measurement, and a classically controlled correction, its use in reversible circuit design remained somewhat limited until Craig Gidney [9] observed that by delaying the uncomputation of the temporary product $x_1x_2$, the $T$-cost of uncomputing certain temporary values in a reversible circuit can be reduced to 0. Gidney introduced the *temporary logical-AND* construction by explicitly separating Jones's Toffoli into a $T$-count 4 circuit for initializing an ancilla with a logical AND of two bits and a corresponding *termination* circuit with $T$-count 0. We use the term termination, corresponding to the transformation $|x_1\rangle\,|x_2\rangle\,|(x_1x_2)\rangle \mapsto |x_1\rangle\,|x_2\rangle$ , to denote the fact that the circuit is non-unitary. Both circuits are shown below.



This construction gives rise to a $k$-controlled Toffoli gate with $4(k-1)$ $T$ gates using $k-1$ clean ancillas, as well as implementations of classical functions

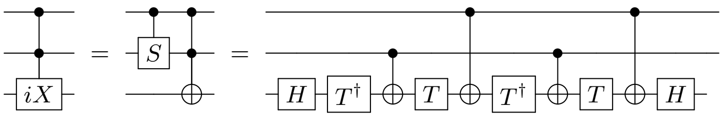


FIG. 1. The doubly-controlled $iX$ gate [6].

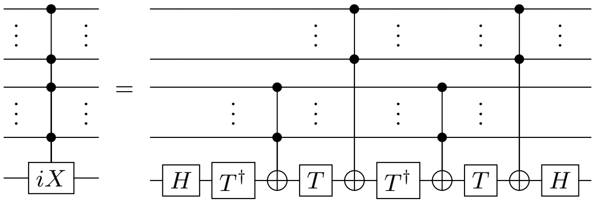


FIG. 2. A circuit implementing a multiply-controlled $iX$ gate [5].

$f$ with multiplicative complexity [17] $c_\wedge(f)$ using at most $4c_\wedge(f)$ $T$ gates and $c_\wedge(f)$ ancillas [10]. More recently, Berry *et al.* [18] designed a measurement assisted termination circuit for QROM states, while Soeken and Roetteler [19] studied similar measurement assisted termination in the context of Clifford plus arbitrary single-qubit rotations. Gidney also explored pebble game strategies using measurement-assisted uncomputation in [20].

In a complementary direction, other efficient generalized permutations were discovered following [6]. Giles and Selinger [5] gave an implementation of the multi-qubit $iX$ gate without ancillas, shown in Figure 2. Dmitri Maslov [8] later looked at implementations of the doubly- and triply-controlled Toffoli gates up to other relative phases. One of Maslov's discoveries was a relative phase triply-controlled Toffoli gate which is shown in Figure 3. The circuit, implementing the generalized permutation

$$|\mathbf{x}\rangle|y\rangle \mapsto i^{x_1x_2+x_1x_2x_3}(-1)^{x_1x_2y}|\mathbf{x}\rangle|y\oplus(x_1x_2x_3)\rangle,$$

reduces the space usage to compute a product of three bits with only 8 $T$ gates, at the expense of a target-dependent phase of $i^{x_1x_2+x_1x_2x_3}(-1)^{x_1x_2y}$.

By using this relative phase 4-qubit Toffoli, as well as other generalized permutations, novel implementations of reversible functions with reduced space usage were given in [8]. In the case of the Toffoli gate of Barenco *et al.* [12, Lemma 7.2], these techniques reduced the $T$-count from $12n+O(1)$ to $8n+O(1)$.

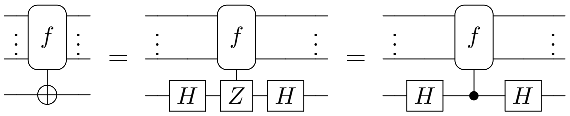
## III. CIRCUITS WITH ANCILLAS

The constructions discussed in the previous section use the phase/state duality in a variety of ways to design efficient circuits. Typically, these circuits are then used as blackboxes: when a more complicated functionality is required, it is reduced to a combination of known circuits. In the remainder of the paper, we study circuit design techniques which generalize the constructions of the previous section and we use these techniques to define new and efficient circuits.

Central to the techniques that we study here, and in general to the design of relative-phase circuits, is the fact that the Hadamard gate induces a bijection between states of the form $|y\oplus f(\mathbf{x})\rangle$ and $(-1)^{yf(\mathbf{x})}|y\rangle$. Specifically, it is an easy calculation to show that conjugation by Hadamard gates $\phi_H(\cdot)$ has the effect

$$|y\oplus f(\mathbf{x})\rangle\langle y| \xleftrightarrow{\phi_H(\cdot)} (-1)^{yf(\mathbf{x})}|y\rangle\langle y|$$

Stated as circuit equalities, this is the standard fact [21] that a target can be swapped for a "control", taken here as a $Z$ gate, and vice versa:



This simple fact can have perhaps surprising applications when oracles are allowed to be implemented up to (relative) phase. In the first application that we study — to circuits with ancillas — it provides an alternative way to uncompute temporary values: by turning them into relative phases. In particular, if an ancilla initially in the state $|a\rangle$ is used to store a temporary value $|a\oplus f(\mathbf{x})\rangle$, rather than uncompute this temporary value one can simply push $f(\mathbf{x})$ into the phase since $\phi_H(|a\oplus f(\mathbf{x})\rangle\langle a|) = (-1)^{af(\mathbf{x})}|a\rangle\langle a|$. We record this fact with the following statement, and illustrate its use by implementing a $\Lambda_k(X)$ gate up to relative phase using dirty ancillas:

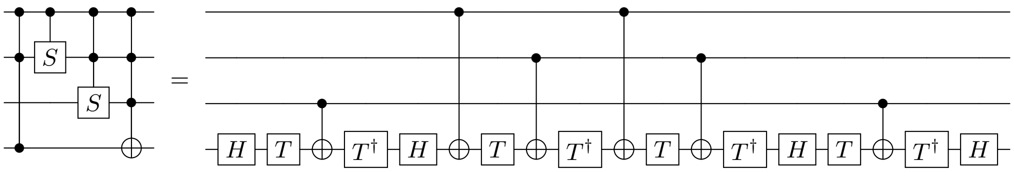
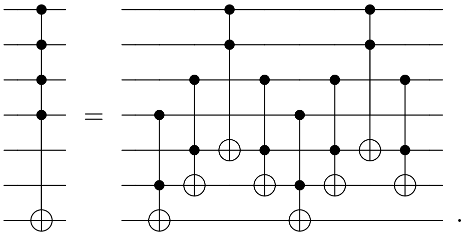


FIG. 3. A relative phase 4-qubit Toffoli gate [8].

*A temporary value stored in a dirty ancilla can always be left uncomputed, at the expense of a relative phase.*

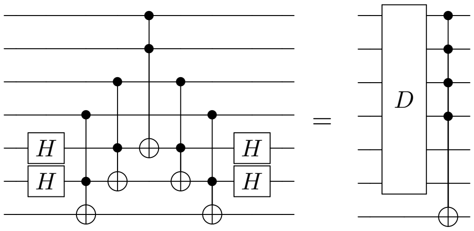
**Construction 1** (Relative phase $\Lambda_k(X)$)**.** Consider a 4-control Toffoli gate constructed using 2 dirty ancillas, as in [12, Lemma 7.2]:



The role of the final three Toffoli gates is to uncompute the temporary values in the ancillas. In their absence, the circuit would map an input state $|\mathbf{x}\rangle\,|a\rangle\,|b\rangle\,|y\rangle$ to the state

$$|\mathbf{x}\rangle\,|a \oplus (x_1x_2)\rangle\,|b \oplus (x_1x_2x_3)\rangle\,|y \oplus (x_1x_2x_3x_4)\rangle\,.$$

Rather than perform the final Toffoli gates, we can conjugate the ancillas with Hadamard gates to return them to their initial state at the expense of a relative phase. We then get



where the diagonal gate $D$ imparts a relative phase of $(-1)^{ax_1x_2+bx_1x_2x_3}$.

Care must be taken when trading (local) uncomputations for relative phases, as the ancillas must remain in the same state when matched with a (global) uncomputation later. Moreover, gates on the ancillas may *not* in general commute with such an implementation, while such gates do commute with an exact implementation.

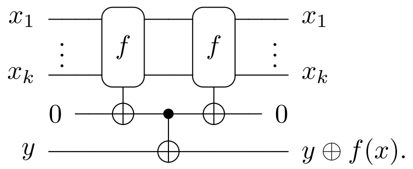
One may wonder whether a similar trick can be played with circuits using clean ancillas. In this case, conjugation by $H$ uncomputes a temporary value with *no* relative phase:

$$|f(\mathbf{x})\rangle\,\langle 0| \xleftrightarrow{\phi_H(\cdot)} |0\rangle\,\langle 0|$$

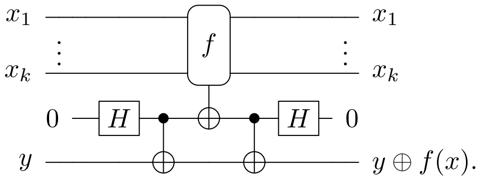
Internally, the clean ancilla is replaced with a dirty one and the clean value is effectively "stored" in the phase, to be retrieved later. Specifically, since $H\,|0\rangle = \frac{1}{\sqrt{2}}\sum_z |z\rangle$, after the initial Hadamard gate the ancilla is placed in a dirty state. Adding $f(\mathbf{x})$ to this ancilla results in the state $\frac{1}{\sqrt{2}}\sum_z |z \oplus f(\mathbf{x})\rangle$. Finally, since $\sum_z |z \oplus f(x)\rangle = \sum_{z'} |z'\rangle$ for any value of $f(x)$, the final Hadamard sends this state back to the clean state $|0\rangle$. We again summarize this fact and illustrate its use with a circuit construction.

*A temporary value stored in a dirty ancilla can always be left uncomputed without adding a relative phase by using a clean ancilla.*

**Construction 2** (Phase-based Bennett)**.** Recall Bennett's compute-copy-uncompute scheme to reclaim temporary ancillas:



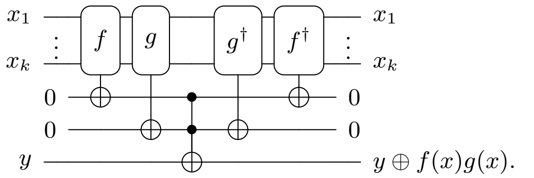
The compute-copy-uncompute construction can be reduced to a single compute by using the phase space to temporarily store the ancilla's (clean) value. Specifically, by applying a Hadamard gate to the ancilla, the clean $|0\rangle$ state is swapped into the phase space. This clean phase can later be swapped back into the state space with a Hadamard gate, uncomputing the intermediate state as below.



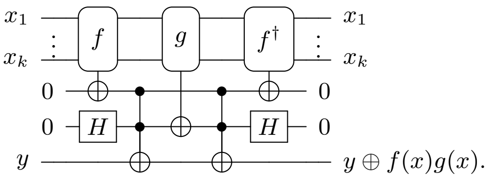
An additional $\Lambda_1(X)$ gate is needed, as after swapping the ancilla's initial (clean) state into the phase the ancilla exists in a *dirty* state. This dirty value is then added to the target register twice, canceling out, as in the constructions of [12].

To use the phase-based Bennett trick above for cleanup, the compute circuit needs to be designed to work with dirty ancillas — this can be prohibitive as implementations of oracles using clean ancillas are typically more time-efficient. However, in cases where the temporary values are expensive to compute it can sometimes be advantageous to adjust the inner computation to work with dirty ancillas, as the following construction shows.

**Construction 3** (Oracle multiplication)**.** Consider the circuit below multiplying two Boolean functions $f$ and $g$ using two clean ancillas:



We can eliminate one uncomputation of either $f$ or $g$ by swapping the phase and state space of the corresponding ancilla, at the expense of one extra Toffoli gate to deal with the now dirty ancilla.



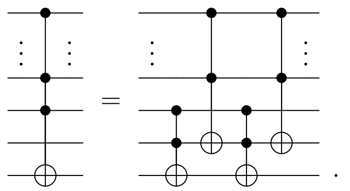
(1)

Note that here, the two Toffoli gates can be replaced with appropriate $\Lambda_2(iZ)$ and Hadamard gates, requiring 8 $T$ gates rather than the 14 that would be required if Toffoli gates were used.

**Proposition 4.** *Let $f, g : \mathbb{Z}_2^k \to \mathbb{Z}_2$ be Boolean functions and suppose the oracles $U_f$ and $U_g$ can be implemented with $T$-count $\tau(U_f)$ and $\tau(U_g)$, respectively. With two additional clean ancillas, the oracle $U_{f\cdot g}$ can be implemented by a circuit of $T$-count $2\tau(U_f) + \tau(U_g) + 8$.*

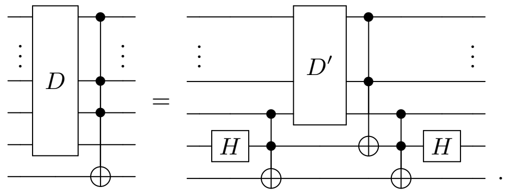
We now apply these observations to construct implementations of multiply-controlled Toffoli gates with a single dirty ancilla, both up to relative phase and implemented exactly, using fewer $T$ gates than previously known.

**Construction 5** ($\Lambda_k(X)$ with a single dirty ancilla)**.** Recall [12, Lemma 7.3] that a $k$-controlled Toffoli gate can be decomposed as follows, using a single dirty ancilla:



The construction can be applied recursively using the $k$th bit as a dirty ancilla. However, this results in an exponential gate count, since the dirty ancilla needs to be cleaned in each recursive instantiation.

We can recover a linear gate count with a simple recursive construction by trading the temporary value held in the dirty ancilla for a phase in each step. We then obtain the following equality, where $D$ and $D'$ are some particular diagonal gates:

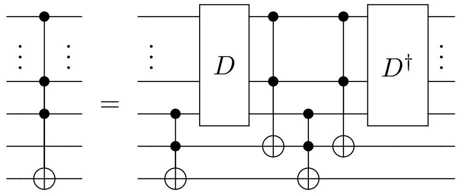


(2)

The precise form of $D$ is given in Appendix B.

The above construction reduces the $T$-count for a $k$-controlled Toffoli gate with a single dirty ancilla to $8(k-2)+4$, using $\Lambda_2(iX)$ gates to implement Toffolis, at the expense of a relative phase on the controls and ancilla. By comparison, the best-known $T$-count for $\Lambda_k(X)$ with a single dirty ancilla is $16(k-1)$ using [12, Lemma 7.3] together with Maslov's circuit [8] for the inner $\Lambda_{k/2}(X)$ gates.

A $k$-controlled Toffoli gate can also be implemented on the nose by performing a final overall cleanup of the dirty ancilla, as below:



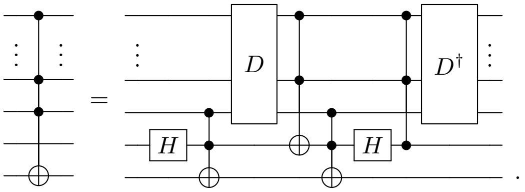
(3)

For $k \geq 4$, this gives a $\Lambda_k(X)$ gate with a single dirty ancilla and $T$-count $16(k-2)$. This reduces the $T$-count of the best-known construction by 16.

**Proposition 6.** *Let $k \in \mathbb{Z}^{\geq 4}$. With a single dirty ancilla, the $\Lambda_k(X)$ gate can be implemented by a circuit of $T$-count $16(k-2)$.*

**Proposition 7.** *Let $k \in \mathbb{Z}^{\geq 2}$. With a single dirty ancilla, the $\Lambda_k(X)$ gate can be implemented up to a phase in the controls and the ancilla by a circuit of $T$-count $8(k-2)+4$.*

*Remark* 8. In general it is preferable to use an alternate form of the $\Lambda_k(X)$ where the ancilla is cleaned by pushing the temporary value into the phase, followed by a phase cleanup. In this case, if the $\Lambda_k(X)$ is later uncomputed the phase cleanup can be cancelled by automated means. The circuit is shown below:



## IV. ANCILLA-FREE CIRCUITS

We now turn our attention to even more space-efficient constructions and in particular to circuits which do not use ancillas. We cover design techniques that can be used to implement generalized permutations. We then use these techniques to give $T$-count efficient relative phase implementations of multiply-controlled Toffoli gates. We also identify Boolean functions for which the $T$-count of the best-known construction [10] can be matched or beaten without the use of ancillas and measurement (but at the cost of a relative phase).

Recall that $\mathbb{D}[\omega]$ is the ring of dyadic fractions extended by $\omega = e^{i\pi/4}$. It was shown in [5] that, for $n \geq 4$, every $n$-qubit unitary with determinant 1 and entries in $\mathbb{D}[\omega]$ can be exactly represented by an ancilla-free Clifford+$T$ circuit. Since the determinant of a permutation is either 1 or $-1$, depending on whether the permutation is even or odd, it follows that exactly the even permutations can be represented by an ancilla-free Clifford+$T$ circuit. If we allow relative phase implementations, however, any permutation can be implemented.

To find generalized permutations with efficient ancilla-free implementations, it can be helpful to *push all computation to the phase space.* In particular, given a relative-phase implementation (suppressing the constant $|\mathbf{x}\rangle$ register)

$$\widetilde{U_f} = e^{ig(\mathbf{x})} |y \oplus f(\mathbf{x})\rangle \langle y|,$$

conjugating by $H$ pushes all computation to the phase:

$$H\widetilde{U_f}H = e^{ig(\mathbf{x})}(-1)^{yf(\mathbf{x})} |y\rangle \langle y|.$$

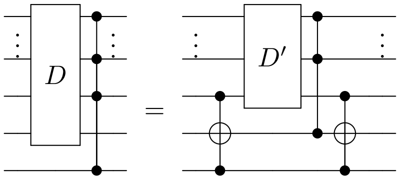
By instead synthesizing the *phase oracle*, we can more easily find relative phases $e^{ig(\mathbf{x})}$ which reduce the overall $T$-count, as the next constructions show.

**Construction 9** (Relative phase $\Lambda_k(Z)$)**.** The circuit from Construction 5 can be equivalently derived by synthesizing the phase-space version of $\Lambda_k(Z)$ up to relative phase. In particular, to perform a multiply-controlled $Z$ gate up to relative phase, we want to compute some phase

$$(-1)^{y \cdot x_1 \cdots x_k} e^{ig(\mathbf{x},a)}$$

where $y$ is the target and $a$ is an ancillary bit.

We can build a circuit to do so by first multiplying (in $\mathbb{F}_2$) $y$ by $x_k$ and adding this to an ancilla $a$, then applying a phase of $(-1)^{x_1 \cdots x_{k-1}(a+yx_k)}$. In particular we have the equality below, for some diagonal gate $D$:



Conjugating by $H$ on the target gives the $\Lambda_k(X)$ circuit from Construction 5, up to swapping controls and targets by commuting the Hadamards.

**Construction 10** (Selinger's $\Lambda_2(iX)$)**.** The doubly-controlled $iX$ gate in Figure 1 computes the following transformation on computational basis states:

$$|x_1\rangle |x_2\rangle |y\rangle \mapsto i^{x_1x_2} |x_1\rangle |x_2\rangle |y \oplus (x_1x_2)\rangle.$$

To see how $\Lambda_2(iX)$ arises naturally as an efficient relative phase implementation of the Toffoli gate over Clifford+$T$, it is helpful to consider the doubly-controlled $Z$ gate:

$$|x_1\rangle |x_2\rangle |y\rangle \mapsto (-1)^{x_1x_2y} |x_1\rangle |x_2\rangle |y\rangle.$$

Since $(-1)^{x_1x_2y} = \omega^{4x_1x_2y}$, we can use the equality [6]

$$\begin{aligned} 4x_1x_2y = x_1 + x_2 + y - (x_1 \oplus x_2) - (x_1 \oplus y) \\ - (x_2 \oplus y) + (x_1 \oplus x_2 \oplus y) \end{aligned}$$

to implement the doubly-controlled $Z$ gate over Clifford+$T$ by computing each of the terms in the above sum (using $\Lambda_1(X)$ gates) and applying a $T$ or a $T^\dagger$ gate. If we only apply the 4 rotations which depend on $y$, and noting that $2x_1x_2 = x_1 + x_2 - (x_1 \oplus x_2)$, the resulting phase term is

$$4x_1x_2y - 2x_1x_2 = y - (y \oplus x_1) - (y \oplus x_2) + (y \oplus x_1 \oplus x_2)$$

Conjugating by $H$ on the target then sends the output $(-1)^{x_1x_2y} i^{-x_1x_2} |x_1\rangle |x_2\rangle |y\rangle$ to $i^{-x_1x_2} |x_1\rangle |x_2\rangle |y \oplus (x_1x_2)\rangle$, implementing the desired transformation up to a phase of $i^{-x_1x_2}$.

In general, for any classical function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$, an oracle for $f$ can be implemented up to relative phase by taking the Fourier transform [22] of $yf(x)$ and then dropping all terms that do not involve $y$. Explicitly, if

$$(-1)^{yf(x)} = \omega^{g(x)+yh(x)},$$

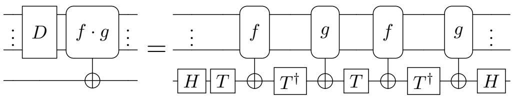
then it suffices to implement the phase oracle $\omega^{yh(x)}$. We summarize this in the following statement:

> *A relative-phase implementation of $U_f$ can be obtained by taking the Fourier transform of $yf(\mathbf{x})$ and truncating all terms which do not depend on $y$.*

Using this idea, we can devise a method to multiple two Boolean functions $f, g$ up to relative phase without ancillas, given oracles for $f$ and $g$. In particular, since

$$(-1)^{y \cdot f(x)g(x)} i^{-f(x)g(x)} = \omega^{y - y \oplus f(x) - y \oplus g(x) + y \oplus f(x) \oplus g(x)},$$

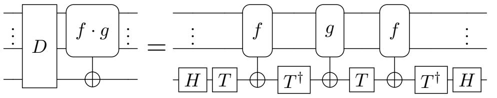
we can alternately add $f(x)$ and $g(x)$ into the target $y$ and apply the appropriate $T$ or $T^\dagger$ gate, as below:



(4)

The above *matched* multiplication construction generalizes the multiply-controlled $iX$ implementation of Giles and Selinger [5].

We arrive at a slightly different form of ancilla-free oracle multiplication by noting that the final computation of $U_g$ serves only to uncompute the temporary value $g(x)$. As noted in Section III, since the target is conjugated with Hadamard gates, this temporary value can instead be swapped into the phase. The result is the *unmatched* oracle multiplication circuit below, which generalizes Maslov's relative phase 4-qubit Toffoli [8]. Note that unlike matched multiplication, unmatched multiplication results in a target-dependent phase.



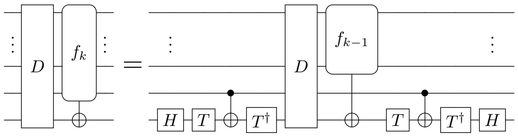
(5)

**Proposition 11.** *Let $f, g : \mathbb{Z}_2^k \to \mathbb{Z}_2$ be Boolean functions and suppose the oracles $U_f$ and $U_g$ can be implemented with $T$-count $\tau(U_f)$ and $\tau(U_g)$, respectively. With no additional ancillas, the oracle $U_{f \cdot g}$ can be implemented up to a phase in the controls by a circuit of $T$-count $2\tau(U_f) + 2\tau(U_g) + 4$.*

**Proposition 12.** *Let $f, g : \mathbb{Z}_2^k \to \mathbb{Z}_2$ be Boolean functions and suppose the oracles $U_f$ and $U_g$ can be implemented with $T$-count $\tau(U_f)$ and $\tau(U_g)$, respectively. With no additional ancillas, the oracle $U_{f \cdot g}$ can be implemented up to a phase in the controls and the target by a circuit of $T$-count $2\tau(U_f) + \tau(U_g) + 4$.*

The multiplication constructions above can be instantiated in various ways to design relative phase circuits without ancillas. We now cover some of these applications.

**Construction 13** (Efficient high-degree oracles)**.** By recursively instantiating $U_g$ in the unmatched multiplication, we can quickly (in the $T$-count) grow the degree of a Boolean function by setting $f(x) = x$, multiplying in one control at each iteration.



(6)

The function $f_k$ is defined by the recurrence

$$\begin{aligned} f_0(x) &= 0 \\ f_1(x) &= x_1 \\ f_k(x) &= x_k \cdot f_{k-1}(x) + f_{k-2}(x) \end{aligned}$$

The contribution of $f_{k-2}$ is due to the relative phase of $(-1)^{y \cdot f_{k-1}}$ from the unmatched multiplication, which eventually gets swapped *back* into the state. For instance, for $k = 4$ we have

$$f_4(x_1, x_2, x_3, x_4) = x_1x_2x_3x_4 \oplus x_1x_4 \oplus x_3x_4.$$

Different recurrences and initial conditions can be obtained by tuning the construction with additional Clifford gates, or by switching to matched multiplication. In particular, the relative phase 4 qubit Toffoli in Figure 3 is obtained by using matched multiplication for $f_2$ at no additional $T$-cost. The result is the recurrence

$$\begin{aligned} f_0(x) &= 0 & f_1(x) &= x_1 \\ f_2(x) &= x_1x_2 & f_3(x) &= x_1x_2x_3 \\ f_k(x) &= x_k \cdot f_{k-1} + f_{k-2} \end{aligned}$$

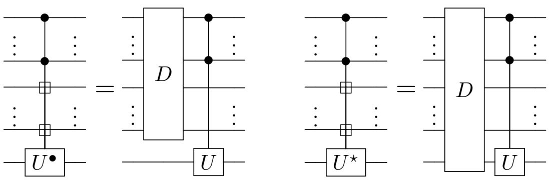
**Proposition 14.** *There exists a maximal degree Boolean function $f : \mathbb{Z}_2^k \to \mathbb{Z}_2$ such that, without ancillas, the oracle $U_f$ can be implemented up to a phase in the controls and the target by a ciruit of $T$-count $4(k-1)$.*

**Proposition 15.** *There exists a maximal degree Boolean function $f : \mathbb{Z}_2^k \to \mathbb{Z}_2$ such that, with a single dirty ancilla, the oracle $U_f$ can be implemented by a ciruit of $T$-count $8(k-1)$.*

*Remark* 16. The construction in Construction 13 is notable in that matches or outperforms the best-known [10] $T$-count for any degree $k$ function, *without ancillas, measurement, or classical control* but at the expense of a relative phase. Specifically, the above construction uses $4(\deg(f_k)-1)$ $T$ gates, where $\deg(f_k)-1 \leq c_\wedge(f_k)$, the multiplicative complexity of $f_k$.

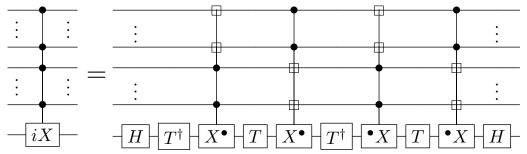
While functions derivable with this construction are not likely to be of practical use for circuit designers, they may be useful in automated circuit synthesis such as LUT-based logic synthesis [11], where arbitrary Boolean functions on a small number of bits are used to synthesize larger oracles. For instance, $f_4$ and $f_5$ — corresponding to the spectral classes #0888 and #a8808000 [23], respectively — reduce the best-known, space-minimal constructions from $T$-count 77 and 490 to 12 and 16 up to phase, or 24 and 32 exactly [23]. We leave it as an area of future work to identify more distinct spectral classes efficiently implementable using variations of this construction.

We end the section by giving novel relative phase implementations of the $k$-control Toffoli gate, our best construction of which halves the $T$-count of the best-known ancilla-free circuit. To simplify our presentation, we introduce shorthand for two types of relative phase gates: $U^\bullet$ and $U^\star$, corresponding to whether the relative phase is on the controls and ancillas, or controls, ancillas, and target, respectively. We use boxes on dirty ancillas to denote relative phases. Hence,



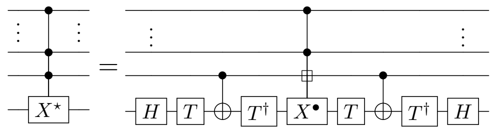
where the gates $D$ are some unspecified diagonal gates. We denote the inverse of $U^\bullet$ or $U^\star$ by ${}^\bullet U$ or ${}^\star U$, respectively.

**Construction 17** (Ancilla-free Toffoli gates)**.** We first note that we can use the relative phase Toffoli of Construction 5 together with matched multiplication, which cancels each of the relative phases, to get an improved (in the $T$-count) implementation of the $\Lambda_k(iX)$ gate ($T$-count $16(k-3)+4$ when $k \geq 4$):



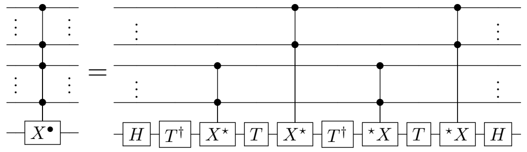
(7)

Next we leverage the un-matched multiplication, placing all but one control on the un-matched Toffoli gate and using the single dirty ancilla relative phase Toffoli from Construction 5 to perform it up to phase. The result is an ancilla-free $k$-controlled Toffoli gate using $8(k-2)$ $T$ gates, roughly half that of the best-known ancilla-free $k$-controlled Toffoli gate, at the expense of a target-dependent phase:



(8)

Our final ancilla-free construction, below, uses the previous circuit to perform a $k$-controlled Toffoli up to a phase *only on the controls*. The construction uses matched multiplication to eliminate the target-dependent phases produced by the intermediate Toffoli gates. The result is an additional 16 $T$ gates of savings compared to the $k$-controlled $iX$ gate above ($T$-count $16(k-4)+4$ when $k \geq 5$):



(9)

**Proposition 18.** *Let $k \in \mathbb{Z}^{\geq 5}$. Without ancillas, the $\Lambda_k(X)$ gate can be implemented up to a phase in the controls by a circuit of $T$-count $16(k-4)+4$.*

**Proposition 19.** *Let $k \in \mathbb{Z}^{\geq 3}$. Without ancillas, the $\Lambda_k(X)$ gate can be implemented up to a phase in the controls and the target by a circuit of $T$-count $8(k-2)$.*

*Remark* 20. Combining the $X^\star$ construction with the Gidney logical-AND [9] gives a method of further reducing the $T$-count of the multiply-controlled Toffoli gate (up to phase) when *some* ancillas are available. In particular, by using Gidney's logical-AND to initialize and terminate $m$ temporary products with $m$ clean ancillas and $4m$ $T$ gates, this gives a $T$-count of $4m + 8(k - m - 2)$ for a $k$-controlled $X^\star$ with $m$ clean ancillas, where $m \leq k - 1$.

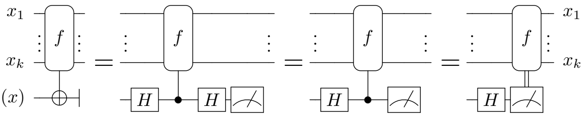
## V. MEASUREMENT-ASSISTED UNCOMPUTATION

The last technique that we study generalizes the constructions of Gidney and Jones for terminating a temporary product [7, 9]. Recall that by termination we mean the dual of initialization, which is distinguished from the (unitary) process of uncomputation.

To terminate an ancilla in the temporary state $|f(x)\rangle$, one typically uncomputes $f$. Instead, we can swap the state into the phase space by applying a Hadamard gate:

$$H\,|f(x)\rangle = \frac{1}{\sqrt{2}} \sum_{y \in \mathbb{Z}_2} (-1)^{yf(x)}\,|y\rangle\,.$$

Measuring the ancillary qubit then leaves a phase of 1 or a phase of $(-1)^{f(x)}$. In the latter case, this phase can be corrected by a classically-controlled $(-1)^{f(x)}$ phase oracle. This is reflected in the following sequence of circuit equalities:



Note that the second equality follows from the fact that single qubit gates preceding a discarded measurement can be dropped [24]. We once again summarize this fact below.

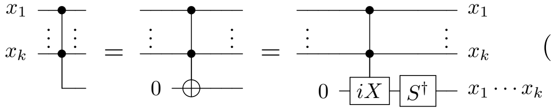
> *A temporary value $|f(x)\rangle$ can be terminated by measuring in the $X$-basis and applying a classically-controlled $(-1)^{f(x)}$ correction.*

In the case of the 2-qubit Toffoli gate, the classically-controlled correction of $(-1)^{x_1x_2}$ can be implemented using only Clifford gates. In contrast, in the general case, correcting the phase $(-1)^{f(x)}$ might require $T$ gates. This can nonetheless still reduce the $T$-count, since uncomputing $f(x)$ without measurement requires computing the phase

$$|x\rangle\,|y\rangle \mapsto (-1)^{yf(x)}\,|x\rangle\,|y\rangle$$

conrresponding to an extra quantum control.

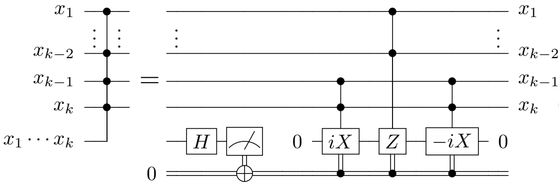
**Construction 21** (Terminating $\Lambda_k(X)$)**.** Consider the logical product of $k$ bits $|(x_1 \cdots x_k)\rangle$, which can be initialized with a clean ancilla by applying a multiply-controlled $iX$ gate and an $S^\dagger$:



(12)

Using the $\Lambda_k(iX)$ gate from Section IV, the logical product above uses $16(k - 3) + 4$ $T$ gates.

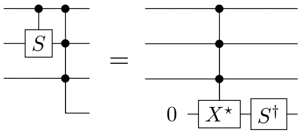
After this temporary product is no longer needed, we can terminate it by measuring in the $X$ basis, resulting in the state $|(x_1 \cdots x_k)\rangle$ or $(-1)^{x_1 \cdots x_k}\,|(x_1 \cdots x_k)\rangle$. Correcting this phase requires a $\Lambda_{k-1}(Z)$ gate. Using the methods from Section III, this gate can be implemented with a single ancilla in $T$-count $16(k - 3)$. With non-destructive measurements, this can be further reduced to $16(k-4)+4$ by re-using the measured qubit as a clean ancilla to initialize a temporary product of two bits:



(13)

In the case where the oracle is implemented up to phase, the $T$-count of the final phase correction can sometimes be reduced further by applying the phase correction *itself up to phase.* The following construction gives a measurement-assisted termination circuit for a temporary logical 3-AND based on Maslov's Toffoli 4.

**Construction 22** (Terminating Maslov's $\Lambda_3(X^\star)$)**.** The ternary logical AND $f(x) = x_1x_2x_3$ can be initialized with a clean ancilla using Maslov's 4-qubit Toffoli gate (see Figure 3) up to a phase of $i^{x_1x_2}$. In particular,



where the 3-control $X^\star$ gate is the right hand side of Figure 3. In this case, measuring the product in the $X$ basis gives either a phase of $i^{x_1x_2}$ if the result is 0, or a phase of $i^{x_1x_2}(-1)^{x_1x_2x_3}$ if the measurement

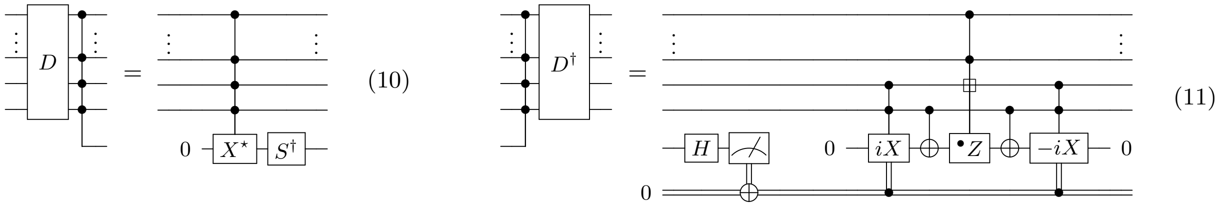
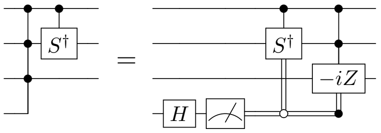


FIG. 4. Initializing and terminating a temporary logical AND of $k$ bits. Combined $T$-count $16(k-2)-10 \pm 2$.

result is 1. In the former case, the phase of $i^{x_1x_2}$ can be corrected with a controlled $S^\dagger$ gate in 3 $T$ gates, while the latter phase of $i^{x_1x_2}(-1)^{x_1x_2x_3}$ can be corrected with a $\Lambda_2(-iZ)$ gate using 4 $T$ gates. The corresponding circuit is shown below.



(14)

We close by giving an efficient logical $k$-AND using our $\Lambda_k(X^\star)$ to initialize the product, and measurement to terminate it (Figure 4). The termination construction shaves roughly 16 $T$ gates off the cost to compute the product.

**Proposition 23.** *A logical AND of $k$ bits can be initialized up to relative phase with $8(k-2)$ $T$ gates and terminated with either $8(k-4)$ or $8(k-4)+4$ $T$ gates.*

As a corollary we additionally obtain a Jones-style circuit for the $\Lambda_k(X)$ gate which uses a single clean ancilla, measurements and classical control.

**Proposition 24.** *Let $k \in \mathbb{Z}^{\geq 4}$. With a single dirty ancilla and measurements, the $\Lambda_k(X)$ gate can be implemented by a circuit of $T$-count $16(k-3)+4$.*

## VI. CONCLUSION

In this paper we described a number of techniques which use the phase/state duality in order to efficiently implement quantum oracles, most notably in the low-space regime. These techniques generalize, among others, the relative phase Toffolis of Maslov and Selinger, as well as the measurement-assisted uncomputations of Gidney and Jones. Using these techniques, we developed several new circuit constructions. These constructions, which are summarized in Table I, include circuits for Toffoli gates, multiplying Boolean functions, and high-degree classical gates.

## ACKNOWLEDGMENTS

The authors would like to thank Craig Gidney for helpful and enlightening comments on an earlier version of this paper.

[1] L. K. Grover, in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96 (ACM, New York, NY, USA, 1996) pp. 212–219.

[2] P. W. Shor, SIAM J. Comput. **26**, 1484 (1997).

[3] C. H. Bennett, SIAM J. Comput. **18**, 766–776 (1989).

[4] H. Buhrman, R. Cleve, M. Laurent, N. Linden, A. Schrijver, and F. Unger, in *2006 47th Annual IEEE Conference on Foundations of Computer Science* (IEEE Computer Society, Los Alamitos, CA, USA, 2006) pp. 411–419.

[5] B. Giles and P. Selinger, Physical Review A **87**, 032332 (2013), arXiv:1212.0506.

[6] P. Selinger, Physical Review A **87**, 042302 (2013), arXiv:1210.0974.

[7] C. Jones, Physical Review A **87**, 022328 (2013), arXiv:1212.5069.

[8] D. Maslov, Physical Review A **93**, 022311 (2016), arXiv:1508.03273.

[9] C. Gidney, Quantum **2**, 74 (2018).

[10] G. Meuli, M. Soeken, E. Campbell, M. Roetteler, and G. de Micheli, in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2019) pp. 1–8, arXiv:1908.01609.

[11] M. Soeken, M. Roetteler, N. Wiebe, and G. D. Micheli, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **38**, 1675 (2019).

[12] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Physical Review A **52**, 3457 (1995), arXiv:quant-ph/9503016.

[13] D. P. DiVincenzo and J. Smolin, in *Proceedings Workshop on Physics and Computation*, PhysComp '94 (1994) pp. 14–23, cond-mat:9409111.

[14] D. Gosset, V. Kliuchnikov, M. Mosca, and V. Russo, Quantum Info. Comput. **14**, 1261–1276 (2014).

[15] M. Amy, D. Maslov, and M. Mosca, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **33**, 1476 (2014), arXiv:1303.2042.

[16] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov, npj Quantum Information **4**, 23 (2018), arXiv:1710.07345.

[17] The minimum number of AND gates required to implement $f$ over {AND, XOR, NOT}.

[18] D. W. Berry, C. Gidney, M. Motta, J. R. McClean, and R. Babbush, Quantum **3**, 208 (2019).

[19] M. Soeken and M. Roetteler, in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* (2020) pp. 366–371.

[20] C. Gidney, Spooky pebble games and irreversible uncomputation, `https://algassert.com/post/1905` (2019), accessed: 2021-06-01.

[21] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge Series on Information and the Natural Sciences (Cambridge University Press, 2000).

[22] M. Amy, P. Azimzadeh, and M. Mosca, Quantum Science and Technology **4**, 015002 (2018), arXiv:1712.01859.

[23] G. Meuli, M. Soeken, M. Roetteler, and G. De Micheli, in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)* (2020) pp. 1–5.

[24] A. Kissinger and S. Uijlen, in *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)* (2017) pp. 1–12.

## Appendix A: Implications of relative phases

As most of our constructions have some form of relative phase associated with them, care must be taken when using them. Here we show that any permutation or reversible circuit which is later uncomputed can be replaced with a relative phase implementation, provided the interior computation doesn't modify the basis state of any qubits on which there is a relative phase.

**Proposition 25.** *Let $U_f$ be an oracle for some Boolean function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ and let $U$ be some unitary transformation on $m > n$ qubits. Without loss of generality, if $U$ is constant up to phase on the first $n$ qubits, then*

$$(U_f^\dagger \otimes I)U(U_f \otimes I) = (\widetilde{U_f}^\dagger \otimes I)U(\widetilde{U_f} \otimes I)$$

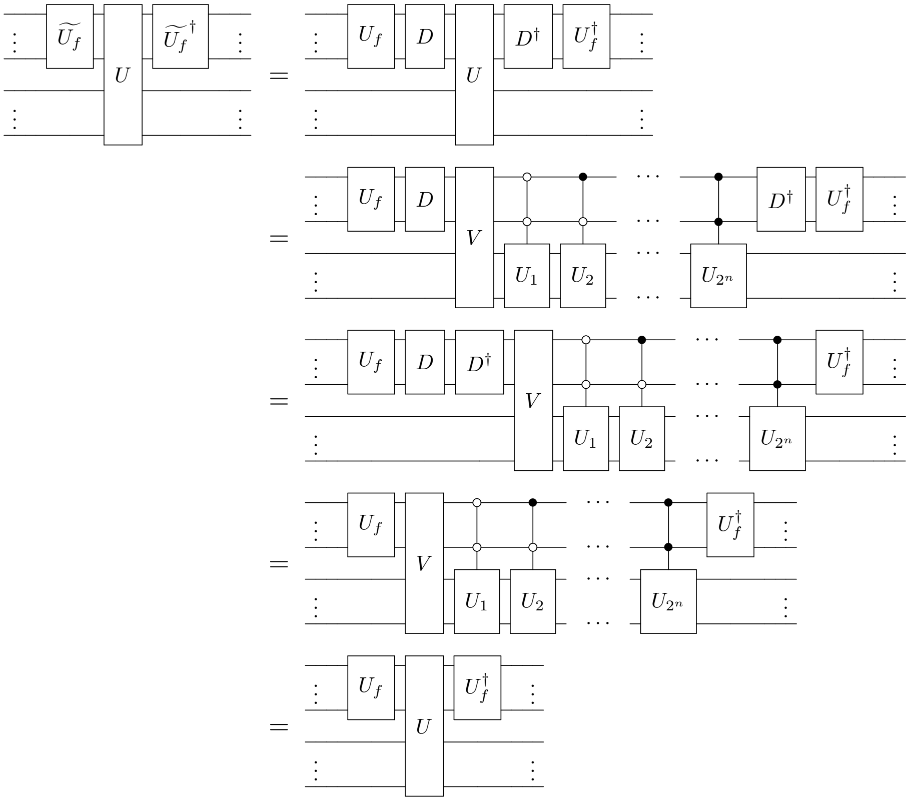
*for any relative phase implementation $\widetilde{U_f}$ of $f$.*

*Proof.* First observe that if $U$ is constant up to phase in the first $n$ qubits, then $U$ is equivalent to some circuit where the first $n$ qubits are *only used as controls*. In particular, $U$ can be written as a product of a diagonal unitary $V$ and $2^n$ matrices controlled on the $2^n$ basis states of the first $n$ qubits:

$$U = \prod_{i=1}^{2^n} \Lambda_n^i(U_i)$$

where we use $\Lambda_n^i(U_i)$ to denote the application of $U_i$ controlled on the first $n$ qubits having basis state equal to the binary expansion of $i$.

Now recalling that a generalized permutation $\widetilde{U_f}$ may be factored equally as $DU_f = \widetilde{U_f} = U_f D'$ for some diagonal matrices $D, D'$, we can proceed by calculation.



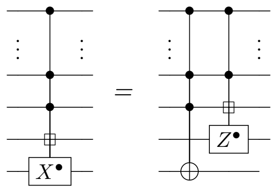
□

A simple rule-of-thumb for when the above proposition can be applied is whenever $U_f$ and $U_f^\dagger$ are used as a compute/uncompute pair, and $U_f$ does not impart a relative phase on a *dirty* ancilla. In such a case, $U$ is necessarily globally constant on the qubits used in $U_f$ in order for $U_f^\dagger$ to correctly uncompute $U_f$.

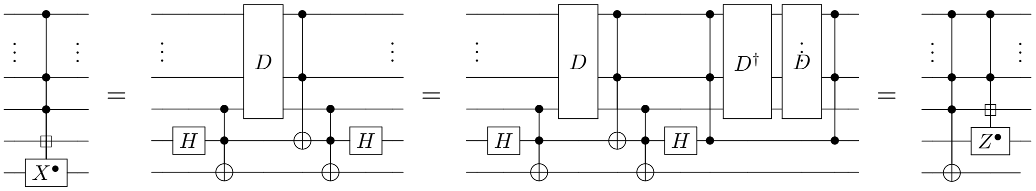
## Appendix B: Correctness of the logical k-AND termination

In this appendix we establish the precise form of the relative phase $D$ in the circuit constructions in Figure 4, and prove correctness of the termination circuit. To do so, we first establish the form of the relative phase in the single dirty ancilla $X^\bullet$.

**Proposition 26.**



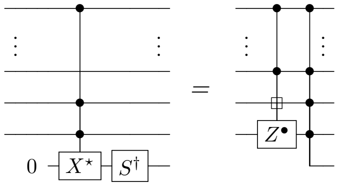
*Proof.*



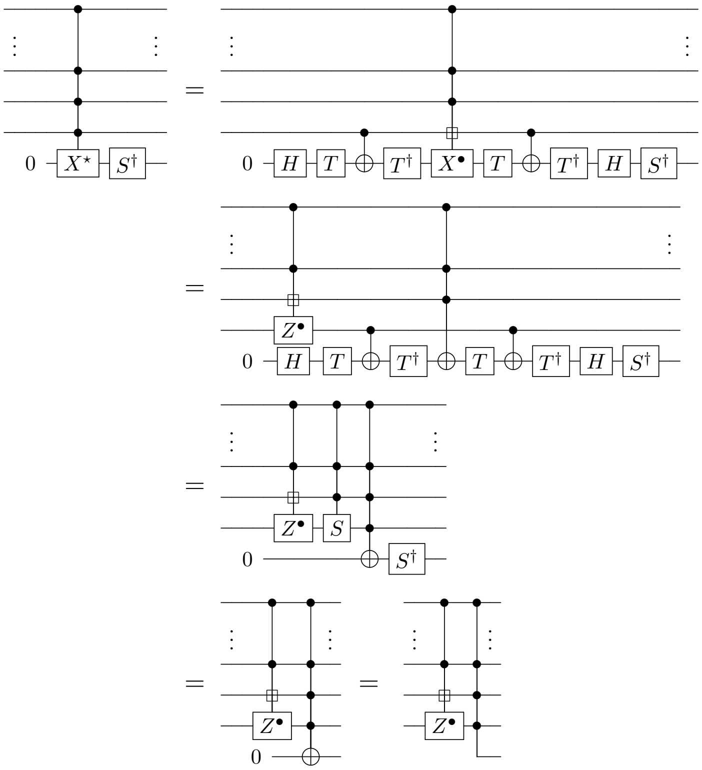
□

We now establish the correctness of the circuit constructions in Figure 4, and give explicit forms for the relative phases.

**Proposition 27.**

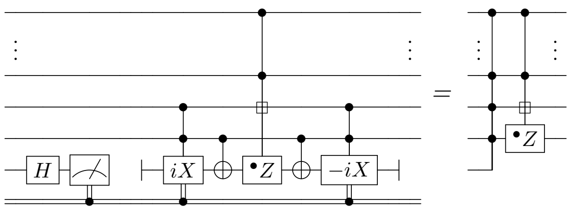


*Proof.*

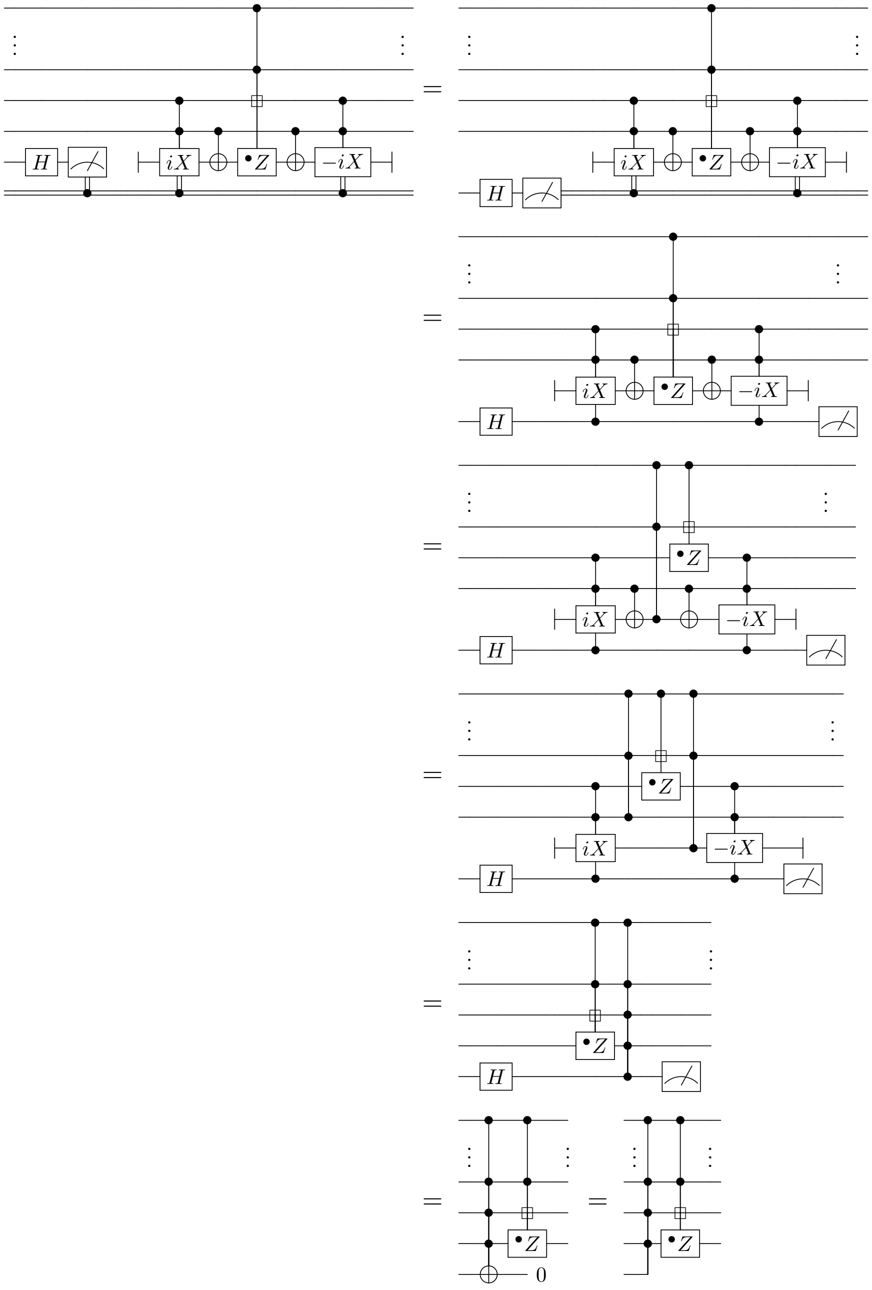


□

**Proposition 28.**



*Proof.*



□