

ABSTRACT

Title of dissertation: Quantum Compiling Methods for
Fault-Tolerant Gate Sets of
Dimension Greater than Two

Andrew Glaudell
Doctor of Philosophy, 2019

Dissertation directed by: Dr. Jacob M. Taylor
Department of Physics

Fault-tolerant gate sets whose generators belong to the Clifford hierarchy form the basis of many protocols for scalable quantum computing architectures. At the beginning of the decade, number-theoretic techniques were employed to analyze circuits over these gate sets on single qubits, providing the basis for a number of state-of-the-art quantum compiling algorithms. In this dissertation, I further this program by employing number-theoretic techniques for higher-dimensional gate sets on both qudit and multi-qubit circuits.

First, I introduce canonical forms for single qudit Clifford+ T circuits and prove that every single-qudit Clifford+ T operator admits a unique such canonical form. I show that these canonical forms are T -optimal and describe an algorithm which takes as input a Clifford+ T circuit and outputs the canonical form for that operator. The algorithm runs in time linear in the number of gates of the circuit. Our results provide a higher-dimensional generalization of prior work by Matsumoto and

Amano who introduced similar canonical forms for single-qubit Clifford+ T circuits. Finally, we show that a similar extension of these normal forms to higher dimensions exists, but do not establish uniqueness.

Moving to multi-qubit circuits, I provide number-theoretic characterizations for certain restricted Clifford+ T circuits by considering unitary matrices over subrings of $Z[1/\sqrt{2}, i]$. We focus on the subrings $Z[1/2]$, $Z[1/\sqrt{2}]$, $Z[1/\sqrt{-2}]$, and $Z[1/2, i]$, and we prove that unitary matrices with entries in these rings correspond to circuits over well-known universal gate sets. In each case, the desired gate set is obtained by extending the set of classical reversible gates $\{X, CX, CCX\}$ with an analogue of the Hadamard gate and an optional phase gate.

I then establish the existence and uniqueness of a normal form for one of these gate sets, the two-qubit gate set of Clifford+Controlled Phase gate CS . This normal form is optimal in the number of CS gates, making it the first normal form that is non-Clifford optimal for a fault tolerant universal multi-qubit gate set. We provide a synthesis algorithm that runs in a time linear in the gate count and outputs the equivalent normal form. In proving the existence and uniqueness of the normal form, we likewise establish the generators and relations for the two-qubit Clifford+ CS group. Finally, we demonstrate that a lower bound of $5 \log_2(1/\varepsilon) + O(1)$ CS gates are required to ε -approximate any 4×4 unitary matrix.

Lastly, using the characterization of circuits over the Clifford+ CS gate set and the existence of an optimal normal form, I provide an ancilla-free inexact synthesis algorithm for two-qubit unitaries using the Clifford+SC gate set for Pauli-rotations.

These operators require $6 \log_2(1/\varepsilon) + O(1)$ CS gates to synthesize in the typical case and $8 \log_2(1/\varepsilon) + O(1)$ in the worst case.

Quantum Compiling Methods for Fault-Tolerant Gate Sets of
Dimension Greater than Two

by

Andrew Glaudell

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Dr. Jacob M. Taylor, Advisor
Professor Andrew Childs, Chair
Dr. Carl Miller
Professor Norbert Linke
Professor Lawrence Washington

© Copyright by
Andrew Glaudell
2019

Acknowledgments

The work contained herein is a testament to the fantastic support I was fortunate to receive before and during graduate school. This space is not large enough to thank the countless individuals who have made an impact on my life, but rest assured that those people not mentioned here remain in my thoughts. Among these, I'd like to give a shout-out to the plethora of artists and musicians, especially Mike Scalzi and The Lord Weird Slough Feg, for providing the best soundtrack to grad school anyone could want.

First, I would like to thank my parents, Ken and Linda Glaudell, for providing such a sturdy bedrock for me and going above and beyond to support my interests. You two made many sacrifices to give your kids their best shot in life. My sister Allie has likewise been there with a laugh and a gentle smile any time that I needed one.

My interest in learning owes a great debt to a number of former educators. I'd like to single-out Mr. Kroening, Mme. Kosmider, Mr. Rose, and Ms. Frank for being phenomenal teachers for whom I will forever be grateful. From my undergraduate studies, I would not be where I am now without the valuable guidance of my former advisors Prof. Wendy Crone and Prof. Maxim Vavilov. I'd also like to give special thanks to Dave Grierson for being such a funny and insightful resource in lab.

My current advisor, Prof. Jacob Taylor, has been such a great asset in pushing

my scientific capabilities, writing efforts, and professional development. I am also indebted to the Taylor group post-docs Michael Gullans, Vanita Srinavasa, Shelby Kimmel, Jianxin Chen, Justyna Zwolak, Xingyao Wu, and Dan Carney and graduate students Dvir Kafri, Xunnong Xu, Steve Ragole, Chiao-Hsuan Wang, Shangjie Guo, Minh Tran, Brittany Richman, and Jonathon Kunjummen. I would especially like to stress how much my colleague and friend Neil Julien Ross, formerly a post-doc at UMD, has shaped my research interest.

To all my friends from previous stops along my life including Cameron Gonring, Jon Gill, Eric Bowron, Adam Reinicke, the entire “Manventure” crew, and all my former roommates: thank you for being the best group of people imaginable with whom to laugh along. To the friends I met in grad school including Joe Hart, Chris Eckberg, Clayton Crocker, Jaime David Campos-Wong, the Seven Seas conglomerate, the UMD Physics intramural squads, the Physics 131 teaching crew and friends, and my office mates: you have made what could’ve been one of the most trying times in my life a great deal more fun than it had any right to be.

Finally, to Mary (and Miska)- you have helped me grow so much in our time together and pushed me to be both a better graduate student and person. Thank you for making every single day worth it.

Table of Contents

Acknowledgements	ii
Table of Contents	iv
List of Tables	vii
List of Figures	viii
List of Abbreviations	x
1 Introduction	1
1.1 Quantum Computing and Quantum Compiling	1
1.2 Applications of Quantum Computing	3
1.2.1 Hidden Subgroup Problem	3
1.2.2 Unsorted Search of a Database	4
1.2.3 Hamiltonian Simulation	5
1.2.4 Linear Equation Solving	5
1.3 Approach to a Scalable Quantum Computer	6
1.4 Notation and Preliminaries	9
1.4.1 Sequences, Groups and Rings	9
1.4.2 Quantum Circuits	15
2 Quantum Circuit Synthesis	24
2.1 Problem Statement	24
2.2 Early Methods – Solovay-Kitaev	26
2.3 Single Qubit Exact Synthesis – Matsumo-Amano Normal Forms	28
2.4 Multi-Qubit Exact Synthesis – Giles Selinger Algorithm	30
2.5 Approximate Synthesis in Single- and Multi-Qubit Circuits	31

2.6	New Directions and Thesis Outline	33
3	Matsumoto-Amano Normal Forms of Dimension Three	36
3.1	Introduction	36
3.2	Canonical forms	38
3.3	Uniqueness of canonical forms	43
3.3.1	Algebraic preliminaries	44
3.3.2	The adjoint representation	47
3.3.3	Uniqueness	55
3.4	Higher Prime Dimensions	72
3.5	Conclusion	77
4	Restricted Clifford + T Circuit Synthesis	79
4.1	Introduction	79
4.2	Overview	83
4.3	Rings and Matrices	85
4.3.1	Rings	85
4.3.2	Matrices	88
4.4	Circuits	89
4.5	Number-Theoretic Characterizations	95
4.5.1	The \mathbb{D} case	95
4.5.2	The $\mathbb{D}[\sqrt{2}]$ case	101
4.5.3	The $\mathbb{D}[\sqrt{-2}]$ case	103
4.5.4	The $\mathbb{D}[i]$ case	106
4.6	Conclusion	109
5	Clifford + Controlled Phase Exact Synthesis	111
5.1	Introduction	111
5.2	Generators	112
5.3	Exact Synthesis	118
5.4	Structure of Optimal Normal Forms	133
5.5	Conclusion	153
6	Clifford + Controlled Phase Inexact Synthesis	155

6.1	Introduction	155
6.2	Overview of Approximation Scheme	155
6.3	Unitary Templates and Lagrange Four-Squares	158
6.4	Finding Approximations with Small Least Denominator Exponent	159
6.4.1	Alternate Algorithm for Finding Approximations	162
6.5	Pauli-Rotation Approximations	172
6.6	Conclusion	175
7	Conclusion	177
A	Software Package for the Clifford + Controlled-Phase Gate Set	179
	Bibliography	211

List of Tables

5.1	Every generator and the explicit row pairings they will be used to reduce under earliest generator ordering.	131
-----	--	-----

List of Figures

4.1	Some subgroups of $\mathcal{U}_{2^n}(\mathbb{D}[\zeta])$. To the left of the cube, in yellow, the symmetric group S_{2^n} corresponds to circuits over the gate set $\{X, CX, CCX\}$. On the bottom face of the cube, in blue, are generalized symmetric groups, and on the top face of the cube, in red, are universal subgroups of $\mathcal{U}_{2^n}(\mathbb{D}[\zeta])$. The edges of the lattice denote inclusion. The gates labeling the edges are sufficient to extend the expressive power of a gate set from one subgroup to the next (and no further). For example, the edge labeled Z going from S_{2^n} to $\mathcal{U}_{2^n}(\mathbb{Z})$ indicates that adding the Z gate to $\{X, CX, CCX\}$ produces a gate set expressive enough to represent every matrix in $\mathcal{U}_{2^n}(\mathbb{Z})$ (but not every matrix in $\mathcal{U}_{2^n}(\mathbb{Z}[i])$).	82
5.1	Proof diagram for the “only if” direction of Lemma 5.4.8. In pane (a) , we observe that the first pair $s_1 \in V_k$ cannot be $\in V_{k-1}$. In pane (b) , we note that a pair $s_2 \in V_k$ must send exactly one element to s'_1 and one elsewhere. Finally, in pane (c) we see that $s_3 \in V_k$ cannot be paired in V_{k-1} , restricting the final outcome to a $2 \times 2 \times 2$ pairing with $V_k \cap V_{k-1} = \emptyset$ and $W_k \cap W_{k-1} = \emptyset$	139
5.2	Proof diagram for the “if” direction of Lemma 5.4.8. In pane (a) , we observe that the first pair $s'_1 \in V_{k+1}$ cannot be $\in V_k$ and likewise cannot be the 2-pairing in a 2×4 pairing. In pane (b) , we apply the same logic to $s'_2 \in V_{k+1}$, noting it may be part of a 4-pairing. Finally, in pane (c) we see that $s'_3 \in V_{k+1}$ has the same restrictions, forcing the final outcome to be a $2 \times 2 \times 2$ pairing with $V \cap V_{k+1} = \emptyset$ and $W_k \cap W_{k+1} = \emptyset$	140

5.3	Proof diagram for the “only if” direction of Lemma 5.4.9. In pane (a) , we observe that the pair $s_1 \in V_k$ cannot be $\in V_{k-1}$. In pane (b) , we note that the pair $s_3 \in V_k$ must be paired in V_{k-1} and cannot be a 2-pair in a 2×4 pairing. In pane (c1) , we see that if V_{k-1} is a $2 \times 2 \times 2$ pairing, the resulting sets in $V_{k-1} = W_{k-1}$ must be such that $W_{k-1} \subset \mathcal{E}$. In pane (c2) , if V_{k-1} is a 2×4 pairing then we see that the 2-pair $s'_1 \in \mathcal{E}$. Under EGO, the remaining pairs of W_{k-1} must then belong to the sets \mathcal{A} and \mathcal{B}	143
5.4	Proof diagram for the induction hypothesis of the “if” direction of Lemma 5.4.9. In pane (a) , we observe that $s'_1 \in \mathcal{E}$ which must form the 2-pair in the 2×4 pairing V_{k+1} must come from the 4-pairing $\in V_k$. In panes (b) and (c) , we form the 4-pairing of V_{k+1} from the remaining elements. Under EGO, $s'_2 \in \mathcal{A}$ and $s'_3 \in \mathcal{B}$	144
5.5	Proof diagram for the base case of the “if” direction of Lemma 5.4.9. In pane (a) , we observe that $s_1 \in \mathcal{E}$ must form the 2-pair s'_1 in the 2×4 pairing V_{k+1} . In panes (b) and (c) , we form the 4-pairing of V_{k+1} from the remaining elements. Under EGO, $s'_2 \in \mathcal{A}$ and $s'_3 \in \mathcal{B}$	144
5.6	Proof diagram for the “only if” direction of Lemma 5.4.10. In pane (a) , we observe that $s_1 \in \mathcal{B} \mathcal{A}$ must not be paired in V_{k-1} . In pane (b) , we see that $s_3 \in \mathcal{A} \mathcal{B}$ must remain paired in V_{k-1} . Pane (c1) establishes that in the case where V_{k-1} is a $2 \times 2 \times 2$ pairing, then W_{k-1} must correspond to one of $\{\mathcal{S}_4, \dots, \mathcal{S}_9\}$. In pane (c2) , we see that there can be instances where V_{k-1} is a 2×4 pairing such that W_{k-1} must likewise correspond to one of $\{\mathcal{S}_4, \dots, \mathcal{S}_9\}$. Finally, in pane (c2) , we show that there can likewise be instances where V_{k-1} is a 2×4 pairing such that W_{k-1} must correspond to one of $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$ with $W_k \cap W_{k-1} = \emptyset$	148
5.7	Proof diagram for the induction hypothesis of the “if” direction of Lemma 5.4.10. In pane (a) , we observe that $s'_1 \in \mathcal{B} \mathcal{A}$ which must form the 2-pair in the 2×4 pairing V_{k+1} must come from the 4-pairing $\in V_k$. In panes (b) and (c) , we form the 4-pairing of V_{k+1} from the remaining elements. Under EGO, $s'_2 \in \bar{\mathcal{E}}_{\supset}$ and $s'_3 \in \mathcal{A} \mathcal{B}$	149
5.8	Proof diagram for the base case of the “if” direction of Lemma 5.4.10. In pane (a) , we observe that $s_1 \in \mathcal{A} \mathcal{B}$ must form the 2-pair s'_1 in the 2×4 pairing V_{k+1} . In panes (b) and (c) , we form the 4-pairing of V_{k+1} from the remaining elements. Under EGO, $s'_2 \in \mathcal{B} \mathcal{A}$ and $s'_3 \in \bar{\mathcal{E}}_{\supset}$	149
6.1	The acceptable values of α fall in the green region which is a segment of the unit disk. This region can be contained within a rotated rectangle which has a width of $\frac{\varepsilon^2}{8}$ and a height of approximately ε	160

List of Abbreviations

$\mathbb{1}$	The Identity operator, dimension and type inferred
$\mathbb{1}_n$	The $n \times n$ $\mathbb{1}$ matrix
(A)	The sequence defined under lexicographic ordering of set A whose j th element is A_j
\mathcal{C}	The qudit Clifford group (dimension and qudit number inferred)
\mathcal{C}_n	The n qudit Clifford group (dimension inferred)
$\mathcal{C} + CCX$	The Clifford and CCX gate set
$\mathcal{C} + CS$	The Clifford and CS gate set
$\mathcal{C} + T$	The Clifford and T gate set
CS	The (non-Clifford) controlled Phase operator
CCX	The (non-Clifford) Toffoli gate for qubits or the CCSUM gate for dimension $p \geq 3$ qudits
$CCX_{a,b,c}$	CCX with explicit controls a and b and target c
CX	The CNOT gate for qubits or the CSUM gate for dimension $p \geq 3$ qudits
$CX_{a,b}$	CX with explicit control a and target b
\mathbb{D}	Ring of Dyadic Rationals
H	The Hadamard gate, a Clifford operator
i	The primitive 4th root of unity and imaginary unit
lde	Least Denominator Exponent (inferred)
lde_ϕ	Least Denominator Exponent with denominator ϕ
LLL	Lenstra-Lenstra-Lovász (lattice basis reduction algorithm)
$\mathcal{M}_{m \times n}$	The set of all $m \times n$ matrices
$\mathcal{M}_{m \times n}(R)$	$\mathcal{M}_{m \times n}$ whose entries belong to the ring R
$M[s_1; s_2]$	The submatrix of M for rows s_1 and columns s_2
MA	Matsumoto-Amano (normal form)
\mathbb{N}	Natural numbers (zero inclusive)
\mathcal{P}	The qudit Pauli group (dimension and qudit number inferred)
\mathcal{P}_n	The n qudit Pauli group (dimension inferred)
$[[p..q]]$	The set $\{p, p + 1, \dots, q\}$ for integral p, q
\mathbb{Q}	Ring of Rational numbers

$R[\phi]$	Ring extension of ring R by ϕ
S	The Phase gate, a Clifford operator
T	The qudit (non-Clifford) T gate
\mathcal{U}_n	The group of $n \times n$ unitary matrices over \mathbb{C}
$\mathcal{U}_n(R)$	The group of $n \times n$ unitary matrices over ring R
ω	A primitive p th root of unity (p inferred from context)
X	The Pauli X operator
Z	The Pauli Z operator
\mathbb{Z}	Ring of Integers
ζ	A primitive p^k th root of unity, (p and k inferred from context)

Chapter 1: Introduction

1.1 Quantum Computing and Quantum Compiling

Quantum computing [1,2] is a computing paradigm in which the quantum mechanical principles of superposition and entanglement are leveraged in conjunction with standard computational techniques in an attempt to improve our computing capabilities [3]. Based on a slew of novel algorithms for quantum computers [4–7], early attempts at quantifying the computational power of quantum computers seemed to suggest that these devices could in principle provide an exponential speedup in some important computational tasks. Further results have only strengthened these hopes for establishing “quantum supremacy” in practice [8–10]. Indeed, if recent results hold under scrutiny [11], we may have already crossed that threshold.

Fundamentally, a quantum computation consists of four major subroutines: (1) fiducial state preparation, (2) unitary evolution with sufficiently noiseless gates, (3) measurement, and (4) classical post-processing [12]. Complex algorithms [13] may involve many rounds of these operations and the use of quantum error correction [14, 15] can increase their intricacy; nonetheless, these four ingredients remain the sole building blocks of quantum algorithms for any architecture [16–20]. Each poses major challenges both theoretically and experimentally due to the fickle nature of

quantum systems. In this dissertation, we focus on the second of these hurdles – how we apply the appropriate unitary evolution of a quantum computer when we are given a gate set G , some target unitary U , and some acceptable tolerance for error ε .

In classical computing, a *compiler* is a computer program that takes as input some “high-level” instructions (a move in a game of Minesweeper, for example) and outputs an equivalent sequence of instructions which can be carried out on the physical hardware of the computer [21]. We say that a compiler forms some portion of the *computing stack*. For a “full-stack” quantum computer [22], we must develop a *quantum compiler* that translates some high-level unitary operation into a sequence of instructions which are executable by the native architecture [23]. We emphasize that without a quantum compiler to translate unitaries into a sequence of native gates for a given architecture, we simply would not know how to perform any of the quantum algorithms on a real device. Crucially, we want the quantum compiler to efficiently (say, polylogarithmically in $1/\varepsilon$) output as short a sequence as possible that approximately or exactly implements U . We also would like the quantum compiler to know whether the output is exactly or approximately equivalent to U . Our work advances all of these capabilities; before we delve headlong into these problems, we first describe the types of unitary operations we are interested in implementing, as well as the restrictions placed on the available instruction set for some hypothetical full-stack quantum computer.

1.2 Applications of Quantum Computing

There is an ever-expanding zoo of quantum algorithms which outperform their classical counterparts [24]. However, boiling down this vast list of algorithms to those that are both most important historically and most flexible to usage in a wide array of contexts, we arrive at four major applications: identifying hidden subgroups [25] (a subclass of which is factoring numbers [5]), unsorted search of a database [26,27], Hamiltonian simulation [7, 28–34], and linear equation solving [35–37]. We briefly discuss the importance of these algorithms and the quantum subroutines used in their implementation according to state-of-the-art algorithms.

1.2.1 Hidden Subgroup Problem

Shor’s Factoring algorithm [5] was one of the foundational linchpins in cementing interest in quantum computing. Though it is perhaps merely a historical curiosity that quantum computers can efficiently solve the problem underlying one of the most popular encryption schemes [38], this algorithm along with its generalized counterpart, the hidden subgroup problem [25], have proven to be vital cornerstones in our understanding of these devices capabilities. The routine provides sub-exponential (but super-polynomial) speedup over the best known classical algorithm [39, 40].

At its core, the most basic implementation of Shor’s algorithm uses phase estimation [41, 42], consisting of two quantum subroutines, to factor some integer N . The first portion of the routine involves applying a singly-controlled unitary (a

gadget for modular exponentiation) which can act on any number of qubits. These circuits can be constructed efficiently from $\mathcal{O}(\log^3 N)$ gates consisting of quantum versions of the Classical reversible gates NOT, CNOT, and Toffoli. The second portion involves applying the inverse quantum Fourier transform. This operation is common among quantum algorithms, and can be implemented approximately using $\mathcal{O}(\log N \log \log N)$ singly-controlled Z-rotations of a qubit and the basis-changing Hadamard gate.

1.2.2 Unsorted Search of a Database

Grover's search algorithm [26, 27] finds a particular element in some unsorted set, providing a quadratic speedup over the best possible classical algorithm. While the improvement is not wholly overwhelming, the underlying principles of a Grover search have proven to be a vital backbone to many other quantum algorithms. Implementation of the algorithm involves applying Householder reflections about some target state $|x\rangle$ within the search space. Finding a unitary that implements such a reflection effectively amounts to finding a unitary that can transform $|0\rangle \rightarrow |x\rangle$ and implementation of the fully-controlled n qubit Z operation. The latter problem can be done using $\mathcal{O}(n)$ Toffoli, CNOT, and NOT operations, whereas the former problem is essentially that of general state-preparation, dependent on the quantum representation of the state one is searching for.

1.2.3 Hamiltonian Simulation

While the two aforementioned problems provide a lot of the machinery for developing new quantum algorithms, perhaps no quantum algorithm will prove as useful in the coming years as Hamiltonian simulation. The vast applications in physics, chemistry, and biology will play a major factor in driving commercial interest in quantum computing. There are essentially four major methods for performing Hamiltonian simulation – product formulas [7, 28, 29], Taylor series truncation [30], quantum walks [31], and quantum signal processing [33, 34]. Describing these algorithms in detail is beyond the scope of this dissertation. However, the first three techniques essentially rely on breaking the simulation unitary into significantly simpler unitaries which act on only a few qubits at a time. Quantum walks and quantum signal processing use phase estimation as in Section 1.2.1. In both cases, the types of circuits that need to be implemented are unitaries with support on only a few qubits.

1.2.4 Linear Equation Solving

Linear equations are so ubiquitous that any improvement to algorithms for linear systems would be significant. The HHL algorithm [35] manages to do just this by mapping the problem of linear system solving onto Hamiltonian simulation. It works by constructing a Hermitian matrix from any invertible matrix and then using Hamiltonian simulation as a sub-routine in phase estimation to solve any invertible linear system [35–37]. Moreover, the algorithm requires preparation of a

few special input states. Having effective methods to approximate unitaries of the various forms in Sections 1.2.1 to 1.2.3 therefore suffices to ensure that we can find approximations to the unitaries used in the linear system problem as well.

1.3 Approach to a Scalable Quantum Computer

The algorithms of Section 1.2 provide instruction sets that tell quantum computers how to solve important problems assuming the quantum computer operates without error. In reality, quantum computers are inherently noisy devices [43, 44] – there is simply no way to build a physical qubit that retains its quantum information for long enough to perform an arbitrarily long computation. The path towards scalable quantum computing then relies on some level of *fault-tolerance* [45], or the ability to mitigate error to arbitrary precision. A long-standing paradigm exists where fault-tolerance is achieved through two ingredients [46–51]: quantum error correction and transversality. These are mutually beneficial concepts, with the caveat that some extra effort [52, 53] is required to enable their simultaneous use.

Quantum error correction [15] works by using “extra” Hilbert space to offload any unwanted evolution while protecting the relevant quantum information. Effectively, some larger-than-necessary Hilbert space on your physical qubits is divided into two subspaces - one encoding the “logical” qubits on which you are performing your calculations, and one into your “error” subspace. Using quantum non-demolition measurements, we intermittently project the state of the qubit back into one of these two subspace completely. Based off of the measurement results, we

are able to either (1) conclude that the logical qubit has been projected back into the logical subspace completely or (2) decide whether and what kind of error has occurred. We can then apply any necessary corrective operations on the physical qubits to always land back in the logical qubit subspace. This scheme is remarkably robust, in principle only sensitive to errors that affect more than some specified threshold of physical qubits. Using a quantum error correction scheme such as concatenated [50, 54] or surface codes [55, 56] ensures that given a physical qubit error rate below some fixed threshold (estimates show this could be as high as 1%), one may carry out a successful computation with very high probability.

One issue with the stabilizer formalism of quantum error correction is that it does not specify how errors tend to propagate within a computation. Consider the scenario of a classical computer which uses a single control bit for a large computation. Because this single bit is used to influence the states of all the other bits, any bit-flip error here would drastically alter the result of the computation. A similar scenario can occur in a quantum computation. Say we wished to perform some relatively straightforward operation on a logical qubit (a bit-flip, perhaps). While this looks like a simple computation in the logical qubit basis, it could in fact be a complex operation on the physical qubits due to the nontrivial nature of quantum error correcting code logical states. An error on one of the physical qubits could easily propagate to a sufficiently large collection of qubits so as to put the number of errors beyond what the quantum error correcting code is capable of correcting.

To combat this propagation of errors between code blocks, we rely on the *transversality* [47, 57, 58] of gates within special families of quantum error correcting

codes (including those mentioned above). Transversality can be summed up by the following idea: simple single qubit operations at the logical level correspond to simple single qubit operations at the physical level. Likewise, simple two-qubit operations at the logical level correspond to simple two-qubit operations at the physical level such that physical qubit n of logical qubit one only interacts with physical qubit n of logical qubit two. Using only operations of this form ensures that we can keep errors “quarantined” from one another by design.

Unfortunately, it has been proven [52] that if a gate set is universal, then that gate set cannot be implemented fully transversally for a quantum error correcting code. Oftentimes, some large subset of the fundamental gates indeed *can* be so implemented [59, 60]; generally speaking, for most basic error correction protocols, these are the *Clifford operators*, which will be defined in Section 1.4. To complete the realization of a fault-tolerant quantum computer, we then need a method to institute some sufficiently error-resistant non-Clifford gate. Fortunately, we can implement these operators through a scheme called *Magic State distillation* [49, 53].

Magic State distillation is the procedure of producing sufficiently noiseless special quantum states which can be used as a resource to implement certain non-Clifford gates fault-tolerantly. The gates which states in this scheme can implement belong to a family of gates called the *Clifford Hierarchy* [61–64], which will be discussed briefly in Section 1.4. Magic State distillation is an iterative procedure which uses state preparation, Clifford circuits, and measurement. To produce states below the required error tolerance can take many iterations, in turn incurring a significant computational cost. In practice, when studying circuit cost for realistic

quantum computation models using quantum error correction, transversal Cliffords, and magic state distillation, virtually the entire cost comes from the number of magic states required for the computation. As the number of magic states needed is effectively the number of non-Clifford gates from the Clifford Hierarchy used in the implementation of a unitary, this metric is often used for the complexity of said unitary [65–67]. For our purposes we will assume to be constructing unitaries under this paradigm.

1.4 Notation and Preliminaries

Before advancing further, we will develop notation and introduce a few definitions upon which the remainder of this dissertation will rely heavily.

1.4.1 Sequences, Groups and Rings

Here we provide basic definitions of some mathematical terms for completeness. We assume familiarity with *sets* and set builder notation. In all that follows, \mathbb{Z} denotes the set of integers, \mathbb{N} the set of nonnegative integers, \mathbb{Q} is the set of rational numbers, \mathbb{Z}_n the set of integers modulo n , and $[[p..q]] \subset \mathbb{N}$ is the set $\{p, p + 1, \dots, q\}$. We denote the set of $m \times n$ matrices by $\mathcal{M}_{m \times n}$. For the curious reader, these definitions and more can be found in various mathematics texts [68, 69].

A *sequence*, like a set, is a collection of mathematical objects; however, sequences have a fixed ordering and repetitions are allowed. We denote a length n sequence as (x_1, x_2, \dots, x_n) , where the j th element of the sequence is x_j . Sometimes,

we refer to the x_j as *syllables* and the entire sequence as a *word* over these syllables. In an abuse of notation, when we would like to build a sequence from a set A equipped with some explicit lexicographic ordering, we write (A) and interpret A_j as the j th element of that sequence according to the specified ordering.

A *group* is a set G equipped with a binary associative operation (\cdot) , formally denoted (G, \cdot) . Two objects $a, b \in G$ if $a \cdot b \in G$. A group must always have a unique element called the identity $\mathbb{1}$ such that for all $a \in G$, $\mathbb{1} \cdot a = a \cdot \mathbb{1} = a$. Finally, every $a \in G$ has a unique inverse $a^{-1} \in G$ such that $a \cdot a^{-1} = a^{-1} \cdot a = \mathbb{1}$. When the group operation is some type of multiplication, we will commonly drop the (\cdot) in $a \cdot b$ and instead write ab . In the case where the group operation is commutative, we call the group *Abelian*. Some common examples of groups are $(\mathbb{Z}, +)$, $(\mathbb{Z}_n, +)$, $([[1..p-1]], \cdot)$ for p prime, or the group of $n \times n$ unitary matrices under matrix multiplication \mathcal{U}_n , the first three being Abelian and the last non-Abelian.

To concisely define a group, rather than specify every element of that group we will usually supply the *generators* of that group. Under action of the group operation, every element of the group is then some explicit sequence, or word, over these generators. We have a *presentation* of a group when along with these generators, we supply a full set of *relations* for the group. These relations usually dictate how the generators “commute” with one another, as well as the *order* (how many applications of the group operation with itself take an element to the identity) of individual generators. In the case where these relations are sufficiently simple to utilize in “rewriting” elements of a group, we can produce a *normal form* for a group. Normal forms, also sometimes called *canonical forms*, try to present a group by

explicitly constructing a standard expression for any element of the group. Finding normal forms generally consists of establishing two things: *existence* and *uniqueness*. A normal form for a group exists if every element of that group can be put into that explicit form, and it is unique if two different instances of a normal form are always different elements of the group. In the language of sequences, we usually say that a normal form consists of words over some set of syllables, where the syllables may just be the original generators themselves or some composition of them under the group operation.

A group G always contains *subgroups*, which are subsets of G that are themselves groups. We write $H \leq G$ if H is a subgroup of G . Given $H \leq G$, the *left* and *right cosets* of H containing $a \in G$ are $aH = \{a \cdot b \mid b \in H\}$ and $Ha = \{b \cdot a \mid b \in H\}$ respectively. The union of all left/right cosets of H are then equal to G . The *normalizer* of a subset S of group G is defined as the set $N_G(S) = \{g \mid g \in G, gS = Sg\}$. Finally we note the definition of a *group homomorphism*: given some function $\phi : G \rightarrow G'$ where G and G' are groups with binary operations (\cdot) and (\star) respectively, ϕ is a group homomorphism if for all $a, b \in G$, $\phi(a) \star \phi(b) = \phi(a \cdot b)$.

A *ring* R is a group under the “additive” binary operation $(+)$ that is also equipped with a “multiplicative” operation (\cdot) . The operation (\cdot) must be associative, distributive with respect to $(+)$, and R must contain a multiplicative identity element $\mathbb{1}_{(\cdot)}$ such that for all $r \in R$, $r \cdot \mathbb{1}_{(\cdot)} = \mathbb{1}_{(\cdot)} \cdot r = r$. When the ring operations are commutative, we again call the ring *Abelian*. Some common examples of Abelian rings include \mathbb{Z} , \mathbb{Q} , and \mathbb{Z}_n , and one non-Abelian ring is $\mathcal{M}_{n \times n}(R)$, the ring of square $n \times n$ matrices whose entries belong to the ring R .

Much like groups, we have the notion of *subrings* of a ring, i.e. a ring contained within another ring. We can also extend the group homomorphism to make it compatible with rings: given $\phi : R \rightarrow R'$ where R and R' are rings with binary operations $(+, \cdot)$ and (\ddagger, \star) respectively, ϕ is a *ring homomorphism* if for all $a, b \in R$, $\phi(a) \ddagger \phi(b) = \phi(a + b)$, $\phi(a) \star \phi(b) = \phi(a \cdot b)$, and $\phi(\mathbb{1}_{(\cdot)}) = \mathbb{1}_{(\star)}$. Any element of a ring with a multiplicative inverse is called a unit, and if a ring is Abelian and every element is a unit, we call it a *field*.

There are special subsets of a ring called *ideals*. For a ring R with additive operation $(+)$ and multiplicative operation (\cdot) , the set I is a *left ideal* if $I \subseteq R$ forms a group under $(+)$ and for every $r \in R$ and $a \in I$, $r \cdot a \in I$. Right ideals are defined similarly, with $a \cdot r \in I$. When a set is both a left and right ideal, we simply refer to it as an ideal. A classic example of an ideal is the even integers $2\mathbb{Z}$; every integer times an even integer yields another even integer, and $2\mathbb{Z}$ is a group under addition.

We also briefly describe the *residue* of a ring. If R is a ring and $r \in R$ we write $R/(r)$ for the quotient of the ring R by the ideal generated by the element r . Two elements s and s' of R are congruent modulo r if $s - s' \in R/(r)$, in which case we write $s \equiv s' \pmod{r}$. We sometimes refer to the elements of the ring $R/(r)$ as residues. Some quotient rings are well-known. For example, $\mathbb{Z}/(2) = \{0, 1\}$ and $\mathbb{Z}/(4) = \{0, 1, 2, 3\}$.

Oftentimes when we define rings, we start with some ring R (generally, the integers or some other common ring) and supply it with a new element ψ called an *extension*, denoting the new *extension ring* of R as $R[\psi]$. This new extension ring

is precisely defined as

$$R[\psi] = \{P(\psi) \mid P \text{ a polynomial with coefficients in } R\}$$

Sometimes, there exists an integer n for which a non-trivial monic polynomial of degree n with coefficients in R is equal to zero. In this case, the polynomial P can always be reduced to one of degree $\leq n - 1$. Other times no such polynomial exists, and all powers of ψ must be included when defining $R[\psi]$. We can also supply R with more than one extension, where it is understood that $R[\psi, \xi] = (R[\psi])[\xi] = (R[\xi])[\psi]$.

Because we shall use them so frequently, we define a few special rings here. First, consider a primitive p th root of unity ω for $p \in \mathbb{N}$. By calling ω a primitive p th root of unity, we mean that $\omega^p = 1$ and every β that satisfies $\beta^p = 1$ must be expressible as ω^n for some integer n . Then we define the *ring of cyclotomic integers of degree p* as

$$\mathbb{Z}[\omega] = \left\{ \sum_{j=0}^{\phi(p)} a_j \omega^j \mid a_j \in \mathbb{Z} \right\}$$

where ϕ is Euler's totient function, i.e. the number of positive integers $\leq p$ that are coprime to p . We also note that the following elements always appear in cyclotomic fields:

- $8 \mid p \implies \sqrt{2} \in \mathbb{Z}[\omega]$
- $p \equiv 1 \pmod{4} \implies \sqrt{p} \in \mathbb{Z}[\omega]$
- $p \equiv 3 \pmod{4} \implies i\sqrt{p} \in \mathbb{Z}[\omega]$

One important instance of a field of cyclotomic integers is that of degree 4, i.e. $\mathbb{Z}[i]$

which we call the *Gaussian integers*. We also define a subring of \mathbb{Q} , which is called the *dyadic rationals* \mathbb{D} :

$$\mathbb{D} = \mathbb{Z} \left[\frac{1}{2} \right] = \left\{ \frac{a}{2^k} \mid a \in \mathbb{Z}, k \in \mathbb{N} \right\}.$$

In defining \mathbb{D} , we see that we can extend some ring R with an extension $\frac{1}{\psi}$ where $\psi \in R$. In these cases, we can always write

$$R \left[\frac{1}{\psi} \right] = \left\{ \frac{a}{\psi^k} \mid a \in R, k \in \mathbb{N} \right\}.$$

In such cases, we call k a *denominator exponent*. As $\psi \in R$,

$$r = \frac{a}{\psi^k} = \frac{\psi a}{\psi^{k+1}}$$

would imply there are an infinite number of denominator exponents for this element $r \in R \left[\frac{1}{\psi} \right]$. To avoid such ambiguity, we will generally refer to the *least denominator exponent* (lde) of r , which is the smallest k' for which

$$r\psi^{k'} \in R.$$

We can extend the notion of least denominator exponent to any tensor \mathcal{T} whose entries belong to such a ring $R \left[\frac{1}{\psi} \right]$: k is the lde of \mathcal{T} if it is the smallest integer for which $\psi^k \mathcal{T}$ only has entries in the ring R . Any other rings referred to in this dissertation will be explicitly defined in their relevant chapters.

1.4.2 Quantum Circuits

For basics in Quantum circuits, we refer the reader to standard quantum computing texts [70, 71]. A *quantum circuit* is a sequence of *quantum logic gates* (often shortened to simply gates), which themselves are reversible transformations on a *quantum register* (essentially, physical objects which may exist a quantum state). In this dissertation, we concern ourselves with quantum circuits which correspond to unitary evolution of the register, leaving the other three aspects of quantum computing aside. We will use two different representations for a quantum circuit – quantum circuit notation and operator notation. We emphasize now that though these representations are equivalent, they are read in *opposite* directions: quantum circuit diagrams have their gates applied to the register “left-to-right” (consistent with the classical circuit model), whereas operator formalism demands the operators be applied “right-to-left” (consistent with matrix multiplication).

Let p be a prime number, and $\omega = \exp\left(\frac{2\pi i}{p}\right)$ a primitive p th root of unity. We consider gates on *qudits* which individually live in a Hilbert space of dimension p^k for some $k \in \mathbb{N}$. When $p = 2$ or 3 , we call these qudits *qubits* or *qutrits*, respectively. The first basic building block of fault-tolerant gate sets is the *Pauli*

group [15, 70, 72–74]. The single-qudit Pauli group is generated by

$$X := \sum_{j=0}^{p-1} |j+1\rangle \langle j| = \begin{bmatrix} 0 & 1 \\ \mathbb{1}_{p-1} & 0 \end{bmatrix} \quad \text{and} \quad Z := \sum_{j=0}^{p-1} \omega^j |j\rangle \langle j| = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \omega & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \omega^{p-1} \end{bmatrix}.$$

We note that the phase $\omega = ZXZ^\dagger X^\dagger$ is explicitly a member of the Pauli group.

For qubits, we also supply the additional generator

$$Y := \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$

which is equivalent to saying that the phase i is present in the qubit Pauli group.

We construct the *multi-qudit Pauli group* via tensor product. Then every member of the n -qudit Pauli group \mathcal{P}_n can be written as

$$\mathcal{P}_n = \{\omega^w (X^{x_1} Z^{z_1} \otimes X^{x_2} Z^{z_2} \otimes \cdots \otimes X^{x_n} Z^{z_n}) \mid x_j, z_j, w \in \mathbb{Z}_p\}$$

for prime dimension $p \geq 3$, and as

$$\mathcal{P}_n = \{i^w (P_1 \otimes P_2 \otimes \cdots \otimes P_n) \mid P_j \in \{\mathbb{1}, X, Y, Z\}, w \in \mathbb{Z}_4\}$$

for qubits. For n qubits, $|\mathcal{P}_n| = 4^{n+1}$, and for n qudits $|\mathcal{P}_n| = p^{2n+1}$. Often, we will drop the n subscript from \mathcal{P}_n and make no explicit reference to which dimension p

we are working in when it is clear from context.

The *Clifford group* \mathcal{C}_n on n qudits is the normalizer of \mathcal{P}_n (which therefore is a subgroup of \mathcal{C}_n). The single qudit version of this group is generated in dimension $p = 2$ by [70]

$$H := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad S := \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

and dimensions $p \geq 3$ [72–74] by the operators

$$H := \frac{1}{\lambda_p \sqrt{p}} \sum_{j,k=0}^{p-1} \omega^{j \cdot k} |j\rangle \langle k| = \frac{1}{\lambda_p \sqrt{p}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{p-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(p-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{p-1} & \omega^{2(p-1)} & \cdots & \omega^{(p-1)^2} \end{bmatrix},$$

$$S := \sum_{j=0}^{p-1} \omega^{\frac{j(j-1)}{2}} |j\rangle \langle j| = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \omega & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \omega^{\frac{(p-1)(p-2)}{2}} \end{bmatrix},$$

where we call H the *Hadamard* operator and S the *Phase* operator (capital P here to distinguish from generic phase operators). For the qubit case, we note that the primitive eighth root of unity $e^{\frac{\pi i}{4}}$ can be constructed from H and S . In the $p \geq 3$ case, we have used λ_p to denote a multiplicative phase on H relative to

the “standard” definitions to enforce $\det H = 1$ for these dimensions [75]. The appropriate λ_p to achieve this is

$$\lambda_p = \begin{cases} 1 & p \equiv 1 \pmod{8} \\ -i & p \equiv 3 \pmod{8} \\ -1 & p \equiv 5 \pmod{8} \\ i & p \equiv 7 \pmod{8} \end{cases}$$

We shall also define $\lambda_2 = 1$. S has determinant ω in dimension $p = 3$ and determinant 1 in higher dimensions.

To generate the full n qudit version of the Clifford group, we must also add the entangling CX gate (sometimes called CSUM in the qudit dimension $p \geq 3$ case and CNOT in the qubit case). This gate acts on two qudits and is defined as

$$CX = \sum_{j,k=0}^{p-1} |j, j+k\rangle \langle j, k| = \mathbb{1}_p \oplus X \oplus X^2 \oplus \dots \oplus X^{p-1} = \begin{bmatrix} \mathbb{1}_p & 0 & \dots & 0 \\ 0 & X & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & X^{p-1} \end{bmatrix}.$$

CX has determinant -1 in the qubit case and 1 otherwise. Thus, for dimension $p = 2$, \mathcal{C}_1 operators have a determinant of some power of i , \mathcal{C}_2 a determinant of ± 1 , and \mathcal{C}_n for $n \geq 3$ a determinant of 1. For dimension $p = 3$, \mathcal{C}_1 operators have a determinant of some power of ω and a determinant of 1 for multi-qudit Cliffords.

For dimension $p \geq 4$, every Clifford operator has a determinant of 1. The cardinality of $|\mathcal{C}_n|$ under these definitions up to a phase is [76, 77]

$$|\mathcal{C}_n| = p^{n^2+2n} \prod_{j=1}^n (p^{2j} - 1). \quad (1.1)$$

With phases included, $|\mathcal{C}_n|$ gains a factor of 8 for qubits and p for qudits of dimension $p \geq 3$.

We would also like to highlight that there is an extremely convenient representation for the n -qudit Clifford group using the group $\text{SL}(2n, \mathbb{Z}_p)$ [77–79]. Every operator in $\text{SL}(2n, \mathbb{Z}_p)$ corresponds to an n -qudit Clifford. In this representation, for single qudits

$$H = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \quad (1.2)$$

Effectively, these 2×2 matrices can be interpreted as a mapping of the Paulis to themselves in the following way. For $\hat{C} \in \text{SL}(2n, \mathbb{Z}_p)$ corresponding to Clifford operator C , we have

$$\hat{C} \cdot \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} x' \\ z' \end{bmatrix} \iff CX^x Z^z C^\dagger = \omega^w X^{x'} Z^{z'}$$

for $w, x, z \in \mathbb{Z}_p$. By construction, this group is only able to describe Clifford operators up to a rightmost Pauli operator, i.e. it can describe uniquely the operators of \mathcal{C}/\mathcal{P} . Nonetheless, they can make calculations for large dimensional qubits very

simple.

The groups \mathcal{P} and \mathcal{C} form the first and second levels of the *Clifford Hierarchy* [61–64]. A gate G belongs to the n th level of the hierarchy if it maps every element of the Pauli group to the $(n-1)$ th level of the hierarchy under conjugation. This family of operators forms the basis of operations that can be implemented fault-tolerantly, as described in Section 1.3.

As the Clifford group is finite, it cannot be a universal gate set [79, 80]. In order to “promote” our gate set up to a universal one, we merely need to add any non-Clifford operator [81]. Common choices generally come from the third level of the Clifford Hierarchy, as these gates are essentially the simplest gates constructable in a fault-tolerant manner through Magic States [49, 53]. Some common examples are the single qudit T gate [82, 83], the two-qudit Controlled Phase gate CS , and the three qudit CCX gate (usually known as Toffoli for qubits or CCSUM for qudits).

These are

$$\begin{aligned}
T &= \sum_{j=0}^{p-1} \zeta_p^{j^3} |j\rangle \langle j| = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \zeta_p & 0 & \cdots & 0 \\ 0 & 0 & \zeta_p^8 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \zeta_p^{(p-1)^3} \end{bmatrix} \\
CS &= \sum_{j=0}^{p-1} |j\rangle \langle j| \otimes S^j = \mathbb{1}_p \oplus S \oplus S^2 \oplus \cdots \oplus S^{p-1} = \begin{bmatrix} \mathbb{1}_p & 0 & \cdots & 0 \\ 0 & S & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & S^{p-1} \end{bmatrix} \\
CCX &= \sum_{j,k=0}^{p-1} |j, k\rangle \langle j, k| \otimes X^{j \cdot k} = \mathbb{1}_{p^2} \oplus CX \oplus CX^2 \oplus \cdots \oplus CX^{p-1} \\
&= \begin{bmatrix} \mathbb{1}_{p^2} & 0 & \cdots & 0 \\ 0 & CX & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & CX^{p-1} \end{bmatrix}
\end{aligned}$$

where we have defined

$$\zeta_p = \begin{cases} e^{\frac{\pi i}{4}} & p = 2 \\ e^{\frac{2\pi i}{9}} & p = 3 \\ \omega = e^{\frac{2\pi i}{p}} & p > 3 \end{cases}$$

When we refer to universal gate sets constructed using the Clifford group, we will

generally refer to them as Clifford + (non-Clifford Gate). For example, here we have specified gates to construct the universal gate sets $\mathcal{C} + T$, $\mathcal{C} + CS$, and $\mathcal{C} + CCX$. Per our discussion of fault-tolerance in Section 1.3, we know that minimizing the number of these non-Cliffords is an effective strategy to ensure the smallest circuits in practice. To that end, we define the *T-count*, *CS-count*, *CCX-count*, and in general *non-Clifford count* as the number of occurrences of the requisite gate in the sequence corresponding to a quantum circuit.

When we specify a gate set, we merely specify a short set of families of gates. Say we had a gate set $\{A, B, C\}$, where A and B are single-qudit gates and C is an (asymmetric) two-qudit gate. What this actually means is that an n qudit quantum computer has at its disposal the gates $\{A_1, A_2, \dots, A_n\}$, $\{B_1, B_2, \dots, B_n\}$, and $\{C_{1:2}, C_{2:1}, C_{1:3}, C_{3:1}, \dots, C_{n-1:n}, C_{n:n-1}\}$. Sometimes, quantum architectures preclude the use of multi-qudit gates between qudits that are sufficiently physically separated. In these cases, we can use the *swap* operation (a completely “local” Clifford) to bring the qudits adequately close before performing the intended operation and swapping them back to their original location.

We note that every reversible classical operation, up to the addition of a single ancilla, is implementable over the gate set $\{X, CX, CCX\}$ [84]. In their quantum forms, these gates constitute every $2^n \times 2^n$ permutation matrix for qubits [85]. For dimension $p \geq 3$ qudits, this no longer holds, but the same statement can be recouped upon addition of further permutations which live in the Clifford group and which shall be described in Section 3.4. We also see the following statement holds

based on our construction here:

$$\mathcal{C} + T \subseteq \mathcal{U}_{p^n \times p^n} \cap \mathcal{M}_{p^n \times p^n}(\mathbb{Z}[\frac{1}{\lambda_p \sqrt{p}}, \zeta_p]) \quad (1.3)$$

$$\mathcal{C} + CS \text{ and } \mathcal{C} + CCX \subseteq \mathcal{U}_{2^n \times 2^n} \cap \mathcal{M}_{2^n \times 2^n}(\mathbb{Z}[\frac{1}{\sqrt{2}}, i]) \text{ for qubits} \quad (1.4)$$

$$\mathcal{C} + CS \text{ and } \mathcal{C} + CCX \subseteq \mathcal{U}_{p^n \times p^n} \cap \mathcal{M}_{2^n \times 2^n}(\mathbb{Z}[\frac{1}{\lambda_p \sqrt{p}}, \omega]) \text{ for qudits} \quad (1.5)$$

for λ_p and ζ_p as defined earlier.

Chapter 2: Quantum Circuit Synthesis

With an understanding of the types of unitaries we commonly need to implement laid out in Section 1.2 and a paradigm of fault-tolerant quantum computation laid out in Section 1.3, we can now forge onward in understanding how best to build a quantum compiler.

2.1 Problem Statement

We begin by describing in precise terms the *quantum compiling*, or *quantum circuit synthesis*, problem [23, 70, 86, 87]:

Definition 2.1.1 (Quantum Compiling Problem). Let U be a target unitary on n p -level qudits and ε be an error tolerance acceptable under some matrix norm \mathcal{N} . Let $G = \{G_1, \dots, G_m\}$ be a universal gate set on $n' \geq n$ p -level qudits. Let c be a sensible cost function that takes as input any length $k \geq 0$ sequence of gates from G and outputs a non-negative real number which measures the complexity of that sequence as a circuit. Then we solve the *inexact* or *approximate* quantum compiling problem by finding some unitary

$$\tilde{U} = G_{j_l} G_{j_{l-1}} \cdots G_{j_1}$$

with associated circuit sequence $(\tilde{U}) = (G_{j_1}, G_{j_2}, \dots, G_{j_l})$ such that

$$\mathcal{N}(U \otimes \mathbb{1}_{p^{n'-n}} - \tilde{U}) \leq \varepsilon.$$

We are only interested in those \tilde{U} which can be found efficiently – say polylogarithmically in $\frac{1}{\varepsilon}$. Moreover, we would like $c((\tilde{U}))$ to be as close to minimal as possible. In the case where U is *exactly* expressible over G , we further demand that we have a method to find (\tilde{U}) such that $\tilde{U} = U$, which we call the *exact* quantum compiling problem.

We have left some details nebulous in Definition 2.1.1: for example, we have not specified how exactly the unitary U is supplied, nor have we given any bound on just how low we would like the output of c to be for a given candidate solution before we stop trying to find others. For practical purposes, the number of qudits on which U acts can be taken to be small – as detailed in Section 1.2, the vast majority of quantum subroutines reduce to the product of many relatively small unitary operators. This ultimately means that how we specify U is not a major concern for our compiler. As for the second question, we can use a simple volume-counting argument to determine a lower bound on gate count. The volume of the special unitary group [88, 89] on n p -dimensional qudits is

$$\text{vol}(\text{SU}(p^n)) = \frac{\sqrt{n \cdot 2^{p^n-1}} \pi^{\frac{(p^n-1)(p^n+2)}{2}}}{\prod_{k=1}^{p^n-1} k!}. \quad (2.1)$$

and the volume of a small ε -ball in the $p^{2n} - 1$ dimensional space of $SU(p^n)$ is

$$\text{vol}(\varepsilon\text{-ball}) = \frac{\pi^{\frac{p^{2n}-1}{2}}}{\Gamma\left(\frac{p^{2n}-1}{2} + 1\right)} \varepsilon^{p^{2n}-1}. \quad (2.2)$$

The total number of possible circuits with a non-Clifford count of exactly k for a gate set consisting of special unitary versions of \mathcal{C} and a single non-Clifford gate is certainly no more than $|\mathcal{C}_n|^{k+1}$, as this would constitute every possible circuit consisting of alternating Clifford and non-Clifford operators. Then the total number of circuits up to a non-Clifford count of k is no more than $\alpha^{k+1}|\mathcal{C}_n|^{k+2}$ for some constant $0 \leq \alpha \leq 1$. If each of these operators were surrounded by an ε -ball, their volume must exceed the total volume of $SU(p^n)$. This means we must have

$$\begin{aligned} \text{vol}(SU(p^n)) &\lesssim \alpha^{k+1}|\mathcal{C}_n|^{k+2} \cdot \text{vol}(\varepsilon\text{-ball}) \\ \implies k &\sim \mathcal{O}\left((p^{2n} - 1) \log_{\alpha|\mathcal{C}_n|}\left(\frac{1}{\varepsilon}\right)\right) \end{aligned} \quad (2.3)$$

As a rough first estimate, the best we can hope to do is find circuits of length linear in the logarithm of $1/\varepsilon$. Any gate set that can achieve this bound is *efficiently universal* [23]. Moreover, any algorithm that achieves this bound up to a constant multiplicative value for small values of ε is called *asymptotically optimal*.

2.2 Early Methods – Solovay-Kitaev

One of the foundational results in quantum information, the Solovay-Kitaev algorithm [86] was the first solution to the quantum compiling problem. In precise

terms, the Solovay-Kitaev algorithm is a classical algorithm which solves the inexact quantum compiling problem, runs in a time $\mathcal{O}(\log^{2.71}(\frac{1}{\varepsilon}))$ and produces gate counts of $\mathcal{O}(\log^{3.97}(\frac{1}{\varepsilon}))$ for single qubit gate approximations. This construction can likewise be extended to multi-qubit and qudit circuits, ensuring that we have a solution to the inexact quantum compiling problem in the general case. Informally, this algorithm states that regardless of your choice of universal gate set, that gate set will fill operator space sufficiently densely to approximate any operator with relatively short circuits.

The algorithm works roughly as follows. First, a sufficiently dense subset of circuits are generated from the gate set (say, all circuits up to gate count 10). The closest candidate among these, U_0 , is used as a zeroth order approximation for the target unitary U . The operator UU_0^\dagger is then decomposed into a balanced group commutator between two operators V and W as $UU_0^\dagger = VWV^\dagger W^\dagger$. Zeroth order approximations V_0 and W_0 are found for these operators, and the resulting first order approximation $U_1 = V_0W_0V_0^\dagger W_0^\dagger U_0$ is returned. Higher order approximations may be found using recursive calls of the algorithm. Given a starting set that is sufficiently dense, the algorithm is guaranteed to succeed.

That Solovay-Kitaev is agnostic to its choice of universal gate set is both its strength and its weakness. Use of the group commutator provides better-than-expected approximations that culminate in the gate count being polylogarithmic in $1/\varepsilon$. However, little information about the structure of the generating gate set is used – a fact that would belie the discovery of improved algorithms. In particular, the best scaling that any Solovay-Kitaev-like algorithm could hope to achieve is

$\mathcal{O}(\log^2(1/\varepsilon))$ [23].

2.3 Single Qubit Exact Synthesis – Matsumo-Amano Normal Forms

While it was known that some gate sets were efficiently universal [23, 23], it was not known how to exactly compile some target unitary known to be expressible over these gate sets. By harnessing the internal structure of the group relations for $\mathcal{C} + T$, Matsumoto and Amano were able to construct a normal form [90], dubbed a Matsumoto-Amano (MA) normal form, in the single qubit case. Remarkably, the MA normal form is T -count optimal – for an operator U with circuit (U) in MA normal form, no equivalent $\mathcal{C} + T$ circuit can be found for U that uses fewer T gates than (U) .

MA normal form for single qubit $\mathcal{C} + T$ circuits can be expressed using regular expressions as [91]

$$(\varepsilon|T)(HT|SHT)^*\mathcal{C}. \tag{2.4}$$

In the language of regular expressions, $(a|b)$ denotes a choice of a or b , and $(\cdot)^*$ denotes any sequence of length $n \in \mathbb{N}$ chosen from the enclosing parentheses. The symbol ε denotes the “empty” sequence of gates and \mathcal{C} denotes any choice from the Clifford operators. That any sequence of single qubit $\mathcal{C} + T$ gates can be put in this

form is provable using the single-qubit Clifford group relations and

$$T^2 = S$$

$$\zeta T = T \zeta$$

$$TS = ST$$

$$XT = TXS\zeta^\dagger$$

where $\zeta = e^{\frac{\pi i}{4}}$ is a primitive eighth root of unity in the Clifford group.

The full impact of MA normal forms was not realized until they were rediscovered years later. In a series of subsequent work [91, 92], it was shown constructively that

$$\mathcal{C} + T = \mathcal{U}_{2 \times 2} \cap \mathcal{M}_{2 \times 2} \left(\mathbb{Z} \left[\frac{1}{\sqrt{2}}, \zeta \right] \right) \quad (2.5)$$

$$\text{Ad}_{\mathcal{C}+T} = \text{SO}(3) \cap \mathcal{M}_{3 \times 3} \left(\mathbb{Z} \left[\frac{1}{\sqrt{2}} \right] \right) \quad (2.6)$$

where $\text{Ad}_{\mathcal{C}+T}$ is the *adjoint representation* of $\mathcal{U}_{2 \times 2}$ restricted to the $\mathcal{C} + T$ operators. These two results gave a complete *number-theoretic* characterization of the single qubit $\mathcal{C} + T$ operators. Notice that $\sqrt{2}$ belongs to the subrings $\mathbb{Z}[\zeta]$ and $\mathbb{Z}[\sqrt{2}]$, and so we can associate with every such matrix in either form a least denominator exponent k .

Proof of these results relied on, for lack of a better term, *constrained Gaussian elimination*. Notice that the gate H performs something akin to row-addition / subtraction when left-multiplied to a unitary U (albeit up to a factor of $\frac{1}{\sqrt{2}}$). Like-

wise, in the adjoint representation, Ad_T serves a similar role. In both forms, the remaining generators are generalized permutation matrices. By alternating these permutations with either H or Ad_T , the lde of the matrix can be reduced one-by-one until the identity operator is achieved. The gate sequence used to do this reduction can then be inverted, yielding a circuit which is equivalent to the initial operator. The lde in both representations encodes the final T -count (exactly so in the adjoint representation case), and the resulting circuits are always in MA normal form.

This technique of using constrained Gaussian elimination based on the underlying ring of matrices to which the circuits over a gate set belong proved very powerful. A variety of other single-qubit gate sets were shown to have normal forms [93–95], and in every case a constructive exact synthesis algorithm was produced which provides the shortest gate sequence over these gate sets. These exact synthesis algorithms form the basis of many of the state-of-the-art compiling programs in use today. Unfortunately, extending these techniques to find optimal normal forms for fault-tolerant qudit or multi-qubit gate sets proved difficult.

2.4 Multi-Qubit Exact Synthesis – Giles Selinger Algorithm

One of the key factors in building a quantum compiler is establishing whether a circuit is exactly expressible over a gate set. The reason for this is two-fold: we would like to know when it is possible to run an exact synthesis algorithm on a circuit and also in which family of unitaries to look for an ε approximation when we

cannot. In Section 2.3, we described such a characterization for single-qubit $\mathcal{C} + T$. Shortly after this work, it was established [96] that

$$\mathcal{C} + T = \mathcal{U}_{2^n \times 2^n} \cap \mathcal{M}_{2^n \times 2^n} \left(\mathbb{Z} \left[\frac{1}{\sqrt{2}}, \zeta \right] \right) \quad (2.7)$$

for *multi qubit* circuits as well, demonstrating equality in Eq. (1.3) for qubits.

The so-called Giles-Selinger algorithm gives a constructive proof of this fact. In principle, it also constitutes a multi-qubit exact synthesis algorithm, though the gate count scaling is doubly-exponential in the lde of the original matrix and is therefore not close to optimal. The algorithm is remarkably simple, and consists of three observations, the first of which is Eq. (1.3). Secondly, it can be observed that given fully-controlled versions of H and T along with permutation matrices (generated by $\{X, CX, CCX\} \in \mathcal{C} + T$), column-by-column denominator exponent reduction can be performed in the same manner as the single qubit case. Finally, explicit constructions are given for fully-controlled H and T operators using the $\mathcal{C} + T$ gate set, completing the proof.

2.5 Approximate Synthesis in Single- and Multi-Qubit Circuits

Finally, given solutions to both the exact synthesis problem and a gate set characterization, researchers began to search for algorithms to meet theoretical bounds on efficiently universal gate sets. A series of papers established asymptotically optimal synthesis algorithms, at first via the use of ancilla [97, 98]. Subsequent results showed how to do this without the use of ancilla in the case of Pauli-rotations [99].

Finally, a result by Ross and Selinger [100] established the optimal approximate synthesis solution for Z -rotations in the presence of a factoring oracle. Without such an oracle, the second-to-optimal solution can be found efficiently. These rotations can then be used to construct any operator through the use of Euler angles.

The Ross-Selinger algorithm works roughly as follows. Given some target special unitary operator of the form

$$U = \begin{bmatrix} e^{\frac{i\phi}{2}} & 0 \\ 0 & e^{-\frac{i\phi}{2}} \end{bmatrix},$$

the goal is to find an approximation

$$\tilde{U} = \begin{bmatrix} u & -t^* \\ t & u^* \end{bmatrix}$$

within ε of U by Frobenius norm. The first step is to search for candidate solutions for u . To find the optimal solution, we need to enumerate the

$$u = \frac{u'}{\sqrt{2}^k}$$

by increasing lde k with $u' \in \mathbb{Z}[\zeta]$ such that $u \approx e^{\frac{i\phi}{2}}$ (roughly speaking, this equates to searching for lattice points in a long and skinny rectangle oriented at an angle). While a scheme to perform this enumeration efficiently was devised in their work, the authors later realized that earlier work by Lenstra, Lenstra, and Lovász, dubbed the LLL lattice basis reduction algorithm, could be used in place of their scheme [101].

Once such a set has been enumerated, one can check if there exists some $t \in \mathbb{Z} \left[\frac{1}{\sqrt{2}}, \zeta \right]$ such that $|u|^2 + |t|^2 = 1$. Though this in general requires factoring, it can be shown that even without a fast factoring algorithm, solutions can be found efficiently that are close to optimal.

That there exists such an optimal algorithm in the case of single-qubit Euler angle rotations is remarkable – it is known that solving a related problem for general single-qubit unitaries over finite fields is hard enough to form the basis of a cryptographic protocol [102]. As a generalized Euler angle decomposition of $SU(n)$ exists [103], this approximation scheme can be used in conjunction with permutations generated by $\{X, CX, CCX\}$ to produce *any* unitary approximation. Other schemes for approximating multi-qubit operators requiring ancillas were developed using Householder reflections expressible as $\mathcal{C} + T$ circuits [104]. In this case, the output circuits have

$$\mathcal{O} \left(4^n n \left(\log \left(\frac{1}{\varepsilon} \right) + n \right) \right) \quad (2.8)$$

gates, which is asymptotically optimal for fixed qubit number.

2.6 New Directions and Thesis Outline

This briefly outlined work has been foundational in kicking off a new revolution in quantum compiling. The influx of number-theoretic ideas and techniques have been a boon to the field. However, many open questions remain.

A wealth of research suggests that fault-tolerant qudit quantum computing

schemes could handily reduce gate-counts when compared to their qubit counterparts [83, 105]. However, no basic exact compiling algorithms for fault tolerant qudit schemes previously existed, let alone approximation schemes. In Chapter 3, we address this issue by introducing a unique MA normal form analogue for qutrits while also highlighting the existence of such normal forms in higher dimensions.

Many recent efforts in quantum compiling have involved trying to tackle multi-qubit exact synthesis [96, 104]. Due to the complexity of this problem, much attention has been paid to *restricted* $\mathcal{C} + T$ circuits [67, 106–111]. While work with these simpler gate sets has proven fruitful, their restriction often comes at the cost of a loss of universality. In Chapter 4 we attempt to rectify this problem by completely characterizing four universal subsets of $\mathcal{C} + T$. We find gate sets that exactly correspond to the appropriately-sized unitaries over the rings \mathbb{D} , $\mathbb{D}[\sqrt{2}]$, $\mathbb{D}[\sqrt{-2}]$, and $\mathbb{D}[i]$ and show that a single ancilla is always sufficient to exactly synthesize such operators.

Building off this work in Chapter 5, we develop the first non-Clifford optimal multi-qubit exact synthesis algorithm using the $\mathcal{C} + CS$ gate set on two qubits. Rather than relying on the adjoint representation as in past work [91], we instead use the accidental isomorphism of $SU(4) \cong Spin(6)$ to find a convenient algorithm for synthesizing these circuits. Afterwards, we show that the form these circuits take is relatively more complex than that of MA normal forms due to group relations that require circuit optimizations over syllables which are otherwise not locally-separated.

Finally, in Chapter 6 we describe an approximation scheme for Pauli rotations in the two-qubit case using the $\mathcal{C} + CS$ gate set, in analogue with past work [100].

In addition, we provide some basic software to perform both exact and inexact synthesis for two-qubit circuits using this gate set.

Chapter 3: Matsumoto-Amano Normal Forms of Dimension Three ¹

3.1 Introduction

Over the past five years, the aforementioned algebraic and number-theoretic methods [93–95, 97–100, 113, 114] have rejuvenated the field of quantum compiling. Until very recently and despite the existence of these successful methods in qubit quantum compiling, the Solovay-Kitaev algorithm [86, 115] remained the standard method in higher dimensions. However, advances in anyonic quantum computation [116], the discovery of protocols for higher-dimensional magic state distillation [83], and the emergence of novel means of error correction using higher dimensional Hilbert spaces [117–120] have drawn the attention of the community to qutrit quantum compiling [116, 121–123].

The single-qutrit Clifford+ T gate set, sometimes also referred to as the *supermetaplectic* gate set [122], consists of the single-qutrit Clifford gates together with a three-dimensional analogue of the single-qubit T gate which are defined in Section 1.4.2. The qutrit version of the T gate was independently introduced in [82] and [83] and shares many properties with its qubit counterpart. Most importantly,

¹This work is a slightly modified version of [112] with an additional section on normal forms in prime dimensions ≥ 5 .

it can be fault-tolerantly implemented via magic state distillation [83]. For the duration of Chapter 3, we let $\omega = e^{\frac{2\pi i}{3}}$ and $\zeta = e^{\frac{2\pi i}{9}}$. Echoing Section 1.4.2, the generators of $\mathcal{C} + T$ are then

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \omega \end{bmatrix}, \quad H = -\frac{1}{i\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix}, \quad \text{and} \quad T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \zeta & 0 \\ 0 & 0 & \zeta^8 \end{bmatrix}.$$

The Clifford group likewise contains the Pauli group generators

$$X = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \text{and} \quad Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \omega & 0 \\ 0 & 0 & \omega^2 \end{bmatrix}.$$

In this chapter, we introduce canonical forms for single-qutrit Clifford+ T circuits inspired by prior work on single-qubit Clifford+ T circuits [90–92, 124]. We prove that every single-qutrit Clifford+ T operator admits a canonical form and give a linear-time algorithm to convert an arbitrary Clifford+ T circuit to canonical form. Finally, we show that distinct canonical forms represent distinct operators. We establish the uniqueness of canonical form representation by giving an algorithm which inputs the matrix of a Clifford+ T operator and deterministically constructs a canonical form circuit for it. This uniqueness property implies that our canonical forms are T -optimal: among all the single-qutrit Clifford+ T circuits implementing a given operator our canonical form uses the least number of T gates. This T -optimality is

desirable in light of the high cost associated with fault-tolerantly implementing T gates.

The organization of Chapter 3 is as follows. First, we define canonical forms and prove that every Clifford+ T operator admits a canonical form in Section 3.2. Next, we prove the uniqueness of canonical form representations in Section 3.3. Finally, we show in Section 3.4 that a similar normal form can be established in higher prime dimension, though proof of uniqueness is not shown. We then conclude in Section 3.5.

3.2 Canonical forms

We define a three-dimensional analogue of the MA normal forms introduced in [90] for single qubit Clifford+ T circuits and we prove that every single-qutrit Clifford+ T operator can be represented by such a canonical form. Our presentation follows [91].

Definition 3.2.1. A *canonical form* is a Clifford+ T circuit of the form

$$(\varepsilon | T | H^2T)(HT | H^3T | SHT | SH^3T | S^2HT | S^2H^3T)*\mathcal{C}. \quad (3.1)$$

Here, following [91], we again use the language of regular expressions to define canonical forms. In Eq. (3.1), ε denotes the empty word and \mathcal{C} denotes any of one of the 648 Clifford operators. Eq. (3.1) therefore states that a canonical form (read from left to right) consists of an optional occurrence of T or H^2T , any number of

syllables chosen from the set $\{HT, H^3T, SHT, SH^3T, S^2HT, S^2H^3T\}$, and a final Clifford operator.

Definition 3.2.2. Let \mathcal{S} be the 81-element subgroup of \mathcal{C} group generated by S and X , \mathcal{M} be the two element subgroup of \mathcal{C} generated by H^2 , and \mathcal{L} and \mathcal{L}' be the following sets of Clifford operators:

$$\mathcal{L} = \{\mathbb{1}, H, SH, S^2H\} \quad \text{and} \quad \mathcal{L}' = \{H, SH, S^2H\}.$$

Note that $\omega \in \mathcal{S}$ and that $\mathcal{M}\mathcal{S} = \mathcal{S}\mathcal{M}$ is the 162-element subgroup of \mathcal{C} which consists of generalized permutation matrices. Note moreover that the syllables used in Definition 3.2.1 are the elements of $\mathcal{L}\mathcal{M}T$.

Lemma 3.2.3. *The following relations hold.*

$$\mathcal{C} = \mathcal{L}\mathcal{M}\mathcal{S} \tag{3.2}$$

$$\mathcal{S}T = T\mathcal{S} \tag{3.3}$$

$$TT = H^2TH^2Z \subseteq \mathcal{M}T\mathcal{M}\mathcal{S} \tag{3.4}$$

$$TH^2T = H^2 \subseteq \mathcal{M}. \tag{3.5}$$

Proof. Eq. (3.2) follows from the fact that the Clifford operators are a disjoint union of the cosets of \mathcal{S} which are \mathcal{S} , $H^2\mathcal{S}$, $H\mathcal{S}$, $H^3\mathcal{S}$, $SH\mathcal{S}$, $SH^3\mathcal{S}$, $S^2H\mathcal{S}$, and $S^2H^3\mathcal{S}$. Eq. (3.3) follows from the three commutation relations $ST = TS$, $XT = T SX$, and $\omega T = T\omega$. Finally, Eqs. (3.4) and (3.5) follow from direct computation.

□

Lemma 3.2.4. *An integer power of T is either a Pauli operator or is Clifford equivalent to T . That is, for $a \in \mathbb{Z}$ we have*

$$T^a = \begin{cases} Z^{\frac{a}{3}} & a = 0 \pmod{3} \\ TZ^{\frac{a-1}{3}} & a = 1 \pmod{3} \\ H^2TH^2Z^{\frac{a+1}{3}} & a = 2 \pmod{3} \end{cases}$$

Proof. This is a consequence of Eqs. (3.4) and (3.5) and the relations $T^9 = \mathbb{1}$ and $TZ = ZT$. □

Definition 3.2.5. The *Clifford-prefix* of a syllable $M = M'T \in \mathcal{LMT}$ is the Clifford operator $M' \in \mathcal{LM}$ that precedes T .

Proposition 3.2.6. *Every Clifford+ T operator can be represented by a circuit in canonical form.*

Proof. We first show that if U is a canonical form and A is one of the generators of the Clifford+ T group then UA admits a canonical form. In the case where A is a Clifford operator there is nothing to show so we can assume that A is a T gate. We now proceed by induction on the T -count of U .

- If U has T -count 0 then by Eqs. (3.1) to (3.3) we have $UT \in \mathcal{LMTS} = \mathcal{LMTS} \subseteq \mathcal{LMTS}$ so that UA has a canonical form of T -count 1.
- If U has T -count 1 then by Eqs. (3.1) and (3.2) we know $U \in \mathcal{LMTLS}$.

Using Eqs. (3.3) to (3.5) we get

$$\begin{aligned}
UT &\in \mathcal{L} \mathcal{M} T \mathcal{L} \mathcal{M} \mathcal{S} T \\
&= \mathcal{L} \mathcal{M} T \mathcal{L} \mathcal{M} T \mathcal{S} \\
&= \mathcal{L} \mathcal{M} T \mathcal{L}' \mathcal{M} T \mathcal{S} \cup \mathcal{L} \mathcal{M} T \mathcal{M} T \mathcal{S} \\
&= \mathcal{L} \mathcal{M} T \mathcal{L}' \mathcal{M} T \mathcal{S} \cup \mathcal{L} \mathcal{M} T T \mathcal{S} \cup \mathcal{L} \mathcal{M} T H^2 T \mathcal{S} \\
&= \mathcal{L} \mathcal{M} T \mathcal{L}' \mathcal{M} T \mathcal{S} \cup \mathcal{L} \mathcal{M} H^2 T H^2 Z \mathcal{S} \cup \mathcal{L} \mathcal{M} H^2 \mathcal{S} \\
&= \mathcal{L} \mathcal{M} T \mathcal{L}' \mathcal{M} T \mathcal{S} \cup \mathcal{L} \mathcal{M} T H^2 \mathcal{S} \cup \mathcal{L} \mathcal{M} \mathcal{S} \\
&\subseteq \mathcal{L} \mathcal{M} T \mathcal{L}' \mathcal{M} T \mathcal{C} \cup \mathcal{L} \mathcal{M} T \mathcal{C} \cup \mathcal{C}.
\end{aligned}$$

It follows that UA has a canonical form of T -count 0, 1, or 2.

- If U has T -count $\ell > 1$ then we can use Eqs. (3.1) and (3.2) again to write U as an element of $\mathcal{L} \mathcal{M} T (\mathcal{L}' \mathcal{M} T)^{\ell-2} \mathcal{L}' \mathcal{M} T \mathcal{L} \mathcal{M} \mathcal{S}$. We can now reason as in the previous case to show that

$$UT \in \mathcal{L} \mathcal{M} T (\mathcal{L}' \mathcal{M} T)^\ell \mathcal{C} \cup \mathcal{L} \mathcal{M} T (\mathcal{L}' \mathcal{M} T)^{\ell-1} \mathcal{C} \cup \mathcal{L} \mathcal{M} T (\mathcal{L}' \mathcal{M} T)^{\ell-2} \mathcal{C}.$$

And it follows that UA has a canonical form of T -count $\ell - 1$, ℓ , or $\ell + 1$.

Now let U be a canonical form and A be either a Clifford operator or a power of T . Assume moreover that the T -count of U is ℓ and the T -count of A is k . Then the above argument, together with Lemma 3.2.4, imply that UA has a canonical form of T -count at most $1 + \ell$.

To complete the proof, let V be a Clifford+ T operator. Then $V = A_1 \dots A_n$ where every A_i either a Clifford operator or a power of T . Starting with the identity operator, one may then proceed by rightward induction on n to put V in canonical form. □

Corollary 3.2.7. *There exists an algorithm to rewrite any Clifford+ T circuit into canonical form. The algorithm runs in time linear in the gate-count of the input circuit.*

Proof. This is a consequence of the constructive proof of Proposition 3.2.6. Indeed, a constant number of operations are needed to update at most six of the rightmost operators of a canonical form upon right-multiplication by a Clifford+ T operator. Any Clifford+ T operator of length n can therefore be put in canonical form in $\mathcal{O}(n)$ steps. To see this algorithm in action, consider visiting Xiaoning Bian’s homepage [126]. □

Remark 3.2.8. Suppose that V is a Clifford+ T circuit for some operator U and that V' is the canonical form for U obtained by applying Corollary 3.2.7 to V . If ℓ is the T -count of V and ℓ' is the T -count of V' then $\ell' \leq \ell$. This follows from the fact that the algorithm of Corollary 3.2.7 never increases the T -count of a circuit.

We close this section by discussing an alternative canonical form for single-qutrit Clifford+ T circuits. The canonical form of Definition 3.2.1 is inspired by the one introduced by Matsumoto and Amano in [90] for single-qubit Clifford+ T circuits. In [124], Forest and others introduced a channel representation for single qubit Clifford-cyclotomic circuits. When restricted to single Clifford+ T circuits

their channel representation can be interpreted as a sequence of $\pi/4$ rotations about the x -, y -, or z -axes of the Bloch Sphere (followed by a single Clifford operator). This sequence is subject to the condition that consecutive rotations revolve around different axes. Below, we define an analogue for single-qutrit Clifford+ T circuits.

Definition 3.2.9. Let P be a Pauli operator and let λ_0 , λ_1 , and λ_2 be the following real numbers:

$$\lambda_0 := \frac{1 + \zeta + \zeta^8}{3}, \quad \lambda_1 := \frac{1 + \zeta^2 + \zeta^7}{3}, \quad \text{and} \quad \lambda_2 := \frac{1 + \zeta^4 + \zeta^5}{3}.$$

Then the P -axis T gate T_P is defined as $T_P := \lambda_0 I + \lambda_1 P + \lambda_2 P^2$.

Definition 3.2.10. A *channel form* is a single-qutrit Clifford+ T circuit of the form

$$T_{P_1^{n_1}} T_{P_2^{n_2}} \dots T_{P_\ell^{n_\ell}} C \tag{3.6}$$

where $\ell \in \mathbb{N}$, $P_i \in \{Z, X, XZ, XZ^2\}$, $n_i \in \mathbb{Z}_3 \setminus \{1\}$, $P_i \neq P_{i+1}$ and $C \in \mathcal{C}$.

It can be shown that channel forms are in bijective correspondence with the canonical forms of Definition 3.2.1 so that every Clifford+ T operator admits a channel form. Moreover, this correspondence preserves the T -count.

3.3 Uniqueness of canonical forms

Proposition 3.2.6 showed that every Clifford+ T operator can be represented by a circuit in canonical form. In this section, we show that this representation is

unique in the sense that if M and N are different canonical forms that M and N represent different Clifford+ T operators.

3.3.1 Algebraic preliminaries

Definition 3.3.1. Let $\alpha = \sin(2\pi/9)$ and $\gamma = 1 - \zeta$. We will employ six extensions of \mathbb{Z} in our analysis.

- $\mathbb{Z}[\omega] = \{a + b\omega \mid a, b \in \mathbb{Z}\}$
- $\mathbb{Z}[\zeta] = \{a + b\zeta + c\zeta^2 + d\zeta^3 + e\zeta^4 + f\zeta^5 \mid a, b, c, d, e, f \in \mathbb{Z}\}$
- $\mathbb{Z}[\frac{1}{\gamma}] = \left\{ \frac{A}{\gamma^k} \mid A \in \mathbb{Z}[\zeta], k \in \mathbb{N} \right\}$
- $\mathbb{D} = \mathbb{Z}[\frac{1}{2}] = \left\{ \frac{a}{2^k} \mid a \in \mathbb{Z}, k \in \mathbb{N} \right\}$
- $\mathbb{Z}[\alpha] = \{A + B\alpha + C\alpha^2 + D\alpha^3 + E\alpha^4 + F\alpha^5 \mid A, B, C, D, E, F \in \mathbb{D}\}$
- $\mathbb{D}[\frac{1}{\alpha}] = \mathbb{Z}[\frac{1}{2}, \frac{1}{\alpha}] = \left\{ \frac{A}{\alpha^k} \mid A \in \mathbb{Z}[\alpha], k \in \mathbb{N} \right\}$

Per our discussion of rings in Section 1.4.1, the ring $\mathbb{Z}[\omega]$ is known as the ring of *cyclotomic integers of degree 3* while the ring $\mathbb{Z}[\zeta]$ is known as the ring of *cyclotomic integers of degree 9*. The ring \mathbb{D} is the ring of *dyadic fractions*.

Remark 3.3.2. We record here some important relations involving α which will be useful in what follows.

- $\frac{\sqrt{3}}{2} = 3\alpha - 4\alpha^3$
- $\sqrt{3} = 6\alpha - 8\alpha^3$

- $3 = 36\alpha^2 - 96\alpha^4 + 64\alpha^6$
- $\alpha^6 = \frac{3}{2^6} - \frac{9}{2^4}\alpha^2 + \frac{3}{2}\alpha^4$
- $\alpha = \frac{3}{2^6} - \frac{9}{2^4}\alpha^2 + \frac{3}{2}\alpha^4$
- $\frac{1}{2} = -1 + 18\alpha^2 - 48\alpha^4 + 32\alpha^6$

In particular, the fifth relation implies that $\mathbb{Z}[\alpha]$ is a subring of $\mathbb{D}[\frac{1}{\alpha}]$.

Remark 3.3.3. The entries of any Clifford+ T operator belong to the ring $\mathbb{Z}[\frac{1}{\gamma}]$. To see this, note that it holds for the generators S , H , and T , since

$$-\frac{1}{i\sqrt{3}} = \frac{1}{\gamma^3}(1 - \zeta + \zeta^2 + \zeta^3 + 2\zeta^4 - 2\zeta^5) \in \mathbb{Z}[\frac{1}{\gamma}].$$

Definition 3.3.4 (Residue). The *residue map* ρ is the ring homomorphism $\rho : \mathbb{Z}[\alpha] \rightarrow \mathbb{Z}_3$ defined by $\rho(q) = q \pmod{\alpha}$.

It follows from Remark 3.3.2 that $\rho(\sqrt{3}) = 0$, $\rho(3) = 0$, and $\rho(\frac{1}{2}) = 2$. These equalities give an intuition of how one might compute $\rho(q)$ given $q \in \mathbb{Z}[\alpha]$. First write q as a sum $q = c_0\alpha^0 + \dots + c_5\alpha^5$ with each $c_j \in \mathbb{D}$ such that $c_j = \frac{a_j}{2^{b_j}}$ where $a_j \in \mathbb{Z}$ and $b_j \in \mathbb{N}$. Then $\rho(q) = \rho(c_0) = \rho(a_0/2^{b_0}) = (2^{b_0}a_0) \pmod{3}$.

Definition 3.3.5 (Denominator Exponent). For every $q \in \mathbb{D}[\frac{1}{\alpha}]$, there exists some $k \geq 0$ such that $\alpha^k q \in \mathbb{Z}[\alpha]$. Then we identify k as a denominator exponent, with the least such k the lde. This notion extends to vectors and matrices as defined in Section 1.4.1.

Definition 3.3.6 (*k*-Residue). Let $q \in \mathbb{D}[\frac{1}{\alpha}]$ and let k be a denominator exponent of q . Then the *k*-residue of q , $\rho_k(q)$ and is defined as $\rho_k(q) = \rho(\alpha^k q) \in \mathbb{Z}_3$. The *k*-residue of a vector or matrix is defined component-wise.

Lemma 3.3.7. *Let $q \in \mathbb{D}[\frac{1}{\alpha}]$ and let $k \in \mathbb{N}$ be a denominator exponent of q . Then k is the lde of q if and only if $\rho_k(q) \neq 0 \pmod{3}$ or $k = 0$.*

Proof. If $k = 0$ then k is a denominator exponent of q if and only if it is the lde of q . Suppose some $k > 0$ is not the lde of q but is a denominator exponent. Since k is a denominator exponent of q we can write q as

$$q = \frac{1}{\alpha^k} \sum_{j=0}^5 c_j \alpha^j$$

with each $c_j \in \mathbb{D}$. Note that $\rho_k(q) = \rho(c_0)$. Since k is not the lde of q , we can rewrite q as

$$q = \frac{1}{\alpha^{k-1}} \left[\alpha^{-1} c_0 + \sum_{j=0}^4 c_{j+1} \alpha^j \right]$$

where it must be the case that $\alpha^{-1} c_0 \in \mathbb{Z}[\alpha]$. If $\rho(c_0) = 0 \pmod{3}$, then we can write $c_0 = 3c'_0$ for some $c'_0 \in \mathbb{D}$ and have

$$\alpha^{-1} c_0 = c'_0 \frac{3}{\alpha} = c'_0 (36\alpha - 96\alpha^3 + 64\alpha^5) \in \mathbb{Z}[\alpha]$$

where the term $\frac{3}{\alpha}$ is simplified using Remark 3.3.2. This proves the “only if” direction. On the other hand, if $\rho(c_0) = r \neq 0 \pmod{3}$, then we have $c_0 = r + 3c''_0$ for

some $c_0'' \in \mathbb{D}$ and $r \in \{1, 2\}$ and can write

$$\alpha^{-1}c_0 = \frac{r}{\alpha} + c_0'' \frac{3}{\alpha} = \frac{r}{\alpha} + c_0' (36\alpha - 96\alpha^3 + 64\alpha^5).$$

The second term is in $\mathbb{Z}[\alpha]$, and so $\alpha^{-1}c_0 \in \mathbb{Z}[\alpha]$ would only hold in this case if $\frac{r}{\alpha}$ is in $\mathbb{Z}[\alpha]$ as well. For $r \in \{1, 2\}$ this is not the case, leading to a contradiction and proving the “if” direction. \square

Remark 3.3.8. Let A and B be two matrices over $\mathbb{D}[\frac{1}{\alpha}]$ with lde k_A and k_B respectively. Then if $k \geq k_A$ and $k \geq k_B$ we have $\rho_k(A + B) = \rho_k(A) + \rho_k(B)$. Similarly, if $k_1 \geq k_A$, $k_2 \geq k_B$, and $k' = k_1 + k_2$, then $\rho_{k'}(AB) = \rho_{k_1}(A) \cdot \rho_{k_2}(B)$. Furthermore, if A has the property that $k_A = 0$ and that $\frac{1}{\alpha}A$ has entries in $\mathbb{Z}[\alpha]$, then

$$\rho_{k'}(AB) = \rho_0\left(\frac{1}{\alpha}A\right) \cdot \rho_{k'+1}(B)$$

for any $k' \geq k_B$. Likewise, if $k_B = 0$ and $(1/\alpha)B$ has entries in $\mathbb{Z}[\alpha]$, then $\rho_{k'}(AB) = \rho_{k'+1}(A) \cdot \rho_0((1/\alpha)B)$. Finally, if $\ell > k_A$ then $\rho_\ell(A) = 0_{m \times n}$ by Lemma 3.3.7.

3.3.2 The adjoint representation

We will use an alternative representation for Clifford+ T operators in analogue with past work. Let P be a Pauli operator and define the operators P_+ and P_- as

$$P_+ := \frac{1}{\sqrt{\text{Tr}[(P + P^\dagger)^2]}}(P + P^\dagger) \quad \text{and} \quad P_- := \frac{i}{\sqrt{-\text{Tr}[(P - P^\dagger)^2]}}(P - P^\dagger).$$

Now consider the sets $\mathcal{Q} = \{\frac{1}{\sqrt{3}}, Z_+, X_+, (XZ)_+, (XZ^2)_+, Z_-, X_-, (XZ)_-, (XZ^2)_-\}$ and $\mathcal{Q}' = \mathcal{Q} \setminus \{\frac{1}{\sqrt{3}}\}$ where $\frac{1}{\sqrt{3}} = \frac{1}{\sqrt{3}}\mathbb{1}$. These sets have a number of properties:

- The set \mathcal{Q} is a complete orthonormal basis for the set $\mathcal{M}_3(\mathbb{C})$ of 3×3 complex matrices with respect to the inner product $\langle A, B \rangle = \langle B, A \rangle^* = \text{Tr}(AB^\dagger)$. That is, if $Q_i, Q_j \in \mathcal{Q}$ then $\langle Q_i, Q_j \rangle = \delta_{i,j}$.
- Every $Q \in \mathcal{Q}$ is Hermitian
- Every $Q \in \mathcal{Q}'$ is traceless.
- Every $Q \in \mathcal{Q}'$ is of one of two forms: either the matrix Q is of the type P_+ and is such that P is a Pauli matrix with $P_+ = \frac{1}{\sqrt{6}}(P + P^2)$ or it is of the type P_- and is such that P is a Pauli matrix with $P_- = \frac{i}{\sqrt{6}}(P - P^2)$.

Note that the set \mathcal{Q} , much like the set of Gell-Mann matrices, does *not* form a group under multiplication.

Because every element of \mathcal{Q} is Hermitian, any unit trace 3×3 Hermitian matrix ρ may be written as

$$\rho = \frac{1}{3}\mathbb{1} + \frac{1}{\sqrt{3}} \sum_{Q_i \in \mathcal{Q}'} c_i Q_i \quad (3.7)$$

with $c_i \in \mathbb{R}$. Conjugation by a unitary operator U preserves the trace of ρ . The action of U will therefore send each c_i to some $c'_i \in \mathbb{R}$, since $\rho' = U\rho U^\dagger$ is still Hermitian. This encourages us to define an adjoint representation for unitary operators using \mathcal{Q}' .

Definition 3.3.9. Let $\hat{\rho}$ and $\hat{\rho}'$ be the eight-component vectors composed of the real c_i and c'_i as in Eq. (3.7) for the 3×3 Hermitian matrices ρ and ρ' . Then the adjoint representation of a unitary operator U , denoted \hat{U} , is defined by

$$\hat{U}\hat{\rho} = \hat{\rho}' \iff U\rho U^\dagger = \rho'.$$

Remark 3.3.10. Composition of operators in the adjoint basis is equivalent to matrix multiplication, which can be seen as follows. If

$$\hat{U}_1\hat{\rho} = \hat{\rho}' \iff U_1\rho U_1^\dagger = \rho',$$

then we have

$$\hat{U}_2\hat{\rho}' = (\hat{U}_2\hat{U}_1)\hat{\rho} \iff U_2\rho'U_2^\dagger = U_2U_1\rho U_1^\dagger U_2^\dagger = (U_2U_1)\rho(U_2U_1)^\dagger$$

To maintain consistency, we impose an ordering for the c_i consistent the ordering of the sequence

$$(\mathcal{Q}') = (Z_+, X_+, (XZ)_+, (XZ^2)_+, Z_-, X_-, (XZ)_-, (XZ^2)_-).$$

This ordering allows us to write an explicit definition for \hat{U} using our inner product.

Lemma 3.3.11. *Let $Q_i \in \mathcal{Q}'$ with Q_i the i th element of (\mathcal{Q}') . The adjoint repre-*

resentation \hat{U} of a Unitary operator U may be calculated

$$\hat{U}_{i,j} = \langle Q_i, UQ_jU^\dagger \rangle = \text{Tr} [Q_iUQ_jU^\dagger]$$

Proof. This follows directly from the orthonormality and Hermiticity of the Q_i . \square

Lemma 3.3.12. *Every adjoint representation \hat{U} of a unitary operator U is a real, special orthogonal matrix.*

Proof. Every element of $\hat{U}_{i,j}$ is real due to the properties of our inner product and the cyclic properties of the trace:

$$\begin{aligned} \hat{U}_{i,j}^* &= \langle Q_i, UQ_jU^\dagger \rangle^* = \langle UQ_jU^\dagger, Q_i \rangle = \text{Tr} [UQ_jU^\dagger Q_i] \\ &= \text{Tr} [Q_iUQ_jU^\dagger] = \langle Q_i, UQ_jU^\dagger \rangle = \hat{U}_{i,j} \end{aligned}$$

To show that every \hat{U} is orthogonal, consider the inverse of U . As U is unitary, the inverse of U is U^\dagger and thus the inverse of \hat{U} must be (\hat{U}^\dagger) . This gives

$$\begin{aligned} \hat{U}_{i,j}^{-1} &= (\hat{U}^\dagger)_{i,j} = \langle Q_i, U^\dagger Q_jU \rangle = \text{Tr} [Q_iU^\dagger Q_jU] \\ &= \text{Tr} [Q_jUQ_iU^\dagger] = \langle Q_j, UQ_iU^\dagger \rangle = \hat{U}_{j,i} = \hat{U}_{i,j}^\top \end{aligned}$$

It just remains to show that $\det \hat{U} = 1$. Nominally, this is true because the group should be connected. For a simple proof, consider again the matrix U . As it is unitary, we know there is a Hermitian operator A such that $U = \exp(-iA)$. We

begin by examining the Trotter decomposition of U :

$$U = \lim_{N \rightarrow \infty} \left[\exp \left(\frac{-i}{N} A \right) \right]^N .$$

This in turn implies that we have

$$\hat{U} = \lim_{N \rightarrow \infty} \left[\hat{V} \left(\frac{A}{N} \right) \right]^N$$

where we have defined $\hat{V}(M)$ as the adjoint representation for the $\exp(-iM)$. From this form, we calculate $\det \hat{U}$:

$$\begin{aligned} \det \hat{U} &= \lim_{N \rightarrow \infty} \left[\det \left(\hat{V} \left(\frac{A}{N} \right) \right) \right]^N \\ &= \lim_{N \rightarrow \infty} \exp \left[N \operatorname{Tr} \left[\log \left(\hat{V} \left(\frac{A}{N} \right) \right) \right] \right] \end{aligned}$$

To lowest orders in $\frac{1}{N}$, we compute $\hat{V} \left(\frac{A}{N} \right)$ using the Hadamard lemma:

$$\begin{aligned} \left(\hat{V} \left(\frac{A}{N} \right) \right)_{j,k} &= \operatorname{Tr} \left[Q_j \exp \left(\frac{-i}{N} A \right) Q_k \exp \left(\frac{i}{N} A \right) \right] \\ &= \operatorname{Tr} [Q_j Q_k] - \frac{i}{N} \operatorname{Tr} [Q_j [A, Q_k]] + \mathcal{O} \left(\left(\frac{1}{N} \right)^2 \right) \end{aligned}$$

The first term here is simply the elemental form of the identity $\delta_{j,k}$ and the second term makes use of the standard definition of an algebra commutator, $[A, B] =$

$AB - BA$. Upon calculating the trace of the matrix logarithm of $\hat{V} \left(\frac{A}{N} \right)$ we have

$$\text{Tr} \left[\log \left(\hat{V} \left(\frac{A}{N} \right) \right) \right] = \frac{-i}{N} \sum_j \text{Tr} [Q_j [A, Q_j]] + \mathcal{O} \left(\left(\frac{1}{N} \right)^2 \right) = \mathcal{O} \left(\left(\frac{1}{N} \right)^2 \right)$$

with the leading term disappearing due to the cyclic properties of the trace. Finally, this yields the desired result:

$$\det \hat{U} = \lim_{N \rightarrow \infty} \exp \left[N \cdot \mathcal{O} \left(\left(\frac{1}{N} \right)^2 \right) \right] = \lim_{N \rightarrow \infty} 1 + \mathcal{O} \left(\frac{1}{N} \right) = 1$$

□

Definition 3.3.13 (Quadrants). Let \hat{U} be the adjoint representation of a unitary operator U . Let $\mathcal{Q}'_+ = \{Z_+, X_+, (XZ)_+, (XZ^2)_+\}$, $\mathcal{Q}'_- = \{Z_-, X_-, (XZ)_-, (XZ^2)_-\}$ be two sets with associated sequences (\mathcal{Q}'_+) and (\mathcal{Q}'_-) with orderings as specified. Let $Q_{i,+} \in \mathcal{Q}'_+$ be the i th element of (\mathcal{Q}'_+) and $Q_{i,-} \in \mathcal{Q}'_-$ be the i th element of (\mathcal{Q}'_-) . Then we define the four 4×4 *quadrants* of \hat{U} (starting from the upper-left quadrant and going counter-clockwise) as follows:

$$\left(\hat{U}_{++} \right)_{i,j} = \langle Q_{i,+}, U Q_{j,+} U^\dagger \rangle$$

$$\left(\hat{U}_{-+} \right)_{i,j} = \langle Q_{i,-}, U Q_{j,+} U^\dagger \rangle$$

$$\left(\hat{U}_{--} \right)_{i,j} = \langle Q_{i,-}, U Q_{j,-} U^\dagger \rangle$$

$$\left(\hat{U}_{+-} \right)_{i,j} = \langle Q_{i,+}, U Q_{j,-} U^\dagger \rangle$$

Lemma 3.3.14. *Every adjoint representation \hat{D} of a diagonal unitary operator D*

is symplectic.

Proof. As D is diagonal and unitary, we may write it as

$$D = \begin{pmatrix} e^{i\beta_1} & 0 & 0 \\ 0 & e^{i\beta_2} & 0 \\ 0 & 0 & e^{i\beta_3} \end{pmatrix}$$

Direct computation of \hat{D} yields

$$\hat{D} = \left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & d_1 & d_2 & d_3 & 0 & d_4 & d_5 & d_6 \\ 0 & d_3 & d_1 & d_2 & 0 & d_6 & d_4 & d_5 \\ 0 & d_2 & d_3 & d_1 & 0 & d_5 & d_6 & d_4 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -d_4 & -d_5 & -d_6 & 0 & d_1 & d_2 & d_3 \\ 0 & -d_6 & -d_4 & -d_5 & 0 & d_3 & d_1 & d_2 \\ 0 & -d_5 & -d_6 & -d_4 & 0 & d_2 & d_3 & d_1 \end{array} \right)$$

where we have defined

$$\begin{aligned}
d_1 &= \frac{1}{3} [\cos(\beta_1 - \beta_2) + \cos(\beta_2 - \beta_3) + \cos(\beta_3 - \beta_1)] \\
d_2 &= \frac{1}{6} [2 \cos(\beta_1 - \beta_2) - \cos(\beta_2 - \beta_3) - \cos(\beta_3 - \beta_1) \\
&\quad + \sqrt{3} (\sin(\beta_2 - \beta_3) - \sin(\beta_3 - \beta_1))] \\
d_3 &= \frac{1}{6} [2 \cos(\beta_1 - \beta_2) - \cos(\beta_2 - \beta_3) - \cos(\beta_3 - \beta_1) \\
&\quad - \sqrt{3} (\sin(\beta_2 - \beta_3) - \sin(\beta_3 - \beta_1))] \\
d_4 &= \frac{1}{3} [\sin(\beta_1 - \beta_2) + \sin(\beta_2 - \beta_3) + \sin(\beta_3 - \beta_1)] \\
d_5 &= \frac{1}{6} [2 \sin(\beta_1 - \beta_2) - \sin(\beta_2 - \beta_3) - \sin(\beta_3 - \beta_1) \\
&\quad - \sqrt{3} (\cos(\beta_2 - \beta_3) - \cos(\beta_3 - \beta_1))] \\
d_6 &= \frac{1}{6} [2 \sin(\beta_1 - \beta_2) - \sin(\beta_2 - \beta_3) - \sin(\beta_3 - \beta_1) \\
&\quad + \sqrt{3} (\cos(\beta_2 - \beta_3) - \cos(\beta_3 - \beta_1))]
\end{aligned}$$

This means we have $\hat{D}_{++} = \hat{D}_{--} = A$ and $\hat{D}_{-+} = -\hat{D}_{+-} = -B$, subject to the conditions that $AA^\top + BB^\top = A^\top A + B^\top B = \mathbb{1}$, $AB^\top = BA^\top$, and $A^\top B = B^\top A$ due to \hat{D} being special orthogonal. To see that these conditions suffice to show \hat{D} is symplectic, we must show $\hat{D}^\top \Omega \hat{D} = \Omega$ with

$$\Omega = \begin{pmatrix} 0 & \mathbb{1}_{4 \times 4} \\ -\mathbb{1}_{4 \times 4} & 0 \end{pmatrix}.$$

Using our properties for A and B , we see

$$\begin{aligned} \begin{pmatrix} A^\top & -B^\top \\ B^\top & A^\top \end{pmatrix} \begin{pmatrix} 0 & \mathbb{1} \\ -\mathbb{1} & 0 \end{pmatrix} \begin{pmatrix} A & B \\ -B & A \end{pmatrix} &= \begin{pmatrix} B^\top A - A^\top B & B^\top B + A^\top A \\ -A^\top A - B^\top B & -A^\top B + B^\top A \end{pmatrix} \\ &= \begin{pmatrix} 0 & \mathbb{1} \\ -\mathbb{1} & 0 \end{pmatrix} \end{aligned}$$

and so \hat{D} is symplectic. □

3.3.3 Uniqueness

Remark 3.3.15. The entries of every adjoint representation \hat{C} of a Clifford operator C belong to the set $\{0, \pm 1, \pm \frac{1}{2}, \pm \frac{\sqrt{3}}{2}\}$. Moreover, \hat{C}_{++} and \hat{C}_{--} are both 4×4 generalized permutation matrices with the same underlying nonzero pattern and with entries in the set $\{0, \pm 1, \pm \frac{1}{2}\}$. On the other hand, \hat{C}_{+-} and \hat{C}_{-+} are less-than-full rank 4×4 matrices with the same nonzero pattern and at most one nonzero entry per row and column, with the entries belonging to the set $\{0, \pm \frac{\sqrt{3}}{2}\}$. These properties may be verified by enumeration of the 216 distinct adjoint representations of the Clifford operators, the full set of which we denote $\hat{\mathcal{C}}$. Referencing Remark 3.3.2 we can also immediately see that the entries of every $\hat{C} \in \hat{\mathcal{C}}$ belong to the ring $\mathbb{Z}[\alpha]$.

Writing the generators of $\mathcal{C} + T$ in our adjoint representation, we have

$$\hat{S} = \left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \right), \quad \hat{H} = \left(\begin{array}{cccc|cccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 & -\frac{\sqrt{3}}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{3}}{2} & 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{array} \right),$$

$$\hat{T} = \left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_1 & t_1 & t_2 & 0 & t_3 & t_3 & t_4 \\ 0 & t_2 & t_1 & t_1 & 0 & t_4 & t_3 & t_3 \\ 0 & t_1 & t_2 & t_1 & 0 & t_3 & t_4 & t_3 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -t_3 & -t_3 & -t_4 & 0 & t_1 & t_1 & t_2 \\ 0 & -t_4 & -t_3 & -t_3 & 0 & t_2 & t_1 & t_1 \\ 0 & -t_3 & -t_4 & -t_3 & 0 & t_1 & t_2 & t_1 \end{array} \right)$$

where we have defined

$$t_1 = \frac{1}{\alpha^2} \left(-\frac{1}{2^2} + 2\alpha^2 - 2\alpha^4 \right), \quad t_2 = \frac{1}{\alpha^2} \left(\frac{1}{2^3} - \frac{3}{2}\alpha^2 + 2\alpha^4 \right),$$

$$t_3 = \frac{1}{\alpha^3} \left(-\frac{1}{2^4} + \frac{1}{2}\alpha^2 - \alpha^4 \right), \quad t_4 = \frac{1}{\alpha^3} \left(\frac{1}{2^3} - \alpha^2 + \alpha^4 \right),$$

Note that the entries of any $\mathcal{C} + T$ operator belong to the ring $\mathbb{Z}[\frac{1}{\gamma}]$. This is true due to the fact that it holds for the generators S , R , and T by using Remark 3.3.3. Furthermore, the adjoint representation of any $\mathcal{C} + T$ operator has entries in the ring $\mathbb{D}[\frac{1}{\alpha}]$, which follows from Remark 3.3.15 and the fact that the statement holds for the remaining generator, \hat{T} .

Remark 3.3.16. Consider $\rho_0(\hat{S})$ and $\rho_0(\hat{H})$. Under the residue map, these Cliffords become the following:

$$\rho_0(\hat{S}) = \left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \right) \quad \text{and} \quad \rho_0(\hat{H}) = \left(\begin{array}{cccc|cccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{array} \right)$$

Thus, *any* adjoint representation \hat{C} of a Clifford operator is such that the following hold:

- The lde of \hat{C} is zero
- $\rho_0(\hat{C})$ is a generalized permutation matrix with entries in \mathbb{Z}_3
- $\rho_0(\hat{C}_{++})$ is a true permutation matrix
- $\rho_0(\hat{C}_{--})$ is a generalized permutation matrix with entries in \mathbb{Z}_3
- $\rho_0(\hat{C}_{+-}) = \rho_0(\hat{C}_{-+}) = 0_{4 \times 4}$
- By Remarks 3.3.2 and 3.3.15 we have $\rho_0\left(\frac{1}{\alpha}\hat{C}_{+-}\right) = \rho_0\left(\frac{1}{\alpha}\hat{C}_{-+}\right) = 0_{4 \times 4}$

In particular, we explicitly write out the nonzero quadrants of every Clifford in the set \mathcal{LM} :

$$\rho_0(\hat{\mathbb{1}}_{++}) = \rho_0(\hat{H}_{++}^2) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\rho_0(\hat{H}_{++}) = \rho_0(\hat{H}_{++}^3) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\rho_0(\hat{S}\hat{H}_{++}) = \rho_0(\hat{S}\hat{H}_{++}^3) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{aligned}
\rho_0(\hat{S}^2 \hat{H}_{++}) = \rho_0(\hat{S}^2 \hat{H}_{++}^3) &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \\
\rho_0(\hat{1}_{--}) = -\rho_0(\hat{H}_{--}^2) &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
\rho_0(\hat{H}_{--}) = -\rho_0(\hat{H}_{--}^3) &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 \end{pmatrix} \\
\rho_0(\hat{S} \hat{H}_{--}) = -\rho_0(\hat{S} \hat{H}_{--}^3) &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
\rho_0(\hat{S}^2 \hat{H}_{--}) = -\rho_0(\hat{S}^2 \hat{H}_{--}^3) &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 \end{pmatrix}
\end{aligned}$$

Definition 3.3.17 (*k*-Adjoint Residue). Let \hat{U} be an 8×8 matrix with entries

in $\mathbb{D}[\frac{1}{\alpha}]$. Furthermore, let \hat{U}_{++} permit the denominator exponent k , \hat{U}_{-+} and \hat{U}_{+-} permit the denominator exponent $k+1$, and \hat{U}_{--} permit the denominator exponent $k+2$. Then we define the k -adjoint residue of \hat{U} , in symbols $\hat{\rho}_k(\hat{U})$ and with $\hat{\rho}_k : \mathcal{M}_8(\mathbb{Z}[\frac{1}{2}, \frac{1}{\alpha}]) \rightarrow \mathcal{M}_8(\mathbb{Z}_3)$, as follows:

$$\hat{\rho}_k(\hat{U}) = \begin{pmatrix} \rho_k(\hat{U}_{++}) & \rho_{k+1}(\hat{U}_{+-}) \\ \rho_{k+1}(\hat{U}_{-+}) & \rho_{k+2}(\hat{U}_{--}) \end{pmatrix}.$$

When we write $\hat{\rho}_k(\hat{U}_{\pm\pm})$, it is to be understood we want the array associated with function $\hat{\rho}_k$ as it applies to the $(\pm\pm)$ quadrant. This means that when we write e.g. $\hat{\rho}_k(\hat{U}_{+-})$, we really mean $\rho_{k+1}(\hat{U}_{+-})$

Remark 3.3.18. Let us briefly examine the consequences of left- or right-multiplication by a Clifford when considering only the k -adjoint residue of a matrix. In particular, let \hat{U} be some adjoint representation of an operator where \hat{U} has entries in $\mathbb{Z}[\frac{1}{2}, \frac{1}{\alpha}]$ and is such that $\hat{\rho}_k(\hat{U})$ is well defined. Right multiplication of \hat{U} by an adjoint representation \hat{C} of a Clifford would yield

$$\hat{U} \cdot \hat{C} = \begin{pmatrix} \hat{U}_{++}\hat{C}_{++} + \hat{U}_{+-}\hat{C}_{-+} & \hat{U}_{++}\hat{C}_{+-} + \hat{U}_{+-}\hat{C}_{--} \\ \hat{U}_{-+}\hat{C}_{++} + \hat{U}_{--}\hat{C}_{-+} & \hat{U}_{-+}\hat{C}_{+-} + \hat{U}_{--}\hat{C}_{--} \end{pmatrix}.$$

Calculating the relevant k -residues of the resulting matrix, we have the following

relations:

$$\begin{aligned}
\rho_k((\hat{U} \cdot \hat{C})_{++}) &= \rho_k(\hat{U}_{++}) \cdot \rho_0(\hat{C}_{++}) + \rho_{k+1}(\hat{U}_{+-}) \cdot \rho_0\left(\frac{1}{\alpha}\hat{C}_{-+}\right) \\
&= \rho_k(\hat{U}_{++}) \cdot \rho_0(\hat{C}_{++}) \\
\rho_{k+1}((\hat{U} \cdot \hat{C})_{-+}) &= \rho_{k+1}(\hat{U}_{-+}) \cdot \rho_0(\hat{C}_{++}) + \rho_{k+2}(\hat{U}_{--}) \cdot \rho_0\left(\frac{1}{\alpha}\hat{C}_{-+}\right) \\
&= \rho_{k+1}(\hat{U}_{-+}) \cdot \rho_0(\hat{C}_{++}) \\
\rho_{k+1}((\hat{U} \cdot \hat{C})_{+-}) &= \rho_{k+1}(\hat{U}_{++}) \cdot \rho_0(\hat{C}_{+-}) + \rho_{k+1}(\hat{U}_{+-}) \cdot \rho_0(\hat{C}_{--}) \\
&= \rho_{k+1}(\hat{U}_{+-}) \cdot \rho_0(\hat{C}_{--}) \\
\rho_{k+2}((\hat{U} \cdot \hat{C})_{--}) &= \rho_{k+1}(\hat{U}_{-+}) \cdot \rho_1(\hat{C}_{+-}) + \rho_{k+2}(\hat{U}_{--}) \cdot \rho_0(\hat{C}_{--}) \\
&= \rho_{k+2}(\hat{U}_{--}) \cdot \rho_0(\hat{C}_{--})
\end{aligned}$$

Left multiplication by a Clifford yields a similar set of relations:

$$\begin{aligned}
\rho_k((\hat{C} \cdot \hat{U})_{++}) &= \rho_0(\hat{C}_{++}) \cdot \rho_k(\hat{U}_{++}) + \rho_0\left(\frac{1}{\alpha}\hat{C}_{+-}\right) \cdot \rho_{k+1}(\hat{U}_{-+}) \\
&= \rho_0(\hat{C}_{++}) \cdot \rho_k(\hat{U}_{++}) \\
\rho_{k+1}((\hat{C} \cdot \hat{U})_{-+}) &= \rho_0(\hat{C}_{-+}) \cdot \rho_{k+1}(\hat{U}_{++}) + \rho_0(\hat{C}_{--}) \cdot \rho_{k+1}(\hat{U}_{-+}) \\
&= \rho_0(\hat{C}_{--}) \cdot \rho_{k+1}(\hat{U}_{-+}) \\
\rho_{k+1}((\hat{C} \cdot \hat{U})_{+-}) &= \rho_0(\hat{C}_{++}) \cdot \rho_{k+1}(\hat{U}_{+-}) + \rho_0\left(\frac{1}{\alpha}\hat{C}_{+-}\right) \cdot \rho_{k+2}(\hat{U}_{--}) \\
&= \rho_0(\hat{C}_{++}) \cdot \rho_{k+1}(\hat{U}_{+-}) \\
\rho_{k+2}((\hat{C} \cdot \hat{U})_{--}) &= \rho_1(\hat{C}_{-+}) \cdot \rho_{k+1}(\hat{U}_{+-}) + \rho_0(\hat{C}_{--}) \cdot \rho_{k+2}(\hat{U}_{--}) \\
&= \rho_0(\hat{C}_{--}) \cdot \rho_{k+2}(\hat{U}_{--})
\end{aligned}$$

By these equations we immediately have for any adjoint representation \hat{C} of a Clifford the following multiplicative rules for $\hat{\rho}_k$:

$$\hat{\rho}_k(\hat{U} \cdot \hat{C}) = \hat{\rho}_k(\hat{U}) \cdot \rho_0(\hat{C}) \quad \text{and} \quad \hat{\rho}_k(\hat{C} \cdot \hat{U}) = \rho_0(\hat{C}) \cdot \hat{\rho}_k(\hat{U})$$

By Remark 3.3.16, we know $\rho_0(\hat{C})$ is simply a generalized permutation matrix with $\rho_0(\hat{C}_{++})$ a true permutation matrix and $\rho_0(\hat{C}_{--})$ a generalized permutation matrix with the same nonzero pattern as $\rho_0(\hat{C}_{++})$. This means that (left-) right-multiplication by a Clifford simply corresponds to a permutation of (rows) columns, with the first 4 undergoing a true permutation and the last 4 receiving the same underlying permutation with potential (row-) column-wide multiplicative factors applied.

Definition 3.3.19 (Clifford Equivalence). Let \hat{U} and \hat{V} be adjoint representations of $\mathcal{C} + T$ operators U and V such that there exists a Clifford operator C with adjoint representation \hat{C} where $\hat{U} \cdot \hat{C} = \hat{V}$. If $\hat{\rho}_k(\hat{U})$ is well defined, then we know that $\hat{\rho}_k(\hat{V})$ is also well defined and call these k -adjoint residues *Clifford equivalent*, in symbols $\hat{\rho}_k(\hat{U}) \sim_{\mathcal{C}} \hat{\rho}_k(\hat{V})$, by which we mean $\hat{\rho}_k(\hat{U}) \cdot \rho_0(\hat{C}) = \hat{\rho}_k(\hat{V})$. We also extend this notion to quadrants, meaning that $\hat{\rho}_k(\hat{U}_{\pm\pm}) \sim_{\mathcal{C}} \hat{\rho}_k(\hat{V}_{\pm\pm})$ if and only if $\hat{\rho}_k(\hat{U}) \sim_{\mathcal{C}} \hat{\rho}_k(\hat{V})$ for the particular Clifford \hat{C} .

Proposition 3.3.20. *Let $U \in \mathcal{C} + T$ be a canonical form, and \hat{U} be the adjoint representation of U . Let n be the T -count of U . Then the least denominator exponent of \hat{U}_{++} is $k = 2n$ and one of the following holds:*

- $n = 0$ and U is a Clifford operator.
- $n > 0$ and one of 8 distinguishable cases holds for $\hat{\rho}_{2n}(\hat{U})$:

$$\hat{\rho}_{2n}(\hat{U}_{++}) \sim_c \rho_0(\hat{M}_{++}) \cdot \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \end{pmatrix},$$

$$\hat{\rho}_{2n}(\hat{U}_{--}) \sim_c \rho_0(\hat{M}_{--}) \cdot \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix},$$

and the leftmost syllable is MT with Clifford prefix $M \in \mathcal{LM}$

Proof. By direct computation, these statements hold for all canonical forms up to T -count three. In particular, enumerating these canonical forms gives the further condition that

$$\hat{\rho}_{2n}(\hat{U}_{+-}) = \hat{\rho}_{2n}(\hat{U}_{++}) \quad \text{and} \quad \hat{\rho}_{2n}(\hat{U}_{-+}) = \hat{\rho}_{2n}(\hat{U}_{--})$$

for all canonical forms of T -count $n = 2$ and $n = 3$ without a rightmost Clifford. Let \hat{U}_{n,M_n} be an adjoint representation of a canonical form with T -count $n > 2$ and leftmost syllable M_n with Clifford prefix $M'_n \in \mathcal{LM}$. Let $\hat{U}_{n-1,M_{n-1}} = \hat{M}_n^T \hat{U}_{n,M_n}$ such that it is also an adjoint representation of a canonical form with T -count

$n - 1 > 1$ and leftmost syllable M_{n-1} with Clifford prefix $M'_{n-1} \in \mathcal{L}'\mathcal{M}$. Consider left-multiplication of \hat{U}_{n,M_n} by \hat{T} :

$$\hat{T}\hat{U}_{n,M_n} = \begin{pmatrix} \hat{T}_{++} & \hat{T}_{+-} \\ -\hat{T}_{+-} & \hat{T}_{++} \end{pmatrix} \begin{pmatrix} (\hat{U}_{n,M_n})_{++} & (\hat{U}_{n,M_n})_{+-} \\ (\hat{U}_{n,M_n})_{-+} & (\hat{U}_{n,M_n})_{--} \end{pmatrix}.$$

As M_1 is the leftmost syllable of \hat{U}_{n,M_n} , we can also rewrite some of its quadrants as

$$\begin{aligned} (\hat{U}_{n,M_n})_{-+} &= (\hat{M}_n)_{-+}(\hat{U}_{n-1,M_{n-1}})_{++} + (\hat{M}_n)_{--}(\hat{U}_{n-1,M_{n-1}})_{-+} \\ (\hat{U}_{n,M_n})_{--} &= (\hat{M}_n)_{-+}(\hat{U}_{n-1,M_{n-1}})_{+-} + (\hat{M}_n)_{--}(\hat{U}_{n-1,M_{n-1}})_{--} \end{aligned}$$

Using these substitutions, we may write the following equations for the resulting quadrant matrices of $\hat{T}\hat{U}_{n,M_n}$:

$$\begin{aligned} (\hat{T}\hat{U}_{n,M_n})_{++} &= \hat{T}_{++}(\hat{U}_{n,M_n})_{++} + \hat{T}_{+-}(\hat{M}_n)_{-+}(\hat{U}_{n-1,M_{n-1}})_{++} \\ &\quad + \hat{T}_{+-}(\hat{M}_n)_{--}(\hat{U}_{n-1,M_{n-1}})_{-+} \\ (\hat{T}\hat{U}_{n,M_n})_{-+} &= -\hat{T}_{+-}(\hat{U}_{n,M_n})_{++} + \hat{T}_{++}(\hat{U}_{n,M_n})_{-+} \\ (\hat{T}\hat{U}_{n,M_n})_{+-} &= \hat{T}_{++}(\hat{U}_{n,M_n})_{+-} + \hat{T}_{+-}(\hat{M}_n)_{-+}(\hat{U}_{n-1,M_{n-1}})_{+-} \\ &\quad + \hat{T}_{+-}(\hat{M}_n)_{--}(\hat{U}_{n-1,M_{n-1}})_{--} \\ (\hat{T}\hat{U}_{n,M_n})_{--} &= -\hat{T}_{+-}(\hat{U}_{n,M_n})_{+-} + \hat{T}_{++}(\hat{U}_{n,M_n})_{--}. \end{aligned}$$

Assume that $\hat{U}_{n,M_n,M_{n-1}}$ and $\hat{U}_{n-1,M_{n-1}}$ have the following $2n$ - and $2(n-1)$ -adjoint

residues, respectively:

$$\hat{\rho}_{2n}(\hat{U}_{n,M_n}) \sim_{\mathcal{C}} \rho_0(\hat{M}'_n) \cdot \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{array} \right)$$

$$\hat{\rho}_{2(n-1)}(\hat{U}_{n-1,M_{n-1}}) \sim_{\mathcal{C}} \rho_0(\hat{M}'_{n-1}) \cdot \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{array} \right)$$

Now, consider $\hat{\rho}_{2(n+1)}(\hat{T}\hat{U}_{n,M_n})$. Using Remarks 3.3.8 and 3.3.18 and our equations

for $(\hat{T}\hat{U}_{n,M_n})_{\pm\pm}$, we have

$$\begin{aligned}
\hat{\rho}_{2(n+1)}((\hat{T}\hat{U}_{n,M_n})_{++}) &= \rho_2(\hat{T}_{++}) \cdot \rho_{2n}((\hat{U}_{n,M_n})_{++}) \\
&\quad + \rho_4(\hat{T}_{+-}(\hat{M}_n)_{-+}) \cdot \rho_{2(n-1)}((\hat{U}_{n-1,M_{n-1}})_{++}) \\
&\quad + \rho_3(\hat{T}_{+-}(\hat{M}_n)_{--}) \cdot \rho_{2n-1}((\hat{U}_{n-1,M_{n-1}})_{-+}) \\
\hat{\rho}_{2(n+1)}((\hat{T}\hat{U}_{n,M_n})_{-+}) &= -\rho_3(\hat{T}_{+-}) \cdot \rho_{2n}((\hat{U}_{n,M_n})_{++}) \\
&\quad + \rho_2(\hat{T}_{++}) \cdot \rho_{2n+1}((\hat{U}_{n,M_n})_{-+}) \\
\hat{\rho}_{2(n+1)}((\hat{T}\hat{U}_{n,M_n})_{+-}) &= \rho_2(\hat{T}_{++}) \cdot \rho_{2n+1}((\hat{U}_{n,M_n})_{+-}) \\
&\quad + \rho_4(\hat{T}_{+-}(\hat{M}_n)_{-+}) \cdot \rho_{2n-1}((\hat{U}_{n-1,M_{n-1}})_{+-}) \\
&\quad + \rho_3(\hat{T}_{+-}(\hat{M}_n)_{--}) \cdot \rho_{2n}((\hat{U}_{n-1,M_{n-1}})_{--}) \\
\hat{\rho}_{2(n+1)}((\hat{T}\hat{U}_{n,M_n})_{--}) &= -\rho_3(\hat{T}_{+-}) \cdot \rho_{2n+1}((\hat{U}_{n,M_n})_{+-}) \\
&\quad + \rho_2(\hat{T}_{++}) \cdot \rho_{2n+2}((\hat{U}_{n,M_n})_{--})
\end{aligned}$$

Enumerating the 6 possibilities for $\rho_4(\hat{T}_{+-}(\hat{M}_n)_{-+})$ yields $\rho_4(\hat{T}_{+-}(\hat{M}_n)_{-+}) = 0_{4 \times 4}$.

Similarly, evaluation of the 36 distinct cases for the terms containing $\rho_3(\hat{T}_{+-}(\hat{M}_n)_{--})$ yields

$$\begin{aligned}
&\rho_3(\hat{T}_{+-}(\hat{M}_n)_{--}) \cdot \rho_{2n-1}((\hat{U}_{n-1,M_{n-1}})_{-+}) \\
&= \rho_3(\hat{T}_{+-}(\hat{M}_n)_{--}) \cdot \rho_{2n}((\hat{U}_{n-1,M_{n-1}})_{--}) = 0_{4 \times 4}.
\end{aligned}$$

Finally, the expressions $\rho_2(\hat{T}_{++}) \cdot \rho_{2n+1}((\hat{U}_{n,M_n})_{-+})$ and $\rho_2(\hat{T}_{++}) \cdot \rho_{2n+2}((\hat{U}_{n,M_n})_{--})$,

with 6 cases each, may be enumerated to yield

$$\rho_2(\hat{T}_{++}) \cdot \rho_{2n+1}((\hat{U}_{n,M_n})_{-+}) = \rho_2(\hat{T}_{++}) \cdot \rho_{2n+2}((\hat{U}_{n,M_n})_{--}) = 0_{4 \times 4}.$$

This leaves us a simplified set of equations

$$\begin{aligned} \hat{\rho}_{2(n+1)}((\hat{T}\hat{U}_{n,M_n})_{++}) &= \rho_2(\hat{T}_{++}) \cdot \rho_{2n}((\hat{U}_{n,M_n})_{++}) \\ \hat{\rho}_{2(n+1)}((\hat{T}\hat{U}_{n,M_n})_{-+}) &= -\rho_3(\hat{T}_{+-}) \cdot \rho_{2n}((\hat{U}_{n,M_n})_{++}) \\ \hat{\rho}_{2(n+1)}((\hat{T}\hat{U}_{n,M_n})_{+-}) &= \rho_2(\hat{T}_{++}) \cdot \rho_{2n+1}((\hat{U}_{n,M_n})_{+-}) \\ \hat{\rho}_{2(n+1)}((\hat{T}\hat{U}_{n,M_n})_{--}) &= -\rho_3(\hat{T}_{+-}) \cdot \rho_{2n+2}((\hat{U}_{n,M_n})_{+-}) \end{aligned}$$

Direct evaluation of the 6 options for each term yield only one possible resulting adjoint representation, summarized as follows $\rho_2(\hat{T}_{++}) \cdot \rho_{2n}((\hat{U}_{n,M_n,M_{n-1}})_{++})$. giving only one possible result:

$$\hat{\rho}_{2(n+1)}(\hat{T}\hat{U}_{n,M_n}) \sim \mathcal{C} \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{array} \right).$$

As $\hat{T}\hat{U}_{n,M_n,M_{n-1}}$ is itself an adjoint representation of a canonical form with T -count $n + 1$ and leftmost syllable T , left-multiplication by any element \hat{M}'_{n+1} from the adjoint representation for the set \mathcal{LM} is also an adjoint representation of a canonical form with T -count $n + 1$ and leftmost syllable M_{n+1} with Clifford Prefix M'_{n+1} . Calling this new operator $\hat{U}_{n+1,M_{n+1}}$, we have

$$\hat{\rho}_{2(n+1)}(\hat{U}_{n+1,M_{n+1}}) \sim_C \rho_0(\hat{M}'_{n+1}) \cdot \left(\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 & 0 & 2 & 2 & 2 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{array} \right).$$

This particular pattern is then persistent under an inductive argument, given two consecutive T -counts possess the stated properties. Because all T -count 2 and 3 canonical forms obey the requisite requirements, we thus have that any canonical form \hat{U}_{n,M_n} of T -count $n \geq 2$ and leftmost syllable M_n with Clifford prefix $M'_n \in$

\mathcal{LM} will obey the relations

$$\hat{\rho}_{2n}(\hat{U}_{++}) \sim_{\mathcal{C}} \rho_0((\hat{M}'_n)_{++}) \cdot \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \end{pmatrix},$$

$$\hat{\rho}_{2n}(\hat{U}_{-+}) \sim_{\mathcal{C}} \rho_0((\hat{M}'_n)_{--}) \cdot \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

Enumeration of canonical forms of T -count one shows that they likewise have this property, and so coupled with the fact that the lde of any Clifford operator is zero we have shown Proposition 3.3.20 to be true. \square

Proposition 3.3.21. *If M and N are different canonical forms then they represent different operators.*

Proof. Let U and V be different canonical forms with T -counts n and m respectively. If $n \neq m$, then by Proposition 3.3.20 the lde's of \hat{U}_{++} and \hat{V}_{++} differ and so must U and V . This leaves the case when $n = m$. Let U and V differ such that their first mismatched syllable starting from the left is the p th syllable counting from the right, with $U_{p,0}$ the associated canonical form of U truncated at this syllable as starting from the right such that $U = U_{n,p+1}U_{p,0}$. Then $\hat{U}_{n,p+1}^\top \cdot \hat{U} \neq \hat{U}_{n,p+1}^\top \cdot \hat{V}$ by Proposition 3.3.20, and thus U and V are different. Now, let U and V be such

that every syllable is identical, but their rightmost Cliffords are different. Then $U^\dagger V \in \mathcal{C} \setminus \{\mathbb{1}\}$ and therefore $U \neq V$. This enumerates all possible cases. \square

Corollary 3.3.22. *Let $U \in \mathcal{C} + T$ have adjoint representation \hat{U} with lde k of \hat{U}_{++} . Then the canonical form M associated with U has T -count $n = \frac{k}{2}$ and can be efficiently computed in $\mathcal{O}(n)$ arithmetic operations.*

Proof. From U , we compute the adjoint representation \hat{U} using a constant number of operations, in the process determining the lde k of \hat{U}_{++} . By Proposition 3.3.20, we have two cases depending on the value of k . If $k = 0$, U is equivalent to a Clifford operator C and M can be found via lookup table. If $k > 0$, k is even by Proposition 3.3.20 and so let $n = \frac{k}{2}$. Then we can find the leftmost syllable M_n in a constant number of operations by evaluating $\hat{\rho}_{2n}(\hat{U}_{++})$ and $\hat{\rho}_{2n}(\hat{U}_{-+})$. Now, calculate $\hat{U}' = \hat{M}_n^\dagger \hat{U}$ - by Proposition 3.3.20, we know \hat{U}' is the adjoint representation of a canonical form with lde $k - 2$ of \hat{U}'_{++} such that the T -count of \hat{U}' is $n - 1$. Carrying out this procedure recursively, we are left with the U equivalent canonical form

$$M = M_n M_{n-1} \dots M_1 C$$

where it took a constant number of operations to calculate each M_i and C , thus requiring an overall runtime of $\mathcal{O}(n)$. \square

We conclude with two important consequences of the uniqueness of canonical forms.

Proposition 3.3.23. *Canonical forms are T -optimal: for any canonical form with*

T -count n there are no equivalent $\mathcal{C} + T$ operators with a number of power of T gates less than n .

Proof. By Proposition 3.2.6 we know that every $\mathcal{C} + T$ operator admits a canonical form, and by Proposition 3.3.21, we will have that these canonical forms are both unique. Furthermore, by Remark 3.2.8, we know that in putting any $\mathcal{C} + T$ operator into either canonical form by the algorithms laid out in Corollary 3.2.7, the T -count may *only* decrease compared to the number of power of T gates. In combination, these statements suffice to show T -optimality. \square

Proposition 3.3.24. *Let $\varepsilon > 0$. There exists $U \in SU(3)$ whose ε -approximation by a Clifford+ T circuit requires a number n of T gates where $n \gtrsim 8 \log_6 \left(\frac{1}{\varepsilon}\right) - K$ for $K \approx 0.543$.*

Proof. This follows from the volume-counting argument described in Eq. (2.2). Indeed, there are $216/5(8 \cdot 6^n - 3)$ canonical forms of T -count at most n . Moreover, each ε -ball occupies a volume of $(\pi^4/24)\varepsilon^8$ as ε asymptotes towards zero (by which the 8-dimensional manifold $SU(3)$ becomes locally Euclidean). We need to cover the full volume $\sqrt{3}\pi^5$ of $SU(3)$ to guarantee that every operator can be approximated up to ε . Therefore n needs to satisfy

$$\frac{216}{5}(8 \cdot 6^n - 3) \frac{\pi^4}{24} \varepsilon^8 \gtrsim \sqrt{3}\pi^5,$$

from which the result follows. \square

3.4 Higher Prime Dimensions

Given that unique T -optimal MA normal forms exist for both qubits and qutrits, it is extremely natural to try and extend this result for higher prime dimensions. In fact, MA normal forms *do* generalize to higher dimensions. This fact has not been published to my knowledge, but has been demonstrated independently by (at a minimum) myself, Earl Campbell, and the pair of Akalank Jain and Shiroman Prakash, who had been co-authors on another paper considering qutrit $\mathcal{C} + T$ compiling which was published simultaneously and independently of our own [125]. Unfortunately, while existence has been established, proving uniqueness of these normal forms has been elusive. Nonetheless, for completeness we now present our proof of existence.

First, we wish to establish a subgroup of the Cliffords analogous to \mathcal{M} as defined in Section 3.2. In prime dimension $p \geq 5$, these are operators of the form

$$\Delta_a = \sum_{j=0}^{p-1} |aj\rangle \langle j|$$

for $a \in [[1..p - 1]]$. These operators can be explicitly represented as

$$HS^a HS^{\frac{1}{a}} HS^a P.$$

where $\frac{1}{a}$ is the multiplicative inverse of a modulo p and $P \in \mathcal{P}$ (precisely which P that is will be irrelevant for this discussion). This can be established by direct

computation, but it is more straightforward to consider the presentation of the Clifford group using $\text{SL}(2, \mathbb{Z}_p)$. Using our definitions from Eq. (1.2), we compute this product as

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{1}{a} & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & a^{-1} \end{bmatrix}$$

Checking the action of Δ_a on X and Z then confirms that the circuit is of the correct form. These operators form a group \mathcal{M} as $\mathbb{1} = \Delta_1$, $\Delta_a^{-1} = \Delta_{\frac{1}{a}}$, and $\Delta_a \Delta_b = \Delta_{ab}$.

Consider conjugation of the T -gate raised to some power t by Δ_a :

$$\begin{aligned} \Delta_a T^t \Delta_a^\dagger &= \sum_{j=0}^{p-1} \omega^{tj^3} |aj\rangle \langle aj| \\ &= \sum_{j=0}^{p-1} \omega^{\frac{t}{a^3} j^3} |j\rangle \langle j| \\ &= T_{a^3}^t \end{aligned} \tag{3.8}$$

By conjugation with Δ_a , we can thus change the power of T . We now consider two families of primes. For primes $p = 2 \pmod{3}$, cubing is a bijection from $[[1..p-1]]$ to $[[1..p-1]]$ for all $a \in [[1..p-1]]$. This means that if a T gate is raised to any power, we can always find a Clifford that upon conjugation transforms that power to a single T gate, as we can choose the a which upon cubing becomes the inverse of t . In the case where $p = 1 \pmod{3}$, however, cubing is a non-surjective function from $[[1..p-1]]$ to $[[1..p-1]]$ – cubing only maps to a third of the elements of $[[1..p-1]]$. This means that we can always map a power of T into one of three operators: T ,

T^g , or T^{g^2} where g is a generator of the group $([[1..p-1]], \cdot)$. In combination with Eq. (3.8), we can thus conclude for prime dimension $p \geq 5$ qudits that

$$\mathcal{M}T^a \subseteq \begin{cases} \mathcal{M}T\mathcal{M} & p = 2 \pmod{3} \\ \bigcup_{b \in \mathbb{Z}_3} \mathcal{M}'T^{g^b}\mathcal{M}' & p = 1 \pmod{3} \end{cases} \quad (3.9)$$

for any $a \in [[1..p-1]]$, g a generator of $([[1..p-1]], \cdot)$, and $\mathcal{M}' \subset \mathcal{M}$ a cardinality $\frac{p-1}{3}$ subset of \mathcal{M} such that there is one representative $\Delta_a \in \mathcal{M}'$ for each value of $a^3 \pmod{p}$. Finally, we note that \mathcal{M} has no phases and a cardinality $p-1$.

We can likewise define an analogue to \mathcal{S} for general prime qudit dimension. Taking again X and S as generators, we define this group as $\mathcal{S} = \langle X, S \rangle$. Both of these generators commute with T :

$$XT = T\omega^{-1}XZ^{-6}S^{-6}$$

$$ST = TS$$

As Z and ω are elements of \mathcal{S} , we can conclude $\mathcal{S}T = T\mathcal{S}$. The group can be factored as S^sP for $P \in \mathcal{P}$ and $s \in \mathbb{Z}_p$, and so it has a cardinality of p^4 (p^3 up to a phase). Finally, it is straightforward to show

$$\Delta_a\mathcal{S}\Delta_a = \mathcal{S}.$$

for any a . This means that the subgroup which we shall call \mathcal{R} generated by Δ_g, X ,

and S for g a generator of the group $([[1..p-1]], \cdot)$ can conveniently be expressed as

$$\mathcal{R} = \mathcal{M}\mathcal{S}.$$

This subgroup has $p^4(p-1)$ elements, and $p^3(p-1)$ up to a phase.

We now construct the full Clifford group using the cosets with respect to \mathcal{R} . By Eq. (1.1), the order of the single qudit Clifford group up to a phase is $p^3(p^2-1)$, and so we know there should be $p+1$ cosets in total. The set of cosets with respect to \mathcal{R} are

$$\{\mathcal{R}\} \cup \{S^s H\mathcal{R} \mid s \in \mathbb{Z}_p\}$$

which can be verified by checking that every element of $\text{SL}(2, \mathbb{Z}_p)$ is contained in this set of sets. The lack of a representation for Pauli group elements is rectified by the presence of this group in the construction of \mathcal{R} , and so we merely need to compute the representations using \mathcal{R}/\mathcal{P} . We calculate

$$\begin{aligned} \mathcal{R}/\mathcal{P} &= \left\{ \left[\begin{array}{c|c} \left[\begin{array}{cc} a & 0 \\ 0 & \frac{1}{a} \end{array} \right] & \left[\begin{array}{cc} 1 & 0 \\ b & 1 \end{array} \right] \\ \hline a \in [[1..p-1]], b \in \mathbb{Z}_p \end{array} \right\} \\ &= \left\{ \left[\begin{array}{c|c} \left[\begin{array}{cc} a & 0 \\ \frac{b}{a} & \frac{1}{a} \end{array} \right] & \\ \hline a \in [[1..p-1]], b \in \mathbb{Z}_p \end{array} \right\} \end{aligned} \quad (3.10)$$

which is precisely every element of $\text{SL}(2, \mathbb{Z}_p)$ with an entry of zero in the upper-right

corner. Using Eq. (3.10), we can then compute for $s \in \mathbb{Z}_p$

$$\begin{aligned}
S^s H \mathcal{R} / \mathcal{P} &= \left\{ \left[\begin{array}{cc|cc} 0 & -1 & a & 0 \\ 1 & s & \frac{b}{a} & \frac{1}{a} \end{array} \right] \middle| a \in [[1..p-1]], b \in \mathbb{Z}_p \right\} \\
&= \left\{ \left[\begin{array}{cc|cc} -\frac{b}{a} & -\frac{1}{a} & & \\ a + \frac{sb}{a} & \frac{s}{a} & & \end{array} \right] \middle| a \in [[1..p-1]], b \in \mathbb{Z}_p \right\} \tag{3.11}
\end{aligned}$$

which is precisely every element of $\text{SL}(2, \mathbb{Z}_p)$ with a non-zero entry of $-\frac{1}{a}$ in the upper-right corner and an entry of $\frac{s}{a}$ in the lower-right corner. The union of the set in Eq. (3.10) and the sets for every value of $s \in \mathbb{Z}_p$ in Eq. (3.11) is then equal to $\mathcal{C} / \mathcal{P}$ in this representation. Defining the set $\mathcal{L} = \{\mathbb{1}\} \cup \{S^s H \mid s \in \mathbb{Z}_p\}$, we can then conclude that

$$\mathcal{C} = \mathcal{L} \mathcal{M} \mathcal{S} \tag{3.12}$$

for all prime dimensions $p \geq 5$ as in the qutrit case. We also define the set $\mathcal{L}' = \mathcal{L} / \{\mathbb{1}\}$

The existence of prime dimension $p \geq 5$ Qudit normal forms then follows from considering some general circuit

$$U = C_n T^{a_n} C_{n-1} T^{a_{n-1}} \dots C_1 T^{a_1} C_0$$

with n power-of- T gates. Using Eqs. (3.9) and (3.12) and $\mathcal{S}T = T\mathcal{S}$, we can then

immediately conclude under the same line of reasoning as in Proposition 3.2.6 that

$$U \subseteq \begin{cases} \mathcal{L}\mathcal{M}T(\mathcal{L}'\mathcal{M}T)^m \mathcal{C} & p = 2 \pmod{3} \\ \mathcal{L}\mathcal{M}T\left(\bigcup_{b \in \mathbb{Z}_3} \mathcal{L}'\mathcal{M}'T^{g^b}\right)^m \mathcal{C} & p = 1 \pmod{3} \end{cases}$$

for $m + 1 \leq n$ and g a generator of $([[1..p - 1]], \cdot)$. We call this a qudit Matsumoto-Amano normal form, and note that rewriting some operator in this form can only reduce the power-of- T gate count and takes a time linear in the size of the circuit.

3.5 Conclusion

Significant advances in our understanding of the Clifford+ T group for both single- and multi-qubit circuits have been made in the past decade. Analogous results for qudits of higher dimension, however, remained elusive. In this chapter we contribute to the theory of single-qudit Clifford+ T circuits by providing a canonical form for fault-tolerant single-qudit Clifford+ T circuits. We show that every Clifford+ T operator admits a unique canonical representation and that this representation is T -optimal. We provide a linear-time algorithm for computing this algorithm which we note has been implemented online at Xiaoning Bian's homepage [126]. Finally, we demonstrate the existence of a similar normal form in higher prime dimension. Unfortunately, there is no characterization of these circuits as there is for $\mathcal{C} + T$ qubits as described in Section 2.4. This is a necessary ingredient to develop efficient inexact synthesis algorithms for qudit $\mathcal{C} + T$ circuits. The failure to establish such a characterization leaves this as an avenue for future work – in the

interim, we turn our attention to multi-qubit circuits.

Chapter 4: Restricted Clifford + T Circuit Synthesis¹

4.1 Introduction

Before attempting to develop normal forms for multi-qubit circuits, we focus on characterizing circuits over universal gate sets which are simpler than the $\mathcal{C} + T$ gate set. This shall provide motivation for the gate sets in which to search for such normal forms. Kliuchnikov, Maslov, and Mosca showed in [92] that a 2-dimensional unitary matrix V can be exactly represented by a single-qubit Clifford+ T circuit if and only if the entries of V belong to the ring $\mathbb{D}[\zeta]$ for $\zeta = e^{\frac{i\pi}{4}}$ a primitive eighth root of unity. This result gives a number-theoretic characterization of single-qubit Clifford+ T circuits. In [96], Giles and Selinger extended the characterization of Kliuchnikov et al. to multi-qubit Clifford+ T circuits by proving that a 2^n -dimensional unitary matrix can be exactly represented by an n -qubit Clifford+ T circuit if and only if its entries belong to $\mathbb{D}[\zeta]$.

These number-theoretic characterizations provide great insight into the structure of Clifford+ T circuits. As a result, single-qubit Clifford+ T circuits are now very well understood [90, 91, 97, 100, 128]. In contrast, our understanding of multi-qubit Clifford+ T circuits remains more limited, despite interesting results [129–132].

¹This work is a slightly modified version of [127].

One of the reasons for this limitation is that large unitary matrices over $\mathbb{D}[\zeta]$ are hard to analyze. In order to circumvent the difficulties associated with multi-qubit Clifford+ T circuits, restricted gate sets have been considered in the literature. This led to important developments in the study of multi-qubit Clifford, CNOT+ T , and CNOT-dihedral circuits [67, 106–111]. Unfortunately, the simpler structure of these restricted gate sets comes at a cost: they are not universal for quantum computing.

In this chapter, our goal is to address both of these limitations by considering universal restrictions of the Clifford+ T gate set. To this end, we study circuits corresponding to unitary matrices over proper subrings of $\mathbb{D}[\zeta]$, focusing on \mathbb{D} , $\mathbb{D}[\sqrt{2}]$, $\mathbb{D}[\sqrt{-2}]$, and $\mathbb{D}[i]$. For each subring, we find a set of quantum gates G with the property that circuits over G correspond to unitary matrices over the given ring. Writing $\mathcal{U}_{2^n}(R)$ for the group of $2^n \times 2^n$ unitary matrices over a ring R , our main results can then be summarized in the following theorem.

Theorem. *A $2^n \times 2^n$ unitary matrix V can be exactly represented by an n -qubit circuit over*

- (i) $\{X, CX, CCX, H \otimes H\}$ if and only if $V \in \mathcal{U}_{2^n}(\mathbb{D})$,
- (ii) $\{X, CX, CCX, H, CH\}$ if and only if $V \in \mathcal{U}_{2^n}(\mathbb{D}[\sqrt{2}])$,
- (iii) $\{X, CX, CCX, F\}$ if and only if $V \in \mathcal{U}_{2^n}(\mathbb{D}[\sqrt{-2}])$, and
- (iv) $\{X, CX, CCX, \zeta H, S\}$ if and only if $V \in \mathcal{U}_{2^n}(\mathbb{D}[i])$,

where $\zeta = e^{i\pi/4}$ and $F \propto \sqrt{H}$. Moreover, in (i)-(iv), a single ancilla is sufficient.

The gate sets in items (i)-(iv) of the above theorem are all universal for quantum computing [133, 134], and we sometimes refer to circuits over these gate sets as *integral*, *real*, *imaginary*, and *Gaussian Clifford+T* circuits, respectively. As a corollary to the above theorem, we obtain two additional characterizations of universal gate sets.

Corollary. *A $2^n \times 2^n$ unitary matrix V can be exactly represented by an n -qubit circuit over*

(i) $\{X, CX, CCX, H\}$ *if and only if $V = W/\sqrt{2^q}$ for some matrix W over \mathbb{Z} and some $q \in \mathbb{N}$, and*

(ii) $\{X, CX, CCX, H, S\}$ *if and only if $V = W/\sqrt{2^q}$ for some matrix W over $\mathbb{Z}[i]$ and some $q \in \mathbb{N}$.*

Moreover, in (i) and (ii), a single ancilla is sufficient.

Restrictions similar to the ones considered here were previously studied in the context of foundations [135], randomized benchmarking [136], and graphical languages for quantum computing [137–139]. Furthermore, our study fits within a larger program, initiated by Aaronson, Grier, and Schaeffer, which aims at classifying quantum operations. Such classifications exist for classical reversible operations [85] and stabilizer operations [140], but no classification is known for a universal family of quantum operations. In this context, our work can be seen as a partial classification of the universal extensions of the set of classical reversible gates $\{X, CX, CCX\}$. This perspective is illustrated in Figure 4.1, which depicts a fragment of the lattice of subgroups of $\mathbb{D}[\zeta]$.

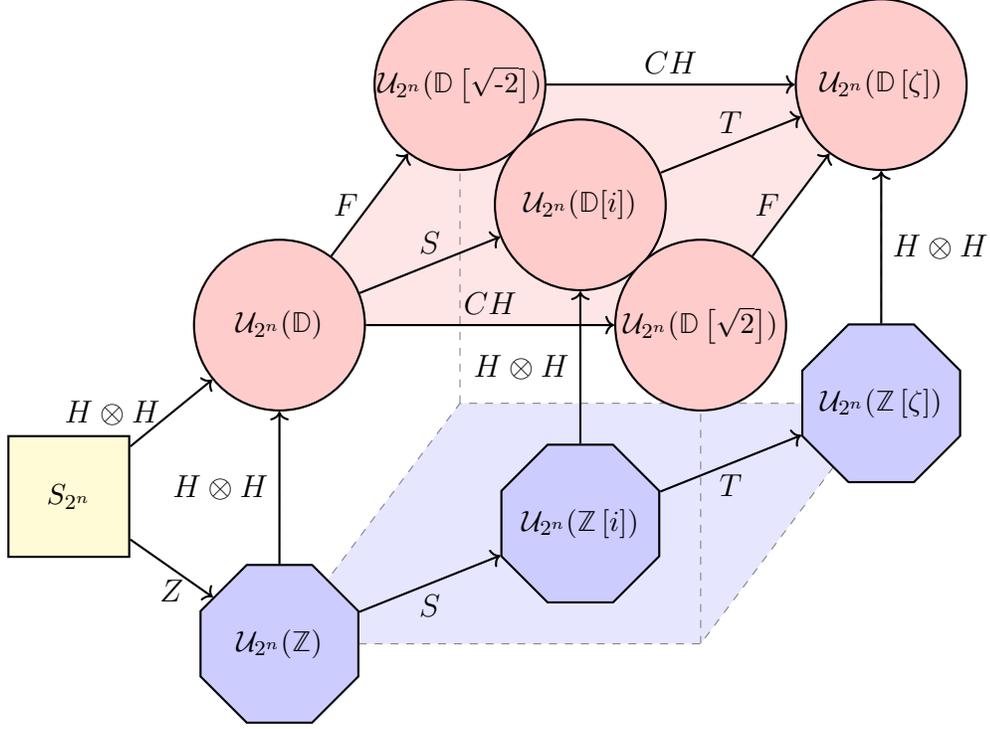


Figure 4.1: Some subgroups of $\mathcal{U}_{2^n}(\mathbb{D}[\zeta])$. To the left of the cube, in yellow, the symmetric group S_{2^n} corresponds to circuits over the gate set $\{X, CX, CCX\}$. On the bottom face of the cube, in blue, are generalized symmetric groups, and on the top face of the cube, in red, are universal subgroups of $\mathcal{U}_{2^n}(\mathbb{D}[\zeta])$. The edges of the lattice denote inclusion. The gates labeling the edges are sufficient to extend the expressive power of a gate set from one subgroup to the next (and no further). For example, the edge labeled Z going from S_{2^n} to $\mathcal{U}_{2^n}(\mathbb{Z})$ indicates that adding the Z gate to $\{X, CX, CCX\}$ produces a gate set expressive enough to represent every matrix in $\mathcal{U}_{2^n}(\mathbb{Z})$ (but not every matrix in $\mathcal{U}_{2^n}(\mathbb{Z}[i])$).

The rest of the paper is organized as follows. In Section 4.2, we give an overview of our methods. In Section 4.3, we introduce the rings and matrices which will be used throughout the paper. In Section 4.4, we show that certain useful matrices can be exactly represented by restricted Clifford+ T circuits. Section 4.5 contains the proofs of our various number-theoretic characterizations. We conclude in Section 4.6.

4.2 Overview

Unrestricted Clifford+ T circuits are generated by H , CX , and T . By Eq. (1.3), we know the generators and thus all circuits over this gate set have entries in the ring $\mathbb{Z}[1/\sqrt{2}, \zeta] = \mathbb{Z}[1/\sqrt{2}, i] = \mathbb{D}[\zeta]$. Hence, if a matrix V can be represented exactly by an n -qubit Clifford+ T circuit, then $V \in \mathcal{U}_{2^n}(\mathbb{D}[\zeta])$, the group of $2^n \times 2^n$ unitary matrices with entries in $\mathbb{D}[\zeta]$. Showing that the ring $\mathbb{D}[\zeta]$ characterizes Clifford+ T circuits thus amounts to proving the converse implication.

The original insight of Kliuchnikov, Maslov and Mosca in the single-qubit Clifford+ T case was to reduce the problem of exact synthesis to the problem of state preparation. The latter problem is to find, given a target vector $v \in \mathbb{D}[\zeta]^n$, a sequence G_1, \dots, G_ℓ of Clifford+ T gates such that $G_\ell \cdots G_1 e_1 = v$ or, equivalently, such that $G_1^\dagger \cdots G_\ell^\dagger v = e_1$. Kliuchnikov et al. realized that this sequence of gates can be found by first writing v as $v = u/\sqrt{2}^q$ for some $u \in \mathbb{Z}[\zeta]$ and then iteratively reducing the exponent q .

This basic premise was extended by Giles and Selinger to the multi-qubit context by adding an outer induction over the columns of an n -qubit unitary, as described briefly in Section 2.4. This method amounts to performing a constrained Gaussian elimination where the row operations are restricted to a few basic moves. The Giles-Selinger algorithm proceeds by reducing the leftmost column of an $n \times n$ unitary matrix to the first standard basis vector by applying a sequence of one- and two-level matrices, which act non-trivially on at most two components of a vector, before recursively dealing with the remaining submatrix. If the target unitary is

$V = \left[v \mid V' \right]$, then the Giles-Selinger algorithm first constructs a sequence of matrices G_1, \dots, G_ℓ such that $G_1 \cdots G_\ell v = e_1$. Left-multiplying V by this sequence of matrices then yields

$$G_1 \cdots G_\ell \left[v \mid V' \right] = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & V'' & \\ 0 & & & \end{array} \right]$$

where V'' is unitary. The fact that the matrices used in this reduction act non-trivially on no more than two rows of the matrix ensures that when the algorithm recursively reduces the columns of V'' it does so without perturbing the previously fixed columns. The Giles-Selinger algorithm thus relies on the following two facts.

1. A unit vector in $\mathbb{D}[\zeta]^n$ can be reduced to a standard basis vector by using one- and two-level matrices and
2. The required one- and two-level matrices can be exactly represented by Clifford+ T circuits.

While each of our characterizations presents specificities, our method in characterizing restricted Clifford+ T circuits follows this general structure.

4.3 Rings and Matrices

In this section, we discuss the rings and matrices that will be used throughout the paper. For further details, the reader is encouraged to consult [141].

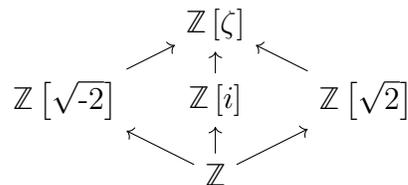
4.3.1 Rings

We will use the extensions of \mathbb{Z} defined below.

Definition 4.3.1. Let

- $\mathbb{Z}[\sqrt{2}] = \{x_0 + x_1\sqrt{2} \mid x_0, x_1 \in \mathbb{Z}\},$
- $\mathbb{Z}[\sqrt{-2}] = \{x_0 + x_1\sqrt{-2} \mid x_0, x_1 \in \mathbb{Z}\},$
- $\mathbb{Z}[i] = \{x_0 + x_1i \mid x_0, x_1 \in \mathbb{Z}\},$ and
- $\mathbb{Z}[\zeta] = \{x_0 + x_1\zeta + x_2\zeta^2 + x_3\zeta^3 \mid x_0, x_1, x_2, x_3 \in \mathbb{Z}\}.$

The rings $\mathbb{Z}[\sqrt{2}]$, $\mathbb{Z}[\sqrt{-2}]$, $\mathbb{Z}[i]$, and $\mathbb{Z}[\zeta]$ are known as the ring of *quadratic integers with radicand 2*, the ring of *quadratic integers with radicand -2*, the ring of *Gaussian integers*, and the ring of *cyclotomic integers of degree 8*, respectively. All of these rings are distinct subrings of $\mathbb{Z}[\zeta]$ and we have the inclusions depicted in the lattice of subrings below.



Further to the rings introduced in Definition 4.3.1, we will consider extensions of \mathbb{D} .

Definition 4.3.2. Let

- $\mathbb{D}[\sqrt{2}] = \{x_0 + x_1\sqrt{2} \mid x_0, x_1 \in \mathbb{D}\},$
- $\mathbb{D}[\sqrt{-2}] = \{x_0 + x_1\sqrt{-2} \mid x_0, x_1 \in \mathbb{D}\},$
- $\mathbb{D}[i] = \{x_0 + x_1i \mid x_0, x_1 \in \mathbb{D}\},$ and
- $\mathbb{D}[\zeta] = \{x_0 + x_1\zeta + x_2\zeta^2 + x_3\zeta^3 \mid x_0, x_1, x_2, x_3 \in \mathbb{D}\}.$

If $v \in \mathbb{D}[\sqrt{2}]$, then v can be written as $v = u/2^q$ for some $q \in \mathbb{N}$ and some $u \in \mathbb{Z}[\sqrt{2}]$. A similar property holds for elements of $\mathbb{D}[\sqrt{-2}]$, $\mathbb{D}[i]$, and $\mathbb{D}[\zeta]$. The following proposition gives an explicit description of certain lesser-known rings of residues which will be useful in what follows.

Proposition 4.3.3. *We have*

- $\mathbb{Z}[\sqrt{2}]/(2) = \{0, 1, \sqrt{2}, 1 + \sqrt{2}\},$
- $\mathbb{Z}[\sqrt{-2}]/(2) = \{0, 1, \sqrt{-2}, 1 + \sqrt{-2}\},$
- $\mathbb{Z}[\sqrt{-2}]/(2\sqrt{-2}) = \{0, 1, 2, 3, \sqrt{-2}, 1 + \sqrt{-2}, 2 + \sqrt{-2}, 3 + \sqrt{-2}\},$ and
- $\mathbb{Z}[i]/(2) = \{0, 1, i, 1 + i\}.$

Proof. To see, for example, that $\mathbb{Z}[\sqrt{2}]/(2) = \{0, 1, \sqrt{2}, 1 + \sqrt{2}\}$, note that $u = x_0 + x_1\sqrt{2}$ and $u' = x'_0 + x'_1\sqrt{2}$ are congruent modulo 2 if there exists an element $t = y_0 + y_1\sqrt{2}$ such that $u - u' = 2t$. This is the case if and only if $(x_0 - x'_0) +$

$(x_1 - x'_1)\sqrt{2} = 2y_0 + 2y_1\sqrt{2}$ which in turn holds if and only if $x_0 \equiv x'_0 \pmod{2}$ and $x_1 \equiv x'_1 \pmod{2}$. \square

We will often take advantage of properties of residues. Some of the properties are generic. For example, if u and v are two elements of a ring R and $u \equiv v \pmod{2}$, then $u \pm v \equiv 0 \pmod{2}$. Other properties of residues are specific to the ambient ring. For example, an integer $u \in \mathbb{Z}$ is odd if and only if $u^2 \equiv 1 \pmod{4}$. Similarly, for an integer $u \in \mathbb{Z}$, we have $u \equiv 3 \pmod{4}$ if and only if $-u \equiv 1 \pmod{4}$. We now state important properties of residues in $\mathbb{Z}[\sqrt{-2}]$ and $\mathbb{Z}[i]$ for future reference. They can be established by reasoning using residue tables in the relevant quotient rings.

Proposition 4.3.4. *The following statements hold.*

- In $\mathbb{Z}[\sqrt{-2}]/(2)$, $u^\dagger u \equiv 0$ or 1 .
- If we have $u^\dagger u \equiv 1$ in $\mathbb{Z}[\sqrt{-2}]/(2)$, then $u \equiv 1, 3, 1 + \sqrt{-2}$, or $3 + \sqrt{-2}$ in $\mathbb{Z}[\sqrt{-2}]/(2\sqrt{-2})$.
- In $\mathbb{Z}[\sqrt{-2}]/(2\sqrt{-2})$, $u \equiv 3$ if and only if $-u \equiv 1$ and $u \equiv 3 + \sqrt{-2}$ if and only if $-u \equiv 1 + \sqrt{-2}$.

Proposition 4.3.5. *The following statements hold.*

- In $\mathbb{Z}[i]/(2)$, if $u^2 \equiv 1$, then $u \equiv 1$ or i .
- In $\mathbb{Z}[i]/(2)$, $u \equiv i$ if and only if $iu \equiv 1$.

4.3.2 Matrices

We write e_j for the j -th standard basis vector. If R is a ring, we write $\mathcal{M}_{n \times n'}(R)$ for the collection of $n \times n'$ matrices over R . We will use one-, two-, and four-level matrices which act non-trivially on only one, two, or four of the components of their input. These matrices will be defined using basic matrices. The construction is best explained with an example. If

$$V = \begin{bmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{bmatrix}$$

is a 2-dimensional unitary matrix, then in 3 dimensions the two-level operator of type V , which is denoted by $V_{[1,3]}$, is the matrix given below.

$$V_{[1,3]} = \begin{bmatrix} v_{1,1} & 0 & v_{1,2} \\ 0 & 1 & 0 \\ v_{2,1} & 0 & v_{2,2} \end{bmatrix}$$

Definition 4.3.6. Let W be an $n \times n$ unitary matrix, let $n \leq n'$, and let $a_1, \dots, a_n \in [[1..n']]$. The n -level matrix of type W is the $n' \times n'$ unitary matrix $W_{[a_1, \dots, a_n]}$ defined by

$$W_{[a_1, \dots, a_n]}_{j,k} = \begin{cases} W_{j',k'} & \text{if } j = a_{j'} \text{ and } k = a_{k'} \\ I_{j,k} & \text{otherwise.} \end{cases}$$

We finally note that we have valid notions for denominator exponents of any

tensor with entries in the ring \mathbb{D} and its extensions.

4.4 Circuits

In this section, we review basic circuit constructions which will be useful below.

A more detailed discussion of quantum circuits can be found in Chapter 4 of [70].

Let η be a primitive m -th root of unity. We sometimes call η a *global phase of order m* . We think of these global phases as gates acting on 0 qubits and in what follows we will be especially interested in the global phases of order 2, 4, and 8, which we denote -1 , i , and ζ , respectively. The single-qubit *phase gate of order m* is defined as

$$P_\eta = \begin{bmatrix} 1 & 0 \\ 0 & \eta \end{bmatrix}.$$

We will be particularly interested in phase gates of order 2, 4, and 8 which are the Z , S , and T gates previously defined in Section 1.4.2, respectively. In addition to phase gates, we will also use the single-qubit Hadamard gate H and NOT gate X , likewise described in Section 1.4.2. The last single-qubit gate we will use is the F gate defined below.

$$F = \frac{1}{2} \begin{bmatrix} 1 + \sqrt{-2} & 1 \\ 1 & -1 + \sqrt{-2} \end{bmatrix}.$$

The F gate is not as common as the other single-qubit gates introduced above. We note that $F^2 = iH$ and that F can be expressed as a product of better known gates since

$$F = SHTSHTSHS\zeta^{-1}$$

We will also make use of the two-qubit $H \otimes H$ gate as well as the *controlled* classical gates CX and CCX as defined in Section 1.4.2. Finally we define the *controlled-H* gate

$$CH = \mathbb{1}_2 \oplus H$$

In general, if G is a gate, then we write $C^n G$ for the n -control- G gate.

As usual, *circuits* are built from gates through composition and tensor product. An *ancilla* is a qubit used locally within a circuit but on which the global action of the circuit is trivial. In particular, we say that a unitary matrix W is exactly represented by a circuit D using n ancillas if for any input state $|\psi\rangle$ and ancilla state $|\phi\rangle$ we have

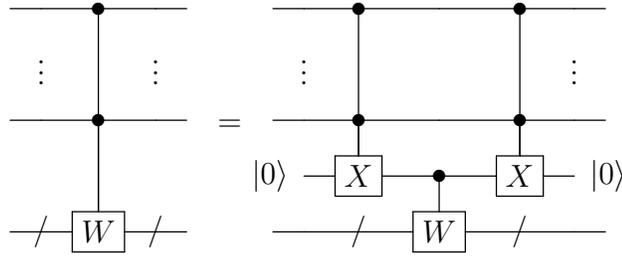
$$D |\psi\rangle |\phi\rangle = (W |\psi\rangle) |\phi\rangle.$$

If $|\phi\rangle = |0\rangle^{\otimes n}$, then the ancillary qubits are said to be *clean*. Without this requirement, the ancillary qubits are said to be *dirty*. Unless otherwise stated, ancillas are assumed to be clean.

In order to characterize restricted Clifford+ T circuits, it is helpful to establish some basic facts about the construction of multi-level matrices over gate sets including the Toffoli gate. It is known (see, e.g., [70, Sec. 4.5.2]) that an n -qubit, 2^m -level matrix of type W can be implemented using the CX gate and the fully-controlled- W gate $C^{n-m}W$. Moreover, if the fully-controlled- X gate can be implemented with one dirty ancilla and the singly-controlled- W gate can be implemented with one dirty ancilla, then the fully-controlled- W gate can be implemented using one clean ancilla.

Lemma 4.4.1. *Let \mathcal{G} be a gate set such that $C^n X$ is representable by a circuit with a single dirty ancilla for any n , and let W be a $2^m \times 2^m$ unitary matrix. If CW is representable over \mathcal{G} with at most one dirty ancilla, then $C^n W$ is also representable over \mathcal{G} for any $n \geq 1$. Moreover, a single ancilla suffices.*

Proof. Follows from standard techniques, e.g. [142]. In particular, if $n = 1$, then CW is implementable with a single dirty, and hence also clean ancilla. If $n > 1$, then $C^n W$ gate can be implemented with the following construction, where each gate on the right has at least one (dirty) ancilla available for use:



□

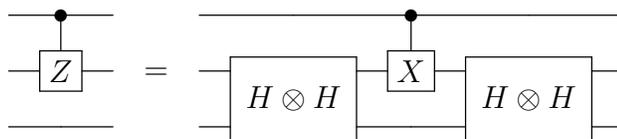
We can now use Lemma 4.4.1 to give constructions of multi-level matrices of different types over their uncontrolled versions in the presence of the Toffoli gate. Recall that the multiply-controlled X gate can be implemented with CCX gates and a single dirty ancilla [142].

Proposition 4.4.2. *The operators*

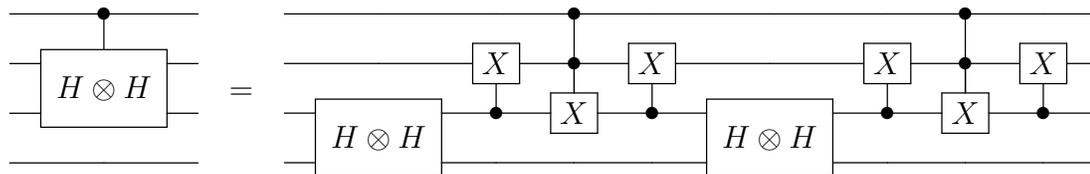
$$\{(-1)_{[a]}, X_{[a,b]}, (H \otimes H)_{[a,b,c,d]}\}$$

where $a, b, c,$ and d are distinct elements of $[n]$ can be exactly represented by quantum circuits over the gate set $\{X, CX, CCX, H \otimes H\}$ using at most one ancilla.

Proof. By Lemma 4.4.1 it suffices to give constructions for the singly-controlled Z and $H \otimes H$ gates. Clearly



and it can be verified that



□

Corollary 4.4.3. *The operators*

$$\{(-1)_{[a]}, X_{[a,b]}, (H \otimes H)_{[a,b,c,d]}, I_{2^{n-1}} \otimes H\}$$

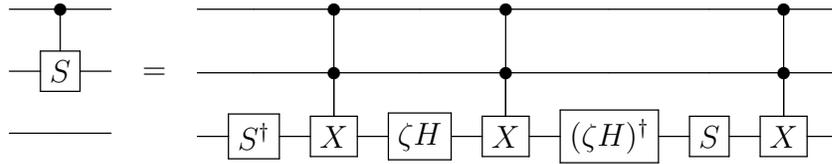
where $a, b, c,$ and d are distinct elements of $[1..n]$ can be exactly represented by quantum circuits over the gate set $\{X, CX, CCX, H\}$ using at most one ancilla.

Proposition 4.4.4. *The operators*

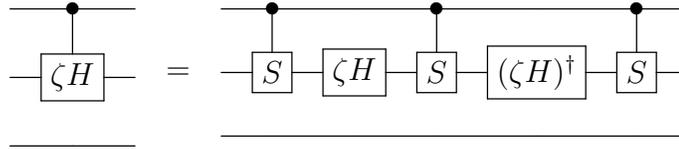
$$\{i_{[a]}, X_{[a,b]}, \zeta H_{[a,b]}\}$$

where a and b are distinct elements of $[n]$ can be exactly represented by quantum circuits over the gate set $\{X, CX, CCX, \zeta H, S\}$ using at most one ancilla.

Proof. Again it suffices to give constructions for the singly-controlled S and ζH gates. In this case it can be verified that



Likewise, we have



□

Corollary 4.4.5. *The operators*

$$\{i_{[a]}, X_{[a,b]}, \zeta H_{[a,b]}, \zeta I\}$$

where a and b are distinct elements of $[1..n]$ can be exactly represented by quantum circuits over the gate set $\{X, CX, CCX, H, S\}$ using at most one ancilla.

Proof. Follows from Proposition 4.4.4 and the fact that $\zeta = SHSHSH$. □

Proposition 4.4.6. *The operators*

$$\{(-1)_{[a]}, X_{[a,b]}, H_{[a,b]}\}$$

where a and b are distinct elements of $[[1..n]]$ can be exactly represented by quantum circuits over the gate set $\{X, CX, CCX, H, CH\}$ using at most one ancilla.

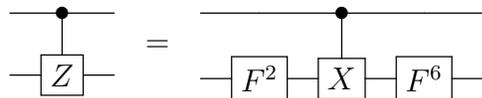
Proof. By Proposition 4.4.2, $(-1)_{[a]}$ can be represented by a quantum circuit over $\{X, CX, CCX, H \otimes H\}$ and hence also $\{X, CX, CCX, H, CH\}$. Since CH is already in the generating set the proof is complete. \square

Proposition 4.4.7. *The operators*

$$\{(-1)_{[a]}, X_{[a,b]}, F_{[a,b]}\}$$

where a and b are distinct elements of $[[1..n]]$ can be exactly represented by quantum circuits over the gate set $\{X, CX, CCX, F\}$ using at most one ancilla.

Proof. To show that CZ is representable over the gate set, it can be observed that since $F^2 = iH$ and $F^6 = -iH$, it follows that



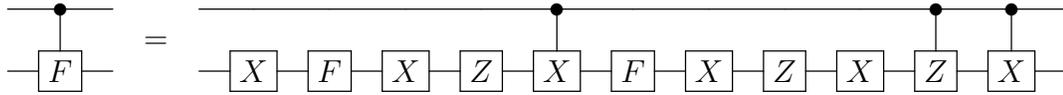
The construction of CF is somewhat more involved, but can be obtained from

standard constructions (e.g., [142]) by noting that

$$(ZXF)^2 = I, \text{ and}$$

$$X(ZXF)X(ZXF)X = ZXF.$$

In particular, the controlled ZXF gate can be constructed by adding a control to the middle X gate above and cancelling the controlled Z and X factors:



□

4.5 Number-Theoretic Characterizations

4.5.1 The \mathbb{D} case

We start by studying the group of $n \times n$ unitary matrices over \mathbb{D} . Since X , CX , CCX , and $H \otimes H$ have entries in \mathbb{D} , any circuit over $\{X, CX, CCX, H \otimes H\}$ must represent a unitary matrix over \mathbb{D} . Here, we show the converse: any unitary matrix over \mathbb{D} can be represented by a circuit over $\{X, CX, CCX, H \otimes H\}$. To prove this, it is sufficient to establish that every unitary over \mathbb{D} can be expressed as a product of the following generators

$$\{(-1)_{[a]}, X_{[a,b]}, (H \otimes H)_{[a,b,c,d]}\}, \quad (4.1)$$

where $a, b, c,$ and d are distinct elements of $[n]$. Indeed, by Proposition 4.4.2, all of the above generators can be exactly represented by quantum circuits over the gate set $\{X, CX, CCX, H \otimes H\}$.

If V is a matrix over \mathbb{D} , then V can be written as

$$V = \frac{1}{2^q} W \tag{4.2}$$

where $q \in \mathbb{N}$ and W is a matrix over \mathbb{Z} . We will consider 2 denominator exponents of such matrices.

The following four lemmas are devoted to proving the analogue of Giles and Selinger’s *Column Lemma* (Lemma 5 in [96]). Here, the goal is to establish that any unit vector over \mathbb{D} can be reduced to a standard basis vector by multiplying it on the left by an appropriately chosen sequence of generators. We consider the case of vectors of dimension $n < 4$ first, before moving on to higher dimensions.

Lemma 4.5.1. *Let $n < 4$ and let $j \in [[1..n]]$. If v is an n -dimensional unit vector over \mathbb{D} , then there exists generators G_1, \dots, G_ℓ from (4.1) such that $G_1 \cdots G_\ell v = e_j$.*

Proof. Write v as $v = u/2^q$ with $u \in \mathbb{Z}^n$ and $q = \text{lde}_2(v)$. Since v is a unit vector, we have $v^\dagger v = 1$ and thus $4^q = \sum u_k^\dagger u_k = \sum u_k^2$. The square of any odd number is congruent to 1 modulo 4. Thus when $n < 4$, we have $\sum u_k^2 \equiv 0 \pmod{4}$ only if every u_k is even. This implies that $\text{lde}_2(v) = 0$ when $n < 4$ and therefore that $v = \pm e_{j'}$ for some $j' \in [[1..n]]$. Hence one of

$$v = e_j, \quad (-1)_{[j]} v = e_j, \quad X_{[j,j']} v = e_j, \quad \text{or} \quad X_{[j,j']} (-1)_{[j']} v = e_j$$

must hold, which completes the proof. \square

Because $(H \otimes H)_{[a,b,c,d]}$ is a four-level matrix, we consider its action on certain 4-dimensional vectors in the lemma below. This is in contrast with Giles and Selinger's algorithm, for which only one- and two-level matrices are needed.

Lemma 4.5.2. *If $u_1, \dots, u_4 \in \mathbb{Z}$ are such that $u_1^2 \equiv \dots \equiv u_4^2 \equiv 1 \pmod{4}$, then there exists m_1, \dots, m_4 such that*

$$(H \otimes H) \begin{pmatrix} (-1)_{[1]}^{m_1} \\ (-1)_{[2]}^{m_2} \\ (-1)_{[3]}^{m_3} \\ (-1)_{[4]}^{m_4} \end{pmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} u'_1 \\ u'_2 \\ u'_3 \\ u'_4 \end{bmatrix}$$

for some $u'_1, \dots, u'_4 \in \mathbb{Z}$ such that $u'_1 \equiv \dots \equiv u'_4 \equiv 0 \pmod{2}$.

Proof. If $u \in \mathbb{Z}$ is such that $u^2 \equiv 1 \pmod{4}$, then $u \equiv 1 \pmod{4}$ or $u \equiv 3 \pmod{4}$. Furthermore, if $u \equiv 3 \pmod{4}$, then $-u \equiv 1 \pmod{4}$. Hence, given $u_1, \dots, u_4 \in \mathbb{Z}$ such that $u_1^2 \equiv \dots \equiv u_4^2 \equiv 1 \pmod{4}$, we can find m_1, \dots, m_4 such that $(-1)^{m_1} u_1 \equiv \dots \equiv (-1)^{m_4} u_4 \equiv 1 \pmod{4}$. It can then be verified that

$$(H \otimes H) \begin{bmatrix} (-1)^{m_1} u_1 \\ (-1)^{m_2} u_2 \\ (-1)^{m_3} u_3 \\ (-1)^{m_4} u_4 \end{bmatrix} = \begin{bmatrix} u'_1 \\ u'_2 \\ u'_3 \\ u'_4 \end{bmatrix}$$

for some $u'_1 \equiv \dots \equiv u'_4 \equiv 0 \pmod{2}$. \square

Lemma 4.5.3. *Let $n \geq 4$. If v is an n -dimensional unit vector over \mathbb{D} and $\text{lde}_2(v) > 0$, then there exists generators G_1, \dots, G_ℓ from (4.1) such that $G_1 \cdots G_\ell v = v'$ and $\text{lde}_2(v') < \text{lde}_2(v)$.*

Proof. Write v as $v = u/2^q$ where $u \in \mathbb{Z}^n$ and $q > 1$. Since v is a unit vector we have $v^\dagger v = 1$ and thus $4^q = \sum u_k^\dagger u_k = \sum u_k^2$ since u is real. The number of u_k such that $u_k^2 \equiv 1 \pmod{4}$ is therefore congruent to 0 modulo 4. Hence, we can group these entries in sets of size 4 and apply Lemma 4.5.2 to each such set in order to reduce the 2 denominator exponent of the vector. \square

Lemma 4.5.4. *Let $j \in [[1..n]]$. If v is an n -dimensional unit vector over \mathbb{D} , then there exists generators G_1, \dots, G_ℓ from (4.1) such that $G_1 \cdots G_\ell v = e_j$.*

Proof. The case of vectors of dimension $n < 4$ was treated in Lemma 4.5.1 so we assume that $n \geq 4$ and we proceed by induction on the least 2 denominator exponent of v .

- If $\text{lde}_2(v) = 0$, then v is a unit vector in \mathbb{Z}^n . Hence $v = \pm e_{j'}$ for some $j' \in [[1..n]]$ and one of

$$v = e_j, \quad (-1)_{[j]} v = e_j, \quad X_{[j,j']} v = e_j, \quad \text{or} \quad X_{[j,j']} (-1)_{[j']} v = e_j$$

must hold.

- If $\text{lde}_2(v) > 0$, apply Lemma 4.5.3 to reduce the 2 denominator exponent of v . \square

We can now use Lemma 4.5.4 to prove that every unitary matrix with entries in \mathbb{D} can be written as a product of generators. This, together with Proposition 4.4.2 establishes our characterization of circuits over the gate set $\{X, CX, CCX, H \otimes H\}$.

Theorem 4.5.5. *If V is an n -dimensional unitary matrix with entries in \mathbb{D} , then there exists generators G_1, \dots, G_ℓ from (4.1) such that $G_1 \cdots G_\ell V = I$.*

Proof. By iteratively applying Lemma 4.5.4 to the columns of V . □

Corollary 4.5.6. *A matrix V can be exactly represented by an n -qubit circuit over $\{X, CX, CCX, H \otimes H\}$ if and only if $V \in \mathcal{U}_{2^n}(\mathbb{D})$. Moreover, a single ancilla always suffices to construct a circuit for V .*

To conclude this case, we leverage Theorem 4.5.5 and Corollary 4.4.3 to characterize circuits over the gate set $\{X, CX, CCX, H\}$. To this end, we consider matrices of the form

$$V = \frac{1}{\sqrt{2}^q} W \tag{4.3}$$

where $q \in \mathbb{N}$ and W is a matrix over \mathbb{Z} . For these matrices, we use $\sqrt{2}$ denominator exponents. We extend the set of generators from (4.1) with a matrix of the form $I \otimes H$. Thus the relevant generators are now

$$\{(-1)_{[a]}, X_{[a,b]}, (H \otimes H)_{[a,b,c,d]}, I_{2^{n-1}} \otimes H\} \tag{4.4}$$

where a, b, c , and d are distinct elements of $[[1..n]]$.

Lemma 4.5.7. *If $V \neq 0$ is as in (4.3), then all the $\sqrt{2}$ denominator exponents of V are congruent modulo 2.*

Proof. Suppose that $q < q'$ are two $\sqrt{2}$ denominator exponents of V . Then $V = W/\sqrt{2}^q = W'/\sqrt{2}^{q'}$ for some integer matrices W and W' . Assume without loss of generality that $q < q'$. Then

$$W' = \sqrt{2}^{q'} V = \sqrt{2}^{q'-q} W$$

so that $\sqrt{2}^{q'-q} W$ is an integer matrix. Hence $q \equiv q' \pmod{2}$, since $V \neq 0$ and $\sqrt{2} \notin \mathbb{Z}$. □

Theorem 4.5.8. *Let n be even. If $V = W/\sqrt{2}^q$ is an n -dimensional unitary matrix such that W is an integer matrix, then there exists generators G_1, \dots, G_ℓ from (4.4) such that $G_1 \cdots G_\ell V = I$.*

Proof. If q is even, the result follows from Theorem 4.5.5. If q is odd, then

$$(I_{2^{n-1}} \otimes H)V = W'/\sqrt{2}^{q'}$$

for some even q' and some integer matrix W' . Hence the result follows by applying Theorem 4.5.5 to $(I_{2^{n-1}} \otimes H)V$. □

Remark 4.5.9. The restriction to even dimensions in Theorem 4.5.8 is not a consequence of the choice of generators. Indeed, it can be shown that there are no unitary matrices of the form (4.3) whose dimension and least $\sqrt{2}$ denominator exponent are both odd [143].

Corollary 4.5.10. *A matrix V can be exactly represented by an n -qubit circuit over $\{X, CX, CCX, H\}$ if and only if V is a 2^n -dimensional unitary matrix such*

$V = W/\sqrt{2}^q$ for some integer matrix W and some $q \in \mathbb{N}$. Moreover, a single ancilla always suffices to construct a circuit for V .

4.5.2 The $\mathbb{D}[\sqrt{2}]$ case

We now focus on the group of $n \times n$ unitary matrices with entries in $\mathbb{D}[\sqrt{2}]$.

The elements of this group can be written as

$$V = \frac{1}{\sqrt{2}^q} W \tag{4.5}$$

where $q \in \mathbb{N}$ and W is a matrix over $\mathbb{Z}[\sqrt{2}]$. We now use $\sqrt{2}$ denominator exponents and the relevant generators are

$$\{(-1)_{[a]}, X_{[a,b]}, H_{[a,b]}\} \tag{4.6}$$

where a and b are distinct elements of $[[1..n]]$. By Proposition 4.4.6, all of the above generators can be exactly represented by quantum circuits over the gate set $\{X, CX, CCX, H, CH\}$. As in the previous cases, we prove our characterization by showing that any unitary matrix of the form (4.5) can be expressed as a product of generators from (4.6).

Lemma 4.5.11. *If $u_1, u_2 \in \mathbb{Z}[\sqrt{2}]$ are such that $u_1 \equiv u_2 \pmod{2}$, then*

$$H \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u'_1 \\ u'_2 \end{bmatrix}$$

for some $u'_1, u'_2 \in \mathbb{Z}[\sqrt{2}]$ such that $u'_1 \equiv u'_2 \equiv 0 \pmod{\sqrt{2}}$.

Proof. Since $u_1 \equiv u_2 \pmod{2}$, we have $u_1 + u_2 \equiv u_1 - u_2 \equiv 0 \pmod{2}$. It can then

be verified that

$$H \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u'_1 \\ u'_2 \end{bmatrix}$$

for some $u'_1 \equiv u'_2 \equiv 0 \pmod{2}$. □

Lemma 4.5.12. *If v is an n -dimensional unit vector over $\mathbb{D}[\sqrt{2}]$ and $\text{lde}_{\sqrt{2}}(v) > 0$, then there exists generators G_1, \dots, G_ℓ from (4.6) such that $G_1 \cdots G_\ell v = v'$ and $\text{lde}_{\sqrt{2}}(v') < \text{lde}_{\sqrt{2}}(v)$.*

Proof. Write v as $v = u/\sqrt{2}^q$ where $u \in \mathbb{Z}[\sqrt{2}]$ and $q > 0$. Since v is a unit vector we have $v^\dagger v = 1$ and thus $2^q = \sum u_j^\dagger u_j = \sum u_j^2$ since u is real. Letting $u_j = x_j + y_j\sqrt{2}$, this yields the following equation

$$2^q = \sum x_j^2 + 2y_j^2 + x_j y_j 2\sqrt{2}.$$

Thus $\sum x_j^2 \equiv 0 \pmod{2}$ and $\sum x_j y_j = 0$. It follows that $u_j \equiv 1 \pmod{2}$ for evenly many j and $u_j \equiv 1 + \sqrt{2} \pmod{2}$ for evenly many j . We can therefore group these entries in sets of size 2 and apply Lemma 4.5.11 to each such set in order to reduce the $\sqrt{2}$ denominator exponent of the vector. □

The following three statements are established like the corresponding ones in the previous section. For this reason, we omit their proofs.

Lemma 4.5.13. *Let $j \in [[1..n]]$. If v is an n -dimensional unit vector over $\mathbb{D}[\sqrt{2}]$, then there exists generators G_1, \dots, G_ℓ from (4.6) such that $G_1 \cdots G_\ell v = e_j$.*

Theorem 4.5.14. *If V is an n -dimensional unitary matrix with entries in $\mathbb{D}[\sqrt{2}]$, then there exists generators G_1, \dots, G_ℓ from (4.6) such that $G_1 \cdots G_\ell V = I$.*

Corollary 4.5.15. *A matrix V can be exactly represented by an n -qubit circuit over $\{X, CX, CCX, H, CH\}$ if and only if $V \in \mathcal{U}_{2^n}(\mathbb{D}[\sqrt{2}])$. Moreover, a single ancilla always suffices to construct a circuit for V .*

4.5.3 The $\mathbb{D}[\sqrt{-2}]$ case

We now consider the group of $n \times n$ unitary matrices with entries in $\mathbb{D}[\sqrt{-2}]$.

Such matrices can be written as

$$V = \frac{1}{(\sqrt{-2})^q} W \quad (4.7)$$

where $q \in \mathbb{N}$ and W is a matrix over $\mathbb{Z}[\sqrt{-2}]$. We now use $\sqrt{-2}$ denominator exponents and the relevant generators are

$$\{(-1)_{[a]}, X_{[a,b]}, F_{[a,b]}\} \quad (4.8)$$

where a and b are distinct elements of $[[1..n]]$. By Proposition 4.4.7, all of the above generators can be exactly represented by quantum circuits over $\{X, CX, CCX, F\}$.

As in the previous cases, we establish our characterization by showing that any unitary matrix of the form (4.7) can be expressed as a product of generators from

(4.8).

Lemma 4.5.16. *If $u_1, u_2 \in \mathbb{Z}[\sqrt{-2}]$ are such that $u_1^\dagger u_1 \equiv u_2^\dagger u_2 \equiv 1 \pmod{2}$, then there exists m_0, m_1, m_2 , and m_3 such that*

$$F^{m_0} (-1)_{[1]}^{m_1} (-1)_{[2]}^{m_2} X^{m_3} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u'_1 \\ u'_2 \end{bmatrix}$$

for some $u'_1, u'_2 \in \mathbb{Z}[\sqrt{-2}]$ such that $u'_1 \equiv u'_2 \equiv 0 \pmod{\sqrt{-2}}$.

Proof. First consider the case in which $u_1 \equiv u_2 \pmod{2}$. Then $u_1 + u_2 \equiv u_1 - u_2 \equiv 0 \pmod{2}$ and it can be verified that

$$F^2 \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = iH \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u'_1 \\ u'_2 \end{bmatrix}$$

for some $u'_1 \equiv u'_2 \equiv 0 \pmod{\sqrt{-2}}$. We now consider the case in which $u_1 \not\equiv u_2 \pmod{2}$. In this case, the fact that $u_1^\dagger u_1 \equiv u_2^\dagger u_2 \equiv 1 \pmod{2}$ implies that one of u_1 or u_2 is congruent to 1 or 3 modulo $2\sqrt{-2}$ while the other is congruent to $(1 + \sqrt{-2})$ or $(3 + \sqrt{-2})$ modulo $2\sqrt{-2}$. We can therefore find m_1, m_2, m_3 such that

$$(-1)_{[1]}^{m_1} (-1)_{[2]}^{m_2} X^{m_3} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u''_1 \\ u''_2 \end{bmatrix}$$

where $u_1'' \equiv 1 + \sqrt{-2} \pmod{2\sqrt{-2}}$ and $u_2'' \equiv 1 \pmod{2\sqrt{-2}}$. Then

$$F \begin{bmatrix} u_1'' \\ u_2'' \end{bmatrix} = \frac{1}{2} \begin{bmatrix} (1 + \sqrt{-2})u_1'' + u_2'' \\ u_1'' + (-1 + \sqrt{-2})u_2'' \end{bmatrix}.$$

But $u_1'' \equiv 1 + \sqrt{-2} \pmod{2\sqrt{-2}}$ and $u_2'' \equiv 1 \pmod{2\sqrt{-2}}$ so that

$$(1 + \sqrt{-2})u_1'' + u_2'' \equiv (1 + \sqrt{-2})^2 + 1 \equiv 2\sqrt{-2} \equiv 0 \pmod{2\sqrt{-2}}.$$

and

$$u_1'' + (-1 + \sqrt{-2})u_2'' \equiv (1 + \sqrt{-2}) + (-1 + \sqrt{-2}) \equiv 2\sqrt{-2} \equiv 0 \pmod{2\sqrt{-2}}.$$

Hence we can set $u_1' = ((1 + \sqrt{-2})u_1'' + u_2'')/2$ and $u_2' = (u_1'' + (-1 + \sqrt{-2})u_2'')/2$ to complete the proof. \square

Lemma 4.5.17. *If v is an n -dimensional unit vector over $\mathbb{D}[\sqrt{-2}]$ and $\text{lde}_{\sqrt{-2}}(v) > 0$, then there exists generators G_1, \dots, G_ℓ from (4.8) such that $G_1 \cdots G_\ell v = v'$ and $\text{lde}_{\sqrt{-2}}(v') < \text{lde}_{\sqrt{-2}}(v)$.*

Proof. Write v as $v = u/\sqrt{-2}^q$ where $u \in \mathbb{Z}[\sqrt{-2}]$ and $q > 0$. Since v is a unit vector we have $v^\dagger v = 1$ and thus $(-2)^q = \sum u_j^\dagger u_j$. Thus $\sum u_j^\dagger u_j \equiv 0 \pmod{2}$ and it follows that $u_j^\dagger u_j \equiv 1 \pmod{2}$ for evenly many j , since modulo 2 we have $u_j^\dagger u_j \equiv 0$ or $u_j^\dagger u_j \equiv 1$. We can therefore group these entries in sets of size 2 and apply Lemma 4.5.16 to each such set in order to reduce the denominator exponent. \square

Lemma 4.5.18. *Let $j \in [[1..n]]$. If v is an n -dimensional unit vector over $\mathbb{D}[\sqrt{-2}]$, then there exists generators G_1, \dots, G_ℓ from (4.8) such that $G_1 \cdots G_\ell v = e_j$.*

Theorem 4.5.19. *If V is an n -dimensional unitary matrix with entries in $\mathbb{D}[\sqrt{-2}]$, then there exists generators G_1, \dots, G_ℓ from (4.8) such that $G_1 \cdots G_\ell V = I$.*

Corollary 4.5.20. *A matrix V can be exactly represented by an n -qubit circuit over $\{X, CX, CCX, F\}$ if and only if $V \in \mathcal{U}_{2^n}(\mathbb{D}[\sqrt{-2}])$. Moreover, a single ancilla always suffices to construct a circuit for V .*

4.5.4 The $\mathbb{D}[i]$ case

Finally, we turn our attention to the group of $n \times n$ unitary matrices with entries in $\mathbb{D}[i]$. The relevant set of generators is

$$\{i_{[a]}, X_{[a,b]}, \omega H_{[a,b]}\} \quad (4.9)$$

where a and b are distinct elements of $[[1..n]]$. We reason as in the previous cases, noting by Proposition 4.4.4 that all of the above generators can be exactly represented by quantum circuits over $\{X, CX, CCX, \omega H, S\}$.

If V is a matrix over $\mathbb{D}[i]$, then V can be written as $V = W/2^q$ where $q \in \mathbb{N}$ and W is a matrix over $\mathbb{Z}[i]$. For our purposes, however, it is more convenient to express these matrices as

$$V = \frac{1}{(1+i)^q} W \quad (4.10)$$

where $q \in \mathbb{N}$ and W is a matrix over $\mathbb{Z}[i]$. This is equivalent since

$$\frac{1}{2^q}W = \frac{i^q}{(1+i)^{2q}}W = \frac{1}{(1+i)^{2q}}W'.$$

We therefore use matrices of the form (4.10) and use $(1+i)$ denominator exponents.

Lemma 4.5.21. *If $u_1, u_2 \in \mathbb{Z}[i]$ are such that $u_1^2 \equiv u_2^2 \equiv 1 \pmod{2}$, then there exists m_1 and m_2 such that*

$$\omega H_{\begin{smallmatrix} [1] \\ [2] \end{smallmatrix}}^{i^{m_1} i^{m_2}} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u'_1 \\ u'_2 \end{bmatrix}$$

for some $u'_1, u'_2 \in \mathbb{Z}[i]$ such that $u'_1 \equiv u'_2 \equiv 0 \pmod{1+i}$.

Proof. If $u^2 \equiv 1 \pmod{2}$, then $u \equiv 1 \pmod{2}$ or $u \equiv i \pmod{2}$. Furthermore, if $u \equiv i \pmod{2}$, then $iu \equiv 1 \pmod{2}$. Hence, given $u_1, u_2 \in \mathbb{Z}$ such that $u_1^2 \equiv u_2^2 \equiv 1 \pmod{2}$, we can find m_1 and m_2 such that $i^{m_1}u_1 \equiv i^{m_2}u_2 \equiv 1 \pmod{2}$. It can then be verified that

$$\omega H_{\begin{smallmatrix} [1] \\ [2] \end{smallmatrix}}^{i^{m_1} i^{m_2}} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u'_1 \\ u'_2 \end{bmatrix}$$

for some $u'_1 \equiv u'_2 \equiv 0 \pmod{1+i}$. □

Lemma 4.5.22. *If v is an n -dimensional unit vector over $\mathbb{D}[i]$ and $\text{lde}_{(1+i)}(v) > 0$, then there exists generators G_1, \dots, G_ℓ from (4.9) such that $G_1 \cdots G_\ell v = v'$ and $\text{lde}_{(1+i)}(v') < \text{lde}_{(1+i)}(v)$.*

Proof. Write v as $v = u/(1+i)^q$ where $u \in \mathbb{Z}[i]$ and $q > 1$. Since $(1+i)^\dagger(1+i) = 2$

and v is a unit vector, we have $2^q = \sum u_j^\dagger u_j$. Thus $0 \equiv \sum u_j^\dagger u_j \equiv \sum u_j^2 \pmod{2}$ and it follows that $u_j^2 \equiv 1 \pmod{2}$ for evenly many j . We can therefore group these entries in sets of size 2 and apply Lemma 4.5.21 to each such set in order to reduce the denominator exponent. \square

Lemma 4.5.23. *Let $j \in [[1..n]]$. If v is an n -dimensional unit vector over $\mathbb{D}[i]$, then there exists generators G_1, \dots, G_ℓ from (4.9) such that $G_1 \cdots G_\ell v = e_j$.*

Theorem 4.5.24. *If V is an n -dimensional unitary matrix with entries in $\mathbb{D}[i]$, then there exists generators G_1, \dots, G_ℓ from (4.9) such that $G_1 \cdots G_\ell V = I$.*

Corollary 4.5.25. *A matrix V can be exactly represented by an n -qubit circuit over $\{X, CX, CCX, \omega H, S\}$ if and only if $V \in \mathcal{U}_{2^n}(\mathbb{D}[i])$. Moreover, a single ancilla always suffices to construct a circuit for V .*

Corollary 4.5.25 characterizes circuits over the gate set $\{X, CX, CCX, \omega H, S\}$.

We now use this result, together with Corollary 4.4.5 to characterize circuits over the gate set $\{X, CX, CCX, H, S\}$. To this end, we consider matrices of the form

$$V = \frac{1}{\sqrt{2}^q} W \tag{4.11}$$

where $q \in \mathbb{N}$ and W is a matrix over $\mathbb{Z}[i]$. We use the $\sqrt{2}$ denominator exponents of such matrices and, as in Section 4.5.1, we make use of the fact that $\sqrt{2} \notin \mathbb{Z}[i]$.

The relevant generators are now

$$\{i_{[a]}, X_{[a,b]}, \omega H_{[a,b]}, \omega I_n\}. \tag{4.12}$$

Lemma 4.5.26. *If $V \neq 0$ is as in (4.11), then all the denominator exponents of V are congruent modulo 2.*

Proof. Similar to the proof of Lemma 4.5.7. □

Theorem 4.5.27. *If $V = W/\sqrt{2}^q$ is an n -dimensional unitary matrix such that W is a matrix over $\mathbb{Z}[i]$, then there exists generators G_1, \dots, G_ℓ from (4.12) such that $G_1 \cdots G_\ell V = I$.*

Proof. If q is even, the result follows from Theorem 4.5.24. If q is odd, then

$$(\omega I_n)V = W'/\sqrt{2}^{q'}$$

for some even q' and some $W' \in \mathbb{Z}[i]^{n \times n}$. Hence the result follows by applying Theorem 4.5.24 to $(I \otimes H)V$. □

Corollary 4.5.28. *A matrix V can be exactly represented by an n -qubit circuit over $\{X, CX, CCX, H, S\}$ if and only if V is a 2^n -dimensional unitary matrix such $V = W/\sqrt{2}^q$ for some matrix W over $\mathbb{Z}[i]$ and some $q \in \mathbb{N}$. Moreover, a single ancilla always suffices to construct a circuit for V .*

4.6 Conclusion

In this chapter, we provided number-theoretic characterizations for several classes of restricted but universal Clifford+ T circuits, focusing on integral, real, imaginary, and Gaussian circuits. We showed that a unitary matrix can be exactly represented by an n -qubit integral Clifford+ T circuit if and only if it is an element

of the group $\mathcal{U}_{2^n}(\mathbb{D})$. We then established that real, imaginary, and Gaussian circuits similarly correspond to the groups $\mathcal{U}_{2^n}(\mathbb{D}[\sqrt{2}])$, $\mathcal{U}_{2^n}(\mathbb{D}[\sqrt{-2}])$, and $\mathcal{U}_{2^n}(\mathbb{D}[i])$, respectively.

One avenue for future research is to improve the performance, in runtime or gate count, of the algorithms introduced in the present paper. Further afield, it would be interesting to study restricted Clifford+ T circuits in the context of fault-tolerance, randomized benchmarking, or simulation. We hope that our characterizations will help deepen our understanding of Clifford+ T circuits, restricted or not. Given that we now have a complete characterization of some relatively simple universal two-qubit gate sets, we return our attention to searching for compiling algorithms in one of these universal multi-qubit gate sets – the $\mathcal{C} + CS$ gate set, which is equivalent to $\mathcal{C} + CCX$.

Chapter 5: Clifford + Controlled Phase Exact Synthesis¹

5.1 Introduction

In the current paradigm of fault-tolerant quantum computing [48] wherein we use both quantum error correction and magic-state distillation, it is generally accepted that non-Clifford gate count is the best “simple” metric by which to define the cost of a quantum computation. Using algebraic and number-theoretic methods, provably optimal compiling algorithms which could handily outperform the Solovay-Kitaev algorithm for this cost metric were developed first for the single qubit Clifford+T gate set [90,97–100] and then for other single-qubit gate sets [93–95,113,114] as well as some single-qutrit gate sets [112,116,121–123]. Forays into extending these results [91,92,124] to the multi-qubit compiling problem have born fruit in the form of heuristic algorithms [67,109] and compilers for restricted gate sets [106], but thus far truly optimal algorithms for universal multi-qubit gate sets [145] have remained elusive, even in the limited two-qubit circuit case [129,146]. Such optimal small-qubit-number compilers could be a boon to not only smaller near-term devices, but also long term in applications such as Hamiltonian simulation [28] where many protocols call for cascading rounds of few-qubit-number sub-circuits.

¹This chapter is a slightly modified version of a forthcoming manuscript [144]

In Chapter 5, we introduce the first optimal compiling algorithm for a universal multi-qubit gate set. Restricting to two-qubit circuits, we provide a normal form for Clifford + controlled-Phase gate CS circuits, a universal gate set that is a strict subset of the Clifford+T gate set. After introducing some notation and definitions in Section 5.2, we then constructively prove in Section 5.3 that every distinct normal form is unique and develop a linear-time algorithm to synthesize any 4×4 unitary corresponding to a Clifford+ CS circuit into its equivalent normal form. Finally, we prove optimality, give a full presentation of the Clifford + Controlled Phase gate group, and comment on some lower bounds of the gate count for the inexact synthesis problem in Section 5.4 before concluding in Section 5.5.

5.2 Generators

We will be considering the Pauli and Clifford groups on two qubits, \mathcal{P}_2 and \mathcal{C}_2 respectively, defined in Section 1.4.2. The Clifford group is well-suited for fault-tolerant quantum computation but is not universal. One may obtain a universal group on two qubits by extending \mathcal{C}_2 with the *controlled-Phase operator* CS defined

as

$$CS = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}.$$

We also note that $CS^2 = CZ \in \mathcal{C}_2$. In what follows, we focus on the group $\mathcal{G} = \mathcal{C}_2 + CS$ of operators which can be represented by a two-qubit circuit over the extended Clifford gate set $\{H, S, CX, CS\}$. Equivalently, $\mathcal{C}_2 + CS$ is the group generated by $H \otimes I, I \otimes H, S \otimes I, I \otimes S, CX_{1:2}, CX_{2:1}$ and CS . From our work in Chapter 4, we know that these unitaries are equivalent to the group of 4×4 unitaries which can be expressed as

$$\frac{1}{\sqrt{2}^k} M$$

for $k \in \mathbb{N}$ and $M \in \mathcal{M}_{4 \times 4}(\mathbb{Z}[i])$.

We now introduce a generalization of the CS gate which will be helpful in describing the elements of \mathcal{G} .

Definition 5.2.1. Let P and Q be distinct elements of $\mathcal{P}_2 \setminus \{\mathbb{1}\}$ such that P and Q are Hermitian and $[P, Q] = 0$. Then $R(P, Q)$ is the operator defined as

$$R(P, Q) = \exp\left(\frac{i\pi}{8} (\mathbb{1} - P - Q + PQ)\right) = \exp\left(\frac{i\pi}{2} \left(\frac{\mathbb{1} - P}{2}\right) \left(\frac{\mathbb{1} - Q}{2}\right)\right).$$

Note that $R(Z \otimes \mathbb{1}, \mathbb{1} \otimes Z) = CS$. The fact that \mathcal{C}_2 is the normalizer of \mathcal{P}_2 then implies that every $R(P, Q)$ for is an element of $\mathcal{C}_2 + CS$. In total, there are 180 $R(P, Q)$ operators, as there are 30 valid choices for P and 6 for Q given a fixed P .

Remark 5.2.2. We can rewrite $R(P, Q)$ as the following sum over Paulis:

$$R(P, Q) = \mathbb{1} + (i - 1) \left(\frac{\mathbb{1} - P}{2}\right) \left(\frac{\mathbb{1} - Q}{2}\right).$$

This may be computed by first recognizing that two operators enclosed in parentheses in the exponent of Definition 5.2.1 are idempotent. Matrix exponentiation is then straightforward.

We now introduce some basic relations that hold for $R(P, Q)$.

Lemma 5.2.3. *Let $C \in \mathcal{C}_2$ and let $P, Q,$ and L be distinct elements of $\mathcal{P}_2 \setminus \{I\}$. Assume that $P, Q,$ and L are Hermitian, that $[P, Q] = [P, L] = 0$ and that $QL = -LQ$. Then the following relations hold:*

$$CR(P, Q) = R(CPC^\dagger, CQC^\dagger)C \quad (5.1)$$

$$R(Q, P) = R(P, Q) \quad (5.2)$$

$$R(P, -PQ) = R(P, Q) \quad (5.3)$$

$$R(P, -Q) \in R(P, Q)\mathcal{C}_2 \quad (5.4)$$

$$R(P, Q)R(P, Q) \in \mathcal{C}_2 \quad (5.5)$$

$$R(P, L)R(P, Q) = R(P, Q)R(P, iQL) \quad (5.6)$$

Proof. Eq. (5.1) holds as we have Clifford C such that $C^\dagger C = \mathbb{1}$ which implies

$$\begin{aligned} CR(P, Q) &= C \exp\left(\frac{i\pi}{2} \left(\frac{\mathbb{1} - P}{2}\right) \left(\frac{\mathbb{1} - Q}{2}\right)\right) C^\dagger C \\ &= \exp\left(\frac{i\pi}{2} C \left(\frac{\mathbb{1} - P}{2}\right) C^\dagger C \left(\frac{\mathbb{1} - Q}{2}\right) C^\dagger\right) C \\ &= \exp\left(\frac{i\pi}{2} \left(\frac{\mathbb{1} - CPC^\dagger}{2}\right) \left(\frac{\mathbb{1} - CQC^\dagger}{2}\right)\right) C \\ &= R(CPC^\dagger, CQC^\dagger)C \end{aligned}$$

where $[CPC^\dagger, CQC^\dagger] = 0$ and both CPC^\dagger and CQC^\dagger are non-identity Hermitian Paulis. Eq. (5.2) holds as $PQ = QP$ by definition. We have explicitly

$$\begin{aligned} R(P, -PQ) &= \exp\left(\frac{i\pi}{8} (\mathbb{1} - P - (-PQ) + (P)(-PQ))\right) \\ &= \exp\left(\frac{i\pi}{8} (\mathbb{1} - P - Q + PQ)\right) = R(P, Q) \end{aligned}$$

which proves Eq. (5.3). To show Eq. (5.4), we have

$$\begin{aligned} R(P, -Q) &= \exp\left(\frac{i\pi}{8} (\mathbb{1} - P - (-Q) + (P)(-Q))\right) \\ &= \exp\left(\frac{i\pi}{8} (\mathbb{1} - P - Q + PQ)\right) \exp\left(\frac{i\pi}{4} (Q - PQ)\right) \end{aligned} \quad (5.7)$$

We can always find a Clifford C that maps $-Z \otimes \mathbb{1}$ and $\mathbb{1} \otimes Z$ to Q and PQ under conjugation, and so we have from Eq. (5.7)

$$\begin{aligned} R(P, -Q) &= \exp\left(\frac{i\pi}{8} (\mathbb{1} - P - Q + PQ)\right) C \exp\left(-\frac{i\pi}{4} (Z \otimes \mathbb{1} + \mathbb{1} \otimes Z)\right) C^\dagger \\ &= \exp\left(\frac{i\pi}{8} (\mathbb{1} - P - Q + PQ)\right) C(-iS \otimes S)C^\dagger \\ &\subset R(P, Q)C \end{aligned}$$

Given that we can again always find some C that maps $Z \otimes \mathbb{1}$ and $\mathbb{1} \otimes Z$ to P and

Q , we directly compute Eq. (5.5) as

$$\begin{aligned} R(P, Q)^2 &= CR(Z \otimes \mathbb{1}, \mathbb{1} \otimes Z)^2 C^\dagger \\ &= C(CZ)C^\dagger \in \mathcal{C}. \end{aligned}$$

Finally, consider Eq. (5.6). We first compute using Remark 5.2.2

$$\begin{aligned} &R(P, L)R(P, Q) \\ &= \left[\mathbb{1} + (i-1) \left(\frac{\mathbb{1}-P}{2} \right) \left(\frac{\mathbb{1}-L}{2} \right) \right] \left[\mathbb{1} + (i-1) \left(\frac{\mathbb{1}-P}{2} \right) \left(\frac{\mathbb{1}-Q}{2} \right) \right] \\ &= \mathbb{1} + (i-1) \left(\frac{\mathbb{1}-P}{2} \right) \left[\frac{3+i}{4} \mathbb{1} - \frac{1+i}{4} L - \frac{1+i}{4} Q - \frac{1+i}{4} (-iLQ) \right]. \end{aligned} \quad (5.8)$$

Under the transformation $L \rightarrow Q$ and $Q \rightarrow iQL$ we have

$$(-iLQ) \rightarrow (-i)(Q)(iQL) = L.$$

Applying these to Eq. (5.8), we have

$$R(P, Q)R(P, iQL) = R(P, L)R(P, Q)$$

which completes the proof. □

Using Eqs. (5.2) to (5.4) we can reduce the number of $R(P, Q)$ operators we

need to consider by applying transformations of the form

$$R(P, Q) = R(P', Q')C$$

for C a Clifford. Eq. (5.2) yields a factor of two reduction, Eq. (5.3) a factor of three, and Eq. (5.4) a factor of two for a grand total of a factor of 12 reduction. Applying these operations, we come to a set of $180/12 = 15$ generators to consider:

Definition 5.2.4. The set \mathcal{S} is defined as

$$\begin{aligned} \mathcal{S} = \{ & R(X \otimes I, I \otimes X), R(Y \otimes I, I \otimes Y), R(Z \otimes I, I \otimes Z), \\ & R(Y \otimes I, I \otimes Z), R(Z \otimes I, I \otimes Y), R(Z \otimes I, I \otimes X), \\ & R(X \otimes I, I \otimes Z), R(X \otimes I, I \otimes Y), R(Y \otimes I, I \otimes X), \\ & R(X \otimes X, Y \otimes Y), R(X \otimes X, Z \otimes Y), R(Z \otimes X, Y \otimes Y), \\ & R(Y \otimes X, X \otimes Y), R(Z \otimes X, X \otimes Y), R(Y \otimes X, Z \otimes Y)\}. \end{aligned}$$

We consider this set lexicographically ordered as written.

Definition 5.2.5. The sequence (\mathcal{S}) is the sequence of the elements of set \mathcal{S} in lexicographical ordering. The j th element of this sequence is \mathcal{S}_j .

Proposition 5.2.6. Let $V \in \mathcal{G}$. Then $V = R_1 \cdots R_n C$ where $C \in \mathcal{C}$, and $R_j = R(P_j, Q_j) \in \mathcal{S}$ for $j \in [[1..n]]$.

Proof. Let $V \in \mathcal{G}$. Then V can be written as a product of the form

$$V = C_1 \cdot CS \cdot C_2 \cdot CS \cdot \dots \cdot C_n \cdot CS \cdot C_{n+1}$$

where $C_j \in \mathcal{C}$ for $j \in [[1..n + 1]]$. Since $CS = R(Z \otimes I, I \otimes Z)$ we have

$$V = C_1 \cdot R(Z \otimes I, I \otimes Z) \cdot C_2 \cdot R(Z \otimes I, I \otimes Z) \cdot \dots \cdot C_n \cdot R(Z \otimes I, I \otimes Z) \cdot C_{n+1}. \quad (5.9)$$

The result follows from repeated iterations of a single application of Eq. (5.1) followed by any of Eqs. (5.2) to (5.4) to ensure $R_j \rightarrow R'_j \in \mathcal{S}$. \square

5.3 Exact Synthesis

In this section, we leverage the exceptional isomorphism $SU(4) \cong Spin(6)$ (itself a double-cover of $SO(6)$) to find optimal decompositions for the elements of $\mathcal{C}_2 + CS$. We first describe explicitly the transformation from $SU(4)$ to $SO(6)$.

Consider some $U \in SU(4)$. U induces a transformation of a vector $v \in \mathbb{C}_4$ that preserves $|v|$ and acts through multiplication Uv . Let $\{e_j\}$ be the standard orthonormal basis of \mathbb{C}^4 . From this basis, we shall construct an alternative six-component basis through the *wedge product* \wedge :

Definition 5.3.1 (Wedge product). Let $a \wedge b$ be defined as the *wedge product* of a and b . Wedge products have the following properties given vectors $\vec{a}, \vec{b}, \vec{c} \in \mathbb{C}^n$ and $\alpha, \beta \in \mathbb{C}$:

- Anticommutivity: $a \wedge b = -b \wedge a$

- Associativity: $(a \wedge b) \wedge c = a \wedge (b \wedge c)$
- Bilinearity: $(\alpha a + \beta b) \wedge c = \alpha(a \wedge c) + \beta(b \wedge c)$

Note that the anticommutation of wedge products implies $a \wedge a = 0$. We say $v_1 \wedge \cdots \wedge v_k \in \bigwedge^k \mathbb{C}^n$ for $v_j \in \mathbb{C}^n$. To compute the inner product of two wedge products $v_1 \wedge \cdots \wedge v_k$ and $w_1 \wedge \cdots \wedge w_k$, we compute

$$\langle v_1 \wedge \cdots \wedge v_k, w_1 \wedge \cdots \wedge w_k \rangle = \det(\langle v_q, w_r \rangle)$$

where $\langle v_q, w_r \rangle$ is the entry in the q th row and r th column of a $k \times k$ matrix.

Remark 5.3.2. The magnitude of a wedge product of n vectors can be thought of as the n dimensional volume of the parallelotope (the generalization of a parallelepiped) constructed from those vectors. The orientation of the wedge product defines the direction of circulation around that parallelotope by those vectors.

The wedge product of two vectors in \mathbb{C}^4 can be decomposed into a six-component basis as anticommutativity reduces the 16 potential wedge products of vectors of $\{e_j\}$ to six. We choose this basis as

$$B = \{s_{-,12,34}, s_{+,12,34}, s_{-,23,14}, s_{+,24,13}, s_{-,24,13}, s_{+,23,14}\} \quad (5.10)$$

where we have defined

$$s_{\pm,ij,kl} = \frac{i^{\frac{1\mp 1}{2}}}{\sqrt{2}} (e_i \wedge e_j \pm e_k \wedge e_l). \quad (5.11)$$

We note that B is an orthonormal basis, and define the sequence (B) as the entries of B ordered as in Eq. (5.10). Finally, to compute our new representation of U we need to define how U transforms vectors of both bases:

Definition 5.3.3. Let $U \in \text{SU}(4)$ and \hat{U} be its representation in the transformed basis. Let $v, w \in \mathbb{C}^4$ with $v \wedge w \in \wedge^2 \mathbb{C}^4$. Then the actions of U and \hat{U} are related by

$$\hat{U}(v \wedge w) = (Uv) \wedge (Uw).$$

Finally, we are equipped to define the transformation from $\text{SU}(4)$ to $\text{SO}(6)$:

Definition 5.3.4. Let $U \in \text{SU}(4)$ and $j, k \in [[1..6]]$. Then the entry in the j th row and k th column of the $\text{SO}(6)$ representation of U , \hat{U} is

$$\hat{U}_{j,k} = \langle B_j, \hat{U} B_k \rangle \tag{5.12}$$

where B_j is the j th component of the sequence defined for set B . The action of \hat{U} on B_k is defined by Definitions 5.3.1 and 5.3.3, and the inner product is defined in Definition 5.3.1.

Remark 5.3.5. The fact that this isomorphism yields special orthogonal operators is ultimately due to the fact that the Dynkin diagrams for the Lie algebras of $\text{SU}(4)$, $\text{Spin}(6)$, and $\text{SO}(6)$ are equivalent. However, this fact can be easily illustrated through the Euler decomposition [103] of $\text{SU}(4)$. Direct calculation of \hat{U} for the

operator

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & \alpha^* \end{bmatrix}$$

for $|\alpha| = 1$ and $\alpha = r + ic$ with $r, c \in \mathbb{R}$ yields

$$\hat{U} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & r & 0 & 0 & c \\ 0 & 0 & 0 & r & c & 0 \\ 0 & 0 & 0 & -c & r & 0 \\ 0 & 0 & -c & 0 & 0 & r \end{bmatrix}$$

which is explicitly in $\text{SO}(6)$. Computation of the other 14 Euler angle rotations required for $\text{SU}(4)$ parameterization yields similar matrices, likewise in $\text{SO}(6)$. As $\text{SO}(6)$ forms a group under multiplication, the isomorphism applied to *any* $U \in \text{SU}(4)$ yields $\hat{U} \in \text{SO}(6)$.

Explicitly calculating our $\text{SO}(6)$ representations for the generators of $\mathcal{C}_2 + CS$,

we have

$$\begin{aligned}
\widehat{(\zeta^\dagger S) \otimes \mathbb{1}} &= \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, & \widehat{(iH) \otimes \mathbb{1}} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \\
\widehat{\mathbb{1} \otimes (\zeta^\dagger S)} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, & \widehat{\mathbb{1} \otimes (iH)} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \\
\widehat{\zeta^\dagger CZ} &= \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, & \widehat{\sqrt{\zeta^\dagger} CS} &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned}$$

Note that we have multiplied by overall phase factors to ensure that each operator has determinant one, and furthermore that single-qubit operators have

determinant one on their single-qubit subspace. We now study the $SO(6)$ representation of $\mathcal{C}_2 + CS$ operators, which we denote $\hat{\mathcal{C}}_2 + \widehat{CS}$ in further details. In general, when referring to gates or their $SO(6)$ representation, we will not explicitly write any overall phase for readability.

Definition 5.3.6. The group $\mathcal{D} \leq SO(6)$ is defined as

$$\mathcal{D} = \left\{ (1/\sqrt{2})^k M \mid k \in \mathbb{N}, M \in \mathbb{Z}^{6 \times 6} \right\}.$$

Remark 5.3.7. We will commonly use denominator exponents of $\sqrt{2}$ throughout Chapter 5. These are well-defined as we will always be considering tensors whose entries belong to a ring R such that \mathbb{Z} is a subring of R . As $\sqrt{2}^2 \in \mathbb{Z}$, we can always find the lde of a tensor \mathcal{T} , which we denote $\text{lde}(\mathcal{T})$.

Definition 5.3.8 (*k*-parity). Let V be an $n \times m$ matrix of the form $V = (1/\sqrt{2})^k M$ for some $k \in \mathbb{N}$ and some $M \in \mathbb{Z}^{n \times m}$ and let ℓ be a denominator exponent of V . Then the matrix $\rho_\ell(V) \in \mathbb{Z}_2^{n \times m}$ is defined as

$$\rho_\ell(V)_{i,j} = (\sqrt{2}^\ell V)_{i,j} \pmod{2}.$$

Note that Remark 5.3.7 and Definition 5.3.8 apply to any matrix, vector, or scalar.

Lemma 5.3.9. *We have $\hat{\mathcal{C}}_2 + \widehat{CS} \leq \mathcal{D}$. Moreover, if V is a $\mathcal{C}_2 + CS$ circuit with k CS gates then $\text{lde}(\hat{V}) \leq k$.*

Proof. We have $\hat{\mathcal{C}} + \widehat{CS} \leq \mathcal{D}$ since the images of every generator belong to \mathcal{D} .

Furthermore, the lde of \hat{V} is at most k because there are k factors of $1/\sqrt{2}$ from the k \widehat{CS} gates. \square

Lemma 5.3.10. *We have $\hat{\mathcal{C}}_2 = \{\hat{V} \in \mathcal{D} \mid \text{lde}(\hat{V}) = 0\} = \text{SO}(6, \mathbb{Z})$. That is, the image of the Clifford group in $\text{SO}(6)$ is the group of signed permutation matrices.*

Proof. The equality $\{\hat{V} \in \mathcal{D} \mid \text{lde}(\hat{V}) = 0\} = \text{SO}(6, \mathbb{Z})$ follows from the fact that the elements of \mathcal{D} of lde 0 are precisely the special orthogonal matrices with integer entries. To see that $\hat{\mathcal{C}}_2 = \text{SO}(6, \mathbb{Z})$, note that

$$\begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \widehat{(\zeta^\dagger S)} \otimes \mathbf{1},$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \widehat{(H \otimes H)} \widehat{(\zeta^\dagger CZ)} \widehat{(Z \otimes Z)}.$$

The fact that $\widehat{(\zeta^\dagger S)} \otimes \mathbf{1}$ and $\widehat{(H \otimes H)} \widehat{(\zeta^\dagger CZ)} \widehat{(Z \otimes Z)}$ generate $\text{SO}(6, \mathbb{Z})$ and are Cliffords completes the proof. \square

Corollary 5.3.11. $|\hat{\mathcal{C}}_2| = 2^5 6! = 23040$.

Now, let us consider $\hat{U} \in \mathcal{D}$ with $\text{lde } k = 1$. Each row/column of M must have a norm-squared of 2 while being pairwise orthogonal. The only such matrices are, up to a signed permutation of rows/columns, equivalent to

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = \widehat{\sqrt{\zeta^\dagger} C S}. \quad (5.13)$$

Noting that we've just shown Cliffords are equivalent to signed permutations in Lemma 5.3.10, we can use Proposition 5.2.6 to immediately conclude

$$\left\{ \hat{U} = \frac{1}{\sqrt{2}} M \mid \hat{U} \in \mathcal{D} \right\} \equiv \hat{\mathcal{S}} \cdot \hat{\mathcal{C}}_2.$$

where we have defined $\hat{\mathcal{S}}$ as the set \mathcal{S} multiplied by an overall phase to ensure unit determinant and then transformed to the $\text{SO}(6)$ basis.

Finally, we consider $\text{lde } k > 1$. We have the column relations

$$\sum_l M_{lm}^2 = 0 \pmod{4} \quad (5.14)$$

$$\sum_l M_{lm} M_{ln} = 0 \pmod{2} \quad \forall m \neq n \quad (5.15)$$

as well as analogous row relations. For some $x \in \mathbb{Z}$, $x^2 = 0 \pmod{4} \iff x = 0 \pmod{2}$ and $x^2 = 1 \pmod{4} \iff x = 1 \pmod{2}$, and so there must be exactly zero or four odd entries in every column/row of M by Eq. (5.14). By Eq. (5.15), we see that the number of instances where columns m and n modulo 2 “collide” (i.e. both have odd entries in the same row) must be even. Up to a permutation of rows/columns, we can then deduce that $\rho_k(\hat{U}) = M \pmod{2}$ can be one of two cases:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \quad (5.16)$$

We now introduce an important definition and subsequent property of \mathcal{D} from these observations:

Definition 5.3.12 (row/column paired). Let $\bar{M} \in \mathcal{M}_{2n \times 2n}(\mathbb{Z}_2)$. We say \bar{M} is *row paired* if for every row r of \bar{M} , there are an even number of rows r' in \bar{M} with $r' = r$ (including r itself). An analogous definition holds for *column pairing*. When a matrix is row paired, we specify that pairing as a set of sets

$$W = \{s_1, \dots, s_p\}$$

with $|s_1| \leq \dots \leq |s_p|$ and $s_l < s_m \forall l < m$. We impose the conditions $s_l \neq \emptyset$, $s_l \cap s_m = \emptyset \forall l \neq m$, $s_1 \cup \dots \cup s_p = [[1..2n]]$, and with each s_l consisting of all row indices of identical rows of \bar{M} . We call W a $|s_1| \times \dots \times |s_p|$ pairing.

Remark 5.3.13. Note that row/column paired matrices remain row/column paired regardless of permuted rows/columns. Moreover, if a matrix has a particular row pairing, then permutation of columns leaves this row pairing unchanged.

Lemma 5.3.14. *Let $\hat{U} \in \mathcal{D}$ with lde $k \geq 1$. Then $\rho_k(\hat{U})$ is both row and column paired with either a $2 \times 2 \times 2$ or 2×4 pairing.*

Proof. Examination of Eqs. (5.13) and (5.16) and Remark 5.3.13 immediately imply this fact. □

Definition 5.3.15. We will often refer to submatrices of a matrix $M \in \mathcal{M}_{m \times n}$. Let s_1 be a set of row indices s.t. $s_1 \subseteq [[1..m]]$ and s_2 a set of column indices s.t. $s_2 \subseteq [[1..n]]$. Then the submatrix of M for row indices s_1 and column indices s_2 is denoted

$$M[s_1; s_2].$$

We are now equipped to provide the key lemma on which optimal synthesis hinges.

Lemma 5.3.16. *Let $\hat{U} \in \mathcal{D}$ with lde $k \geq 1$. Then for (at least) one $\hat{G} \in \hat{\mathcal{S}}$, we have $\hat{G}^T \hat{U} \in \mathcal{D}$ with an lde of $k' = k - 1$.*

Proof. Consider the set of generators \mathcal{S} in SO(6) representation, $\hat{\mathcal{S}}$. Each of these generators is such that for $\hat{G} \in \hat{\mathcal{S}}$, $\rho_1(\hat{G}^T)$ is row/column paired with a row/column

pairing of

$$W = \{\{x_1, x_2\}, \{x_3, x_4\}, \{x_5, x_6\}\}.$$

Moreover, the submatrix formed by rows $\{x_l, x_m\} \in W$ and columns $\{x_n, x_p\} \in W$ is of the form

$$\hat{G}^T[\{x_l, x_m\}; \{x_n, x_p\}] = \begin{cases} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & \pm 1 \\ \mp 1 & 1 \end{bmatrix} & \{x_l, x_m\} = \{x_n, x_p\} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \text{otherwise} \end{cases}.$$

The pairing W of each generator is unique, and spans the set of all possible $2 \times 2 \times 2$ pairings.

We know by Lemma 5.3.14 that $\rho_k(\hat{U})$ is row paired. Suppose that $\rho_k(\hat{U})$ has the row pairing V . By Lemma 5.3.14, V must be either a $2 \times 2 \times 2$ or 2×4 pairing. If V is $2 \times 2 \times 2$ paired, choose \hat{G} such that $W = V$. If V is 2×4 paired, choose \hat{G} such that $W \cap V \neq \emptyset$ (i.e. share a two-pairing). In either case, when we examine the submatrix consisting of rows x_l and x_m with $\{x_l, x_m\} \in W$ of $\hat{G}^T \hat{U}$, we have

$$\begin{aligned} (\hat{G}^T \hat{U})[\{x_l, x_m\}; [[1..6]]] &= \sum_{\{x_n, x_p\} \in W} \hat{G}^T[\{x_l, x_m\}; \{x_n, x_p\}] \cdot \hat{U}[\{x_n, x_p\}; [[1..6]]] \\ &= \hat{G}^T[\{x_l, x_m\}; \{x_l, x_m\}] \cdot \hat{U}[\{x_l, x_m\}; [[1..6]]] \end{aligned}$$

Rows x_l and x_m of $\rho_k(\hat{U})$ must be paired per our choice of \hat{G} , and so their corre-

spending rows in \hat{U} must be of the form

$$\hat{U}[\{x_l, x_m\}; [[1..6]]] = \frac{1}{\sqrt{2}^k} \begin{bmatrix} \vec{r} + 2\vec{a} \\ \vec{r} + 2\vec{b} \end{bmatrix}$$

with $\vec{r} = (\rho_k(\hat{U}))[\{x_l, x_m\}; [[1..6]]]$ and \vec{a}, \vec{b} vectors of integers. Thus, we have

$$\begin{aligned} (\hat{G}^\top \hat{U})[\{x_l, x_m\}; [[1..6]]] &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & \pm 1 \\ \mp 1 & 1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}^k} \begin{bmatrix} \vec{r} + 2\vec{a} \\ \vec{r} + 2\vec{b} \end{bmatrix} \\ &= \frac{1}{\sqrt{2}^{k+1}} \begin{bmatrix} (1 \pm 1)\vec{r} + 2(\vec{a} \pm \vec{b}) \\ (1 \mp 1)\vec{r} + 2(\vec{b} \mp \vec{a}) \end{bmatrix} \\ &= \frac{1}{\sqrt{2}^{k-1}} \begin{bmatrix} (\frac{1 \pm 1}{2})\vec{r} + \vec{a} \pm \vec{b} \\ (\frac{1 \mp 1}{2})\vec{r} + \vec{b} \mp \vec{a} \end{bmatrix}. \end{aligned} \quad (5.17)$$

This holds for all $\{x_l, x_m\} \in W$, and as $\bigcup_{s \in W} s = [[1..6]]$, we have that

$$\hat{G}^\top \hat{U} = \frac{1}{\sqrt{2}^{k-1}} M'$$

where $M' \in \mathcal{M}_{6 \times 6}(\mathbb{Z})$. Finally, as both $\hat{G}^\top, \hat{U} \in \mathcal{D}$, we can conclude that $\hat{G}^\top \hat{U} \in \mathcal{D}$ with an lde of $k' = k - 1$. \square

Theorem 5.3.17. $\hat{U} \in \mathcal{D}$ if and only if \hat{U} is the $SO(6)$ representation of a two-qubit Clifford + Controlled-Phase operator.

Proof. The if direction holds by Lemma 5.3.9. Now, suppose \hat{U} has an lde 0. Then $\hat{U} = \hat{C}$ is a Clifford operator and we are done. Now, suppose \hat{U} has lde $k > 0$.

We know there exists some $\hat{G}_k \in \hat{\mathcal{S}}$ such that $\hat{G}_k^\top \hat{U} \in \mathcal{D}$ has an lde of $k - 1$ by Lemma 5.3.16. Then by induction, we can find a sequence $\hat{G}_1^\top \cdots \hat{G}_k^\top \hat{U} = \hat{C}$ with an lde of 0, which must again be a Clifford. Thus, \hat{U} is equivalent to

$$\hat{U} = \hat{G}_k \cdots \hat{G}_1 \cdot \hat{C}$$

which is the $\text{SO}(6)$ representation of the Clifford + Controlled-Phase operator

$$U = G_k \cdots G_1 \cdot C.$$

□

Remark 5.3.18. In using Lemma 5.3.16 to prove Theorem 5.3.17, we actually have freedom in our selection of some \hat{G}_l whenever $\rho_l(\hat{G}_{l+1}^\top \cdots \hat{G}_k^\top \hat{U})$ is 2×4 row paired. In particular, we could choose 3 different elements of $\hat{\mathcal{S}}$ in such cases. This owes itself to the relation of Eq. (5.6), in which we have

$$R(P, L)R(P, Q) = R(P, Q)R(P, iQL) = R(P, iQL)R(P, L)$$

where the leftmost operators are 3 explicitly different generators. As our goal is to produce a *unique* normal form for Clifford + Controlled-Phase operators, this ambiguity must be lifted. We choose to do so in the following way:

Definition 5.3.19 (Earliest Generator Ordering). We define *earliest generator ordering*, or EGO, as always using the lowest indexed member of our ordered sequence

Generator	Associated Row Pairings Under Earliest Generator Ordering
$R(X \otimes \mathbb{1}, \mathbb{1} \otimes X)$	$\{\{1, 4\}, \{2, 3\}, \{5, 6\}\}, \{\{1, 4\}, \{2, 3, 5, 6\}\}, \{\{2, 3\}, \{1, 4, 5, 6\}\}, \{\{5, 6\}, \{1, 2, 3, 4\}\}$
$R(Y \otimes \mathbb{1}, \mathbb{1} \otimes Y)$	$\{\{1, 3\}, \{2, 5\}, \{4, 6\}\}, \{\{1, 3\}, \{2, 4, 5, 6\}\}, \{\{2, 5\}, \{1, 3, 4, 6\}\}, \{\{4, 6\}, \{1, 2, 3, 5\}\}$
$R(Z \otimes \mathbb{1}, \mathbb{1} \otimes Z)$	$\{\{1, 2\}, \{3, 6\}, \{4, 5\}\}, \{\{1, 2\}, \{3, 4, 5, 6\}\}, \{\{3, 6\}, \{1, 2, 4, 5\}\}, \{\{4, 5\}, \{1, 2, 3, 6\}\}$
$R(Y_1, Z_2)$	$\{\{1, 2\}, \{3, 5\}, \{4, 6\}\}, \{\{3, 5\}, \{1, 2, 4, 6\}\}$
$R(Z_1, Y_2)$	$\{\{1, 3\}, \{2, 6\}, \{4, 5\}\}, \{\{2, 6\}, \{1, 3, 4, 5\}\}$
$R(Z_1, X_2)$	$\{\{1, 2\}, \{3, 4\}, \{5, 6\}\}, \{\{3, 4\}, \{1, 2, 5, 6\}\}$
$R(X_1, Z_2)$	$\{\{1, 6\}, \{2, 3\}, \{4, 5\}\}, \{\{1, 6\}, \{2, 3, 4, 5\}\}$
$R(X_1, Y_2)$	$\{\{1, 5\}, \{2, 3\}, \{4, 6\}\}, \{\{1, 5\}, \{2, 3, 4, 6\}\}$
$R(Y_1, X_2)$	$\{\{1, 3\}, \{2, 4\}, \{5, 6\}\}, \{\{2, 4\}, \{1, 3, 5, 6\}\}$
$R(X_1 X_2, Y_1 Y_2)$	$\{\{1, 4\}, \{2, 5\}, \{3, 6\}\}$
$R(X_1 X_2, Z_1 Y_2)$	$\{\{1, 4\}, \{2, 6\}, \{3, 5\}\}$
$R(Z_1 X_2, Y_1 Y_2)$	$\{\{1, 6\}, \{2, 5\}, \{3, 4\}\}$
$R(Y_1 X_2, X_1 Y_2)$	$\{\{1, 5\}, \{2, 4\}, \{3, 6\}\}$
$R(Z_1 X_2, X_1 Y_2)$	$\{\{1, 5\}, \{2, 6\}, \{3, 4\}\}$
$R(Y_1 X_2, Z_1 Y_2)$	$\{\{1, 6\}, \{2, 4\}, \{3, 5\}\}$

Table 5.1: Every generator and the explicit row pairings they will be used to reduce under earliest generator ordering.

$(\hat{\mathcal{S}})$ that satisfies the requisite properties to reduce the denominator exponent when invoking Lemma 5.3.16. The unique matching enforced by EGO is summarized in Section 5.3

Theorem 5.3.20. *There exists a unique Clifford + Controlled-Phase gate optimal normal form for the Clifford + Controlled-Phase group. Moreover, if a Clifford + Controlled-Phase operator has an $SO(6)$ representation with an lde of k , then this*

normal form contains k Controlled-Phase gates and we can synthesize this sequence in $\mathcal{O}(k)$ operations.

Proof. Suppose U is a Clifford + Controlled-Phase gate operator with Controlled-Phase gate count k' . Then its $\text{SO}(6)$ representation $\hat{U} \in \mathcal{D}$ has denominator exponent $k \leq k'$ by Lemma 5.3.9. If $k = 0$, then \hat{U} is a Clifford \hat{C} . If $k > 0$, using Lemma 5.3.16 and Definition 5.3.19 there is a unique choice of some $\hat{G}_k \in \hat{\mathcal{G}}$ such that $\hat{G}_k^T \hat{U}$ has an lde of $k - 1$. Then by induction on the denominator exponent, we have a deterministic synthesis algorithm to find a sequence such that

$$\hat{U} = \hat{G}_k \cdots \hat{G}_1 \cdot \hat{C}$$

which implies that

$$U = G_k \cdots G_1 \cdot C$$

which has a Controlled-Phase gate count of $k \leq k'$. This algorithm implies both existence and uniqueness of a normal form. To show that the normal form defined by the output of this synthesis algorithm is optimal in Controlled-Phase gate count, we note that to have an lde of k , the Clifford+Controlled-Phase gate circuit corresponding to $\hat{U} \in \mathcal{D}$ must contain at least k Controlled-Phase gates. \square

Definition 5.3.21 (Clifford+Controlled-Phase Gate Normal Form). We define the normal form which is the output of the synthesis algorithm defined in Theorem 5.3.20 as the *Clifford+Controlled-Phase Gate Normal Form*

5.4 Structure of Optimal Normal Forms

While we have described the optimal Clifford+Controlled Phase gate normal form in Theorem 5.3.20, we have not actually described the structure of the circuits to which these normal forms correspond. The goal of this section is to establish what the output of the synthesis algorithm in Theorem 5.3.20 actually looks like. We shall do so with the help of some basic graph theory terminology.

Definition 5.4.1 (\mathcal{F}_m Graph). We define the directed \mathcal{F}_m graph with vertices V and edges E as

$$\begin{aligned} V &= [[1..m]] \\ E &= \left\{ (x, y) \mid (x, y) \in V^2 \text{ and } \rho_2(\hat{\mathcal{S}}_x \cdot \hat{\mathcal{S}}_y) \text{ is } 2 \times 2 \times 2 \text{ paired} \right\} \\ \mathcal{F}_m &= (V, E) \end{aligned}$$

where the edge (x, y) is interpreted as directed *from* x *to* y .

Remark 5.4.2. We can equivalently conclude that edge $E = (x, y)$ is on the graph \mathcal{F}_m if and only if $\rho_1(\hat{\mathcal{S}}_x)$ and $\rho_1(\hat{\mathcal{S}}_y)$ with row pairings W_x and W_y respectively are such that $W_x \cap W_y = \emptyset$.

Definition 5.4.3 ($\mathcal{B}_{j,m}$ Graph). We define the directed $\mathcal{B}_{j,m}$ graph with vertices V'

and edges E' as

$$V' = [[0..m]]$$

$$E' = \{(0, x) \mid j \leq x \leq m\}$$

$$\mathcal{B}_{j,m} = (V', E').$$

Again, edge (x, y) is interpreted as directed *from* x *to* y .

Definition 5.4.4 ($\mathcal{F}_{j,m}$ Automaton Graph). The $\mathcal{F}_{j,m}$ Automaton Graph is the union of the graphs \mathcal{F}_m and $\mathcal{B}_{j,m}$. This is to say that if $\mathcal{F}_m = (V, E)$ and $\mathcal{B}_{j,m} = (V', E')$, then

$$\mathcal{F}_{j,m} = \mathcal{F}_m \cup \mathcal{B}_{j,m} = (V \cup V', E \cup E').$$

Definition 5.4.5 (Vertex Map). We define the *vertex map* $\phi : [[0..15]] \rightarrow \{\varepsilon\} \cup \mathcal{S}$ by

$$\phi(x) = \begin{cases} \varepsilon & x = 0 \\ \hat{\mathcal{S}}_x & \text{otherwise} \end{cases}$$

Thus, input of the vertex x from the graph $\mathcal{F}_{j,m}$ into ϕ results in the output of either the empty sequence ε or the x th element of (\mathcal{S}) .

Definition 5.4.6 ($\mathcal{F}_{j,m}$ Automaton Walk). Draw the graph $\mathcal{F}_{j,m} = (V, E)$. Take any length n walk \mathcal{W} on this graph starting from the vertex 0. This walk takes the sequence of vertices

$$(V_0, V_1, \dots, V_n)$$

where $V_0 = 0$ and $(V_j, V_{j+1}) \in E$. We then define the output of a $\mathcal{F}_{j,m}$ automaton walk as

$$\phi(V_0)\phi(V_1)\cdots\phi(V_n).$$

Proposition 5.4.7. *A circuit is in Clifford+Controlled-Phase gate normal form if and only if it is of the form*

$$(\mathcal{F}_{1,3} \text{ Automaton Walk})(\mathcal{F}_{4,9} \text{ Automaton Walk})(\mathcal{F}_{10,15} \text{ Automaton Walk}) \cdot \mathcal{C}. \quad (5.18)$$

Proof. We will first establish a few lemmas that in combination suffice to prove the result. For these lemmas, let $\hat{U} \in \mathcal{D}$ with lde k such that $\rho_k(\hat{U})$ has the row pairing V_k . V_k implies under EGO that we use the unique generator $\hat{G}_k \in \hat{\mathcal{S}}$ with associated row pairing W_k to reduce the denominator exponent to $k-1$. We can then explicitly check the possible row pairings V_{k-1} of $\rho_{k-1}(\hat{G}_k^T \hat{U})$ to try and deduce which operator $\hat{G}_{k-1} \in \hat{\mathcal{S}}$ with associated row pairing W_{k-1} must follow. We will also consider left multiplication by some $\hat{G}_{k+1} \in \hat{\mathcal{S}}$ with row pairing W_{k+1} such that $\hat{G}_{k+1} \hat{U}$ has lde $k+1$ and associated row pairing V_{k+1} . Operators \hat{G}_{k+1} , \hat{G}_k and \hat{G}_{k-1} will have indices x , y and z , respectively, in the sequence $(\hat{\mathcal{S}})$.

With each proof, we provide a graphical diagram to assist the reader in visualizing the pairing of the operators in question. Each diagram consists of panes. In a given pane, we have two labeled columns, corresponding to a row pairing of $\rho_k(\hat{U})$ which is mapped to $\rho_{k-1}(\hat{G}_k^T \hat{U})$ or $\rho_{k+1}(\hat{G}_{k+1} \hat{U})$ as specified. Within each column are three blue boxes – these correspond to the three sets $s_j = \{x_{j_1}, x_{j_2}\}$ which belong

to the row pairings W_k , W_{k-1} , or W_{k+1} . Sometimes, the blue boxes specify to which set s_j belongs. These sets are \mathcal{A} , \mathcal{B} , \mathcal{E} , $\bar{\mathcal{E}}_3$, and $\underline{\mathcal{E}}_3$ which are defined as follows:

$$\mathcal{A} = \{\{x, y\} \mid (x, y) \in [[1..3]]^2 \text{ and } x \neq y\}$$

$$\mathcal{B} = \{\{x, y\} \mid (x, y) \in [[4..6]]^2 \text{ and } x \neq y\}$$

$$\mathcal{E} = \{\{x, y\} \mid (x, y) \in ([[1..3]], [[4..6]])\}$$

$$\bar{\mathcal{E}}_3 = \{\{x, y\} \mid (x, y) \in ([[1..3]], [[4..6]]) \text{ and } x \neq y \pmod{3}\}$$

$$\underline{\mathcal{E}}_3 = \mathcal{E}/\bar{\mathcal{E}}_3$$

Occasionally, we say that some $s \in \mathcal{A}$ or $s \in \mathcal{B}$ – we denote this as $s \in \mathcal{A}|\mathcal{B}$ (rather than the usual $s \in \mathcal{A} \cup \mathcal{B}$) so that if we have two sets $s_1 \in \mathcal{A}|\mathcal{B}$ and $s_2 \in \mathcal{B}|\mathcal{A}$, we can specify if $s_1 \in \mathcal{A}$ then $s_2 \in \mathcal{B}$ and if $s_1 \in \mathcal{B}$ then $s_2 \in \mathcal{A}$.

The larger green boxes which subsume the blue boxes represent the sets of paired rows in V_k , V_{k-1} , or V_{k+1} . In the case where $W_j = V_j$, these each blue box has an equivalent green box. If V_j is explicitly a 2×4 pairing, one of the green boxes contains two blue boxes corresponding to four paired rows. Sometimes, we use red boxes in place of green boxes. This is to indicate that a 4-pairing may or may not be present. When green boxes overlap with red boxes, we mean that anything in the green box must explicitly be paired, and may or may not be paired with anything bridged by the red box.

Each pane consists of determining to where two indices map upon $\rho_k(\hat{U}) \rightarrow \rho_{k-1}(\hat{G}_k^\top \hat{U})$ or $\rho_k(\hat{U}) \rightarrow \rho_{k+1}(\hat{G}_{k+1} \hat{U})$. An arrow which originates on a blue box

signifies that one element of the set s to which that blue box corresponds is mapped somewhere. When that arrow terminates on the perimeter of a red box, we mean that the element in question ends up at least somewhere within the confines of the red box. When it terminates on the perimeter of a green box, we mean that the element must end up within the space the green box confines. Finally, when the arrow terminates on a blue box, we mean that the row index is explicitly contained within that set. The last pane of every diagram corresponds both to the transformation $W_j \rightarrow W_{j'}$ and $V_j \rightarrow V_{j'}$, sometimes with multiple possible resultant pairings. \square

Lemma 5.4.8. V is a $2 \times 2 \times 2$ pairing if and only if \hat{U} is of the form

$$(\mathcal{F}_{1,15} \text{ Automaton Walk}) \cdot \mathcal{C}. \tag{5.19}$$

Proof. Let us consider the “only if” direction first. If V_k is a $2 \times 2 \times 2$ pairing, then for $s \in V_k$, by Eq. (5.17) we have

$$\left(\rho_{k-1}(\hat{G}_k^\top \hat{U}) \right) [s; [[1..6]]] = \begin{bmatrix} \vec{r}_1 \\ \vec{r}_2 \end{bmatrix}$$

with $\vec{r}_1 \neq \vec{r}_2$. We can immediately conclude $s \notin V_{k-1}$, and moreover as this holds for every $s \in V_k$, no four rows of $\rho_{k-1}(\hat{G}_k^\top \hat{U})$ match and thus V_{k-1} is a $2 \times 2 \times 2$ pairing (see Fig. 5.1). Therefore, since $W_k = V_k$, and $W_{k-1} = V_{k-1}$ by Lemma 5.3.16 and thus $W_k \cap W_{k-1} = \emptyset$, we have by Remark 5.4.2 that (y, z) is an edge on the graph \mathcal{F}_{15} . By induction on the denominator exponent, we see that for any \hat{U} with lde k and with a $2 \times 2 \times 2$ row pairing of $\rho_k(\hat{U})$, synthesis under EGO must result

in a sequence of operators consistent with a walk on the Graph \mathcal{S}_{15} under the vertex map until we reach a denominator exponent of zero. Noting that our initial generator \hat{G}_k may be any of the fifteen elements of \mathcal{S} , we conclude that the output must be some operator of the form in Eq. (5.19).

Now, suppose it is known that \hat{U} has the form of Eq. (5.19) and is a $2 \times 2 \times 2$ pairing. Consider left multiplication of this operator by some \hat{G}_{k+1} such that (x, y) is an edge on the graph \mathcal{F}_{15} . By Remark 5.4.2, we know that $W_{k+1} \cap W_k = \emptyset$. Let $s \in W_{k+1}$. Using these facts and similar methods to Eq. (5.17) we have

$$\begin{aligned} \left(\rho_k(\hat{U})\right) [s; [[1..6]]] &= \begin{bmatrix} \vec{r}_1 \\ \vec{r}_2 \end{bmatrix} \\ \left(\rho_{k+1}(\hat{G}_{k+1}\hat{U})\right) [s; [[1..6]]] &= \begin{bmatrix} \vec{r}_1 + \vec{r}_2 \\ \vec{r}_1 + \vec{r}_2 \end{bmatrix} \end{aligned}$$

with $\vec{r}_1 \neq \vec{r}_2$. Thus, the rows of s are paired in our new operator. We can further conclude $s \in V_{k+1}$ such that $V_{k+1} = W_{k+1}$, as a 2×4 pairing for V_{k+1} would imply that one s is such that $\vec{r}_1 + \vec{r}_2 = 0$ which is a contradiction; thus V_{k+1} must be a $2 \times 2 \times 2$ pairing (see Fig. 5.2). This implies that if \hat{U} has the form of Eq. (5.19), is a $2 \times 2 \times 2$ pairing, and is as a sequence the output of our synthesis algorithm, then so is $\hat{G}_{k+1}\hat{U}$. Note that by Lemma 5.3.9 all length one walks (i.e., operators with lde one) on the graph $\mathcal{F}_{1,15}$ have a $2 \times 2 \times 2$ pairing and are consistent with the output of our synthesis algorithm. Thus, by induction, we conclude that every such sequence must be a valid output of the synthesis algorithm, and in turn can

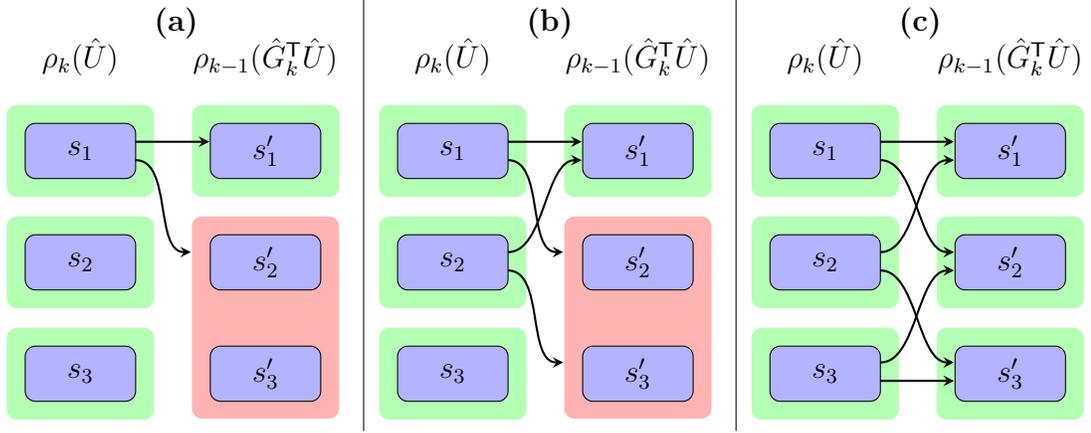


Figure 5.1: Proof diagram for the “only if” direction of Lemma 5.4.8. In pane (a), we observe that the first pair $s_1 \in V_k$ cannot be $\in V_{k-1}$. In pane (b), we note that a pair $s_2 \in V_k$ must send exactly one element to s'_1 and one elsewhere. Finally, in pane (c) we see that $s_3 \in V_k$ cannot be paired in V_{k-1} , restricting the final outcome to a $2 \times 2 \times 2$ pairing with $V_k \cap V_{k-1} = \emptyset$ and $W_k \cap W_{k-1} = \emptyset$.

conclude Lemma 5.4.8 holds. \square

Lemma 5.4.9. V is a 2×4 pairing with some $s \in V$ where $s \in \mathcal{E}$ if and only if \hat{U} is of the form

$$(\mathcal{F}_{1,9} \text{ Automaton Walk})(\mathcal{F}_{10,15} \text{ Automaton Walk}) \cdot \mathcal{C} \quad (5.20)$$

but not of the form in Eq. (5.19).

Proof. Beginning with the “only if” direction, if V_k is a 2×4 pairing with some $s_3 \in V$ such that $s_3 \in \mathcal{E}$, then we have one pair $s_3 \in W_k \cap V_k$ and two pairs $\{s_1, s_2\} = W_k \setminus V_k$. Under EGO, we have $G_k \in \{\mathcal{S}_1, \dots, \mathcal{S}_9\}$ and so $s_1 \in \mathcal{A}$ and

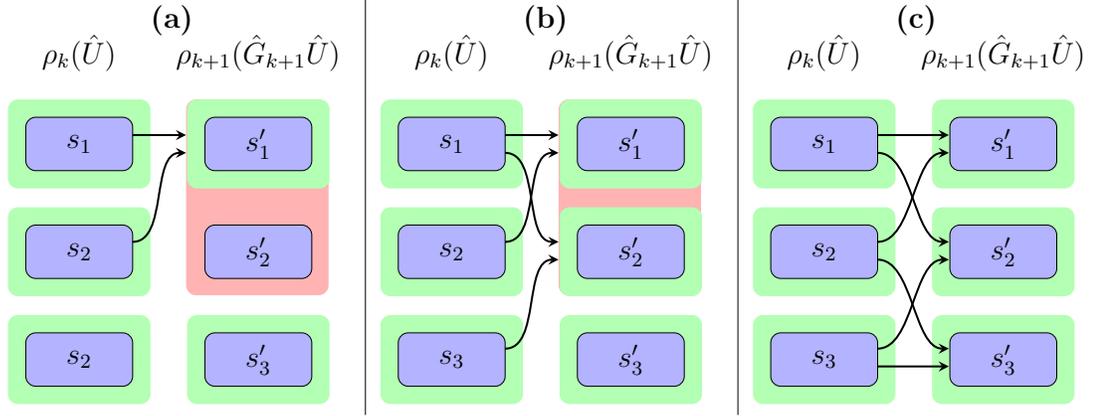


Figure 5.2: Proof diagram for the “if” direction of Lemma 5.4.8. In pane (a), we observe that the first pair $s'_1 \in V_{k+1}$ cannot be $\in V_k$ and likewise cannot be the 2-pairing in a 2×4 pairing. In pane (b), we apply the same logic to $s'_2 \in V_{k+1}$, noting it may be part of a 4-pairing. Finally, in pane (c) we see that $s'_3 \in V_{k+1}$ has the same restrictions, forcing the final outcome to be a $2 \times 2 \times 2$ pairing with $V \cap V_{k+1} = \emptyset$ and $W_k \cap W_{k+1} = \emptyset$.

$s_2 \in \mathcal{B}$. By Eq. (5.17), we have

$$\begin{aligned} \left(\rho_{k-1}(\hat{G}_k^\top \hat{U}) \right) [s_3; [[1..6]]] &= \begin{bmatrix} \vec{r}_1 \\ \vec{r}_1 \end{bmatrix} \\ \left(\rho_{k-1}(\hat{G}_k^\top \hat{U}) \right) [s_j; [[1..6]]] &= \begin{bmatrix} \vec{r}_{j_1} \\ \vec{r}_{j_2} \end{bmatrix} \quad \forall j \neq 3 \end{aligned}$$

with $\vec{r}_{j_1} \neq \vec{r}_{j_2}$ which implies s_3 is either a pair or part of a 4-pairing in V_{k-1} and $\{s_1, s_2\} \cap V_{k-1} = \emptyset$. These constraints immediately imply that either V_{k-1} is a $2 \times 2 \times 2$ pairing such that there are three pairs $s'_j \in \mathcal{E}$, or V_{k-1} is a 2×4 pairing with one pair $s'_1 \in \mathcal{E}$ (see Fig. 5.3). In the first case we know $V_{k-1} = W_{k-1}$, and as we have $s_3 \in V_{k-1}$, then $W_k \cap W_{k-1} \neq \emptyset$. Given the graph $\mathcal{F}_{15} = (V, E)$, this implies

$$G_{k-1} \in \{\mathcal{S}_z \mid z \in [[10..15]], (y, z) \notin E\}. \quad (5.21)$$

On the other hand, if V_{k-1} is a 2×4 pairing, we see that as $s_3 \cap s'_1 = \emptyset$ with $\{s_1, s'_2\} \subset \mathcal{A}$ and $\{s_2, s'_3\} \subset \mathcal{B}$, the pairing of V_{k-1} under EGO corresponds to using one of the generators $\{\hat{\mathcal{S}}_1, \dots, \hat{\mathcal{S}}_9\}$. Restricted to this set, the only way to have pairings W_k and W_{k-1} with s_j and s'_j as specified is if $W_k \cap W_{k-1} = \emptyset$. We can thus conclude that (y, z) is an edge on the graph \mathcal{F}_9 . By induction on the denominator exponent, we see that for any \hat{U} with lde k and with a 2×4 row pairing of $\rho_k(\hat{U})$ with some $s \in V_k$ such that $s \in \mathcal{E}$, synthesis under EGO must result in a sequence of operators consistent with a walk on the graph \mathcal{F}_9 under the vertex map until we reach an operator with an associated $2 \times 2 \times 2$ row pairing which in turn must be consistent with Eq. (5.21). As our initial generator \hat{G}_k may be any of the first 9 elements of (\mathcal{S}) , using Lemma 5.4.8 we conclude that the output of the synthesis algorithm must be some operator of the form in Eq. (5.20).

Now, suppose it is known that \hat{U} has the form of Eq. (5.20) and is a 2×4 pairing with some $s_3 \in V_k$ such that $s_3 \in \mathcal{E}$. Consider left multiplication of this operator by some \hat{G}_{k+1} such that (x, y) is an edge on the graph \mathcal{F}_9 . By Remark 5.4.2, we know that $W_{k+1} \cap W_k = \emptyset$. There must be two pairs $\{s'_2, s'_3\} \subset W_{k+1}$ that are not paired in V_k and one pair $s'_1 \in W_{k+1}$ that is paired in V_k . Furthermore, as $W_{k+1} \cap W_k = \emptyset$, we know that $s'_1 \in \mathcal{E}$. Using these facts and similar methods to

Eq. (5.17) we have

$$\left(\rho_k(\hat{U})\right) [s'_j; [[1..6]]] = \begin{bmatrix} \vec{r}_{j_1} \\ \vec{r}_{j_2} \end{bmatrix} \forall j \neq 1 \quad (5.22)$$

$$\left(\rho_{k+1}(\hat{G}_{k+1}\hat{U})\right) [s'_j; [[1..6]]] = \begin{bmatrix} \vec{r}_{j_1} + \vec{r}_{j_2} \\ \vec{r}_{j_1} + \vec{r}_{j_2} \end{bmatrix} \forall j \neq 1 \quad (5.23)$$

$$\left(\rho_{k+1}(\hat{G}_{k+1}\hat{U})\right) [s'_1; [[1..6]]] = \begin{bmatrix} \vec{0} \\ \vec{0} \end{bmatrix} \quad (5.24)$$

with $\vec{r}_{j_1} \neq \vec{r}_{j_2}$. Thus, the rows of s'_2 and s'_3 are paired in our new operator which must have lde $k + 1$. We see that $s'_1 \in V_{k+1} \cap W_{k+1}$ and that V_{k+1} must be a 2×4 pairing (see Fig. 5.4). This implies that if \hat{U} has the form of Eq. (5.20), is a 2×4 pairing with some $s \in V_k$ where $s \in \mathcal{E}$, and is as a sequence the output of our synthesis algorithm, then so must be $\hat{G}_{k+1}\hat{U}$.

Now, consider left-multiplication of \hat{U} when V_k is a $2 \times 2 \times 2$ pairing with $G_k \in \{\mathcal{S}_{10}, \dots, \mathcal{S}_{15}\}$ by a generator $\hat{G}_{k+1} \in \{\mathcal{S}_1, \dots, \mathcal{S}_9\}$ such that $s_1 \in W_{k+1} \cap W_k$ and with two pairs $\{s'_2, s'_3\} \in W_{k+1} \setminus W_k$. Our set restrictions imply that $s_1 \in \mathcal{E}$. Then we see Eqs. (5.22) to (5.24) hold in this case and so by the same reasoning V_{k+1} must be a 2×4 pairing with $s'_1 = s_1 \in V_{k+1}$ (see Fig. 5.5). Thus, any operator of the form Eq. (5.20) but not Eq. (5.19) where the $\mathcal{F}_{1,9}$ automaton walk is of length one must be such that it has an associated 2×4 pairing V_k with $s \in V_k$ where $s \in \mathcal{E}$. By induction, we conclude that every sequence of the form Eq. (5.20) must be a valid output of the synthesis algorithm, and in turn can conclude Lemma 5.4.9

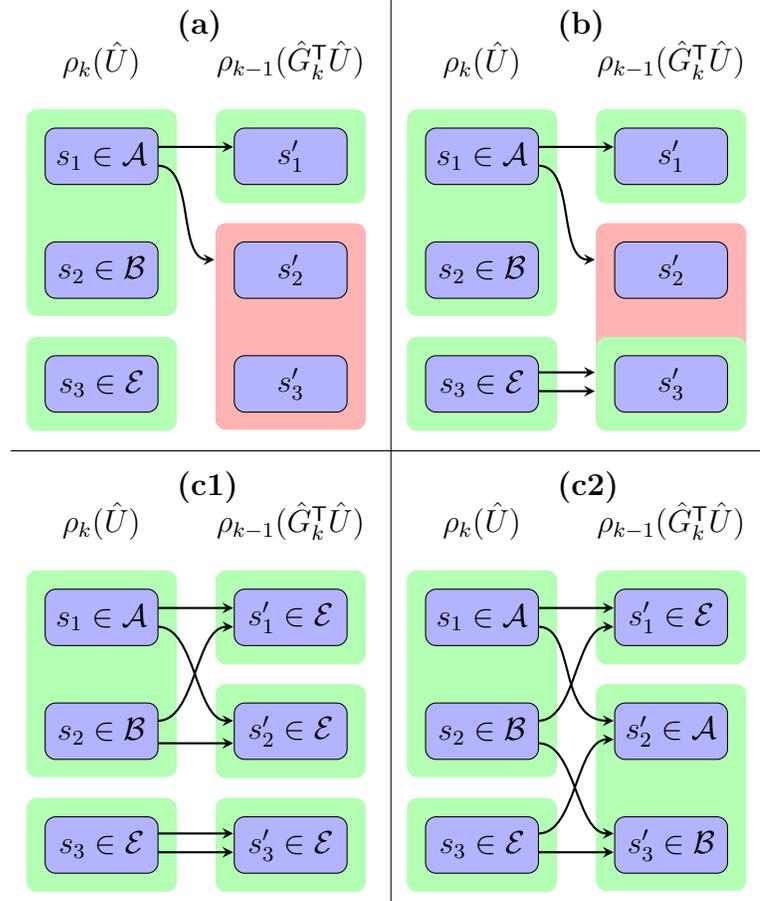


Figure 5.3: Proof diagram for the “only if” direction of Lemma 5.4.9. In pane (a), we observe that the pair $s_1 \in V_k$ cannot be $\in V_{k-1}$. In pane (b), we note that the pair $s_3 \in V_k$ must be paired in V_{k-1} and cannot be a 2-pair in a 2×4 pairing. In pane (c1), we see that if V_{k-1} is a $2 \times 2 \times 2$ pairing, the resulting sets in $V_{k-1} = W_{k-1}$ must be such that $W_{k-1} \subset \mathcal{E}$. In pane (c2), if V_{k-1} is a 2×4 pairing then we see that the 2-pair $s'_1 \in \mathcal{E}$. Under EGO, the remaining pairs of W_{k-1} must then belong to the sets \mathcal{A} and \mathcal{B} .

holds. □

Lemma 5.4.10. *V is a 2×4 pairing with some $s \in V$ such that $s \in \mathcal{A} \cup \mathcal{B}$ if and only if \hat{U} is of the form*

$$(\mathcal{F}_{1,3} \text{ Automaton Walk})(\mathcal{F}_{4,9} \text{ Automaton Walk})(\mathcal{F}_{10,15} \text{ Automaton Walk}) \cdot \mathcal{C} \tag{5.25}$$

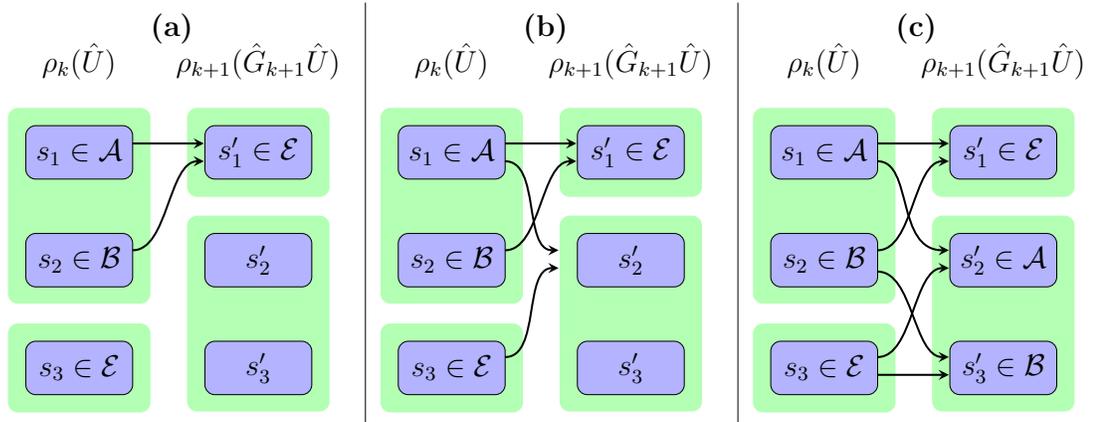


Figure 5.4: Proof diagram for the induction hypothesis of the “if” direction of Lemma 5.4.9. In pane (a), we observe that $s'_1 \in \mathcal{E}$ which must form the 2-pair in the 2×4 pairing V_{k+1} must come from the 4-pairing $\in V_k$. In panes (b) and (c), we form the 4-pairing of V_{k+1} from the remaining elements. Under EGO, $s'_2 \in \mathcal{A}$ and $s'_3 \in \mathcal{B}$.

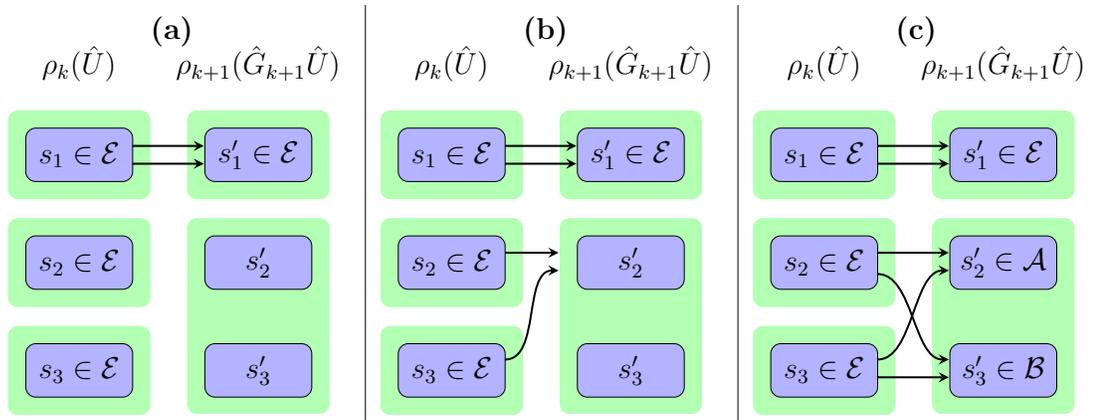


Figure 5.5: Proof diagram for the base case of the “if” direction of Lemma 5.4.9. In pane (a), we observe that $s_1 \in \mathcal{E}$ must form the 2-pair s'_1 in the 2×4 pairing V_{k+1} . In panes (b) and (c), we form the 4-pairing of V_{k+1} from the remaining elements. Under EGO, $s'_2 \in \mathcal{A}$ and $s'_3 \in \mathcal{B}$.

but not of the form in Eq. (5.19) or Eq. (5.20).

Proof. Beginning with the “only if” direction, if V_k is a 2×4 pairing with some $s_3 \in V$ such that $s_3 \in \mathcal{A} \cup \mathcal{B}$, then we have one pair $s_3 \in W_k \cap V_k$. Furthermore, under EGO there are pairs $\{s_1, s_2\} = W \setminus V$ with $s_1 \in \mathcal{A} \cup \mathcal{B}$ and $s_2 \in \overline{\mathcal{E}}_3$. By Eq. (5.17), we have

$$\begin{aligned} \left(\rho_{k-1}(\hat{G}_k^T \hat{U})\right) [s_3; [[1..6]]] &= \begin{bmatrix} \vec{r}_1 \\ \vec{r}_1 \end{bmatrix} \\ \left(\rho_{k-1}(\hat{G}_k^T \hat{U})\right) [s_j; [[1..6]]] &= \begin{bmatrix} \vec{r}_{j_1} \\ \vec{r}_{j_2} \end{bmatrix} \quad \forall j \neq 3 \end{aligned}$$

with $\vec{r}_{j_1} \neq \vec{r}_{j_2}$ which implies s_3 is either a pair or part of a 4-pairing in V_{k-1} and $\{s_2, s_3\} \cap V_{k-1} = \emptyset$.

Suppose V_{k-1} is $2 \times 2 \times 2$ paired. As $s_3 \in V_{k-1}$ and $\{s_2, s_3\} \cap V_{k-1} = \emptyset$, we can conclude there must be two pairs $\{s'_1, s'_2\} \subset V_{k-1}$ such that $s'_1 \in \underline{\mathcal{E}}_3$ and $s'_2 \in \mathcal{A} \cup \mathcal{B}$. Letting the graph \mathcal{F}_9 have edges E , we see that our new row pairing $V_{k-1} = W_{k-1}$ along with $W_k \cap W_{k-1} \neq \emptyset$ implies

$$G_{k-1} \in \{\mathcal{S}_z \mid z \in [[4..9]], (y, z) \notin E\}. \quad (5.26)$$

Suppose instead V_{k-1} is 2×4 paired. There are two possibilities for $s'_1 \in V_{k-1}$: either $s'_1 \in \underline{\mathcal{E}}_3$ or $s'_1 \in \mathcal{A} \cup \mathcal{B}$ with $s'_1 \notin W_k$. In the first case, under EGO we see that the corresponding W_{k-1} is always such that $W_k \cap W_{k-1} \neq \emptyset$ and so again Eq. (5.26)

holds. In the second case, under EGO we immediately conclude that $W_k \cap W_{k-1} = \emptyset$ and so $G_{k-1} \in \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$ such that (y, z) is an edge on the graph \mathcal{F}_3 . Refer to Fig. 5.6 for a visual aid.

By induction on the denominator exponent, we see that for any \hat{U} with lde k and with a 2×4 row pairing of $\rho_k(\hat{U})$ with some $s \in V_k$ such that $s \in \mathcal{A} \cup \mathcal{B}$, synthesis under EGO must result in a sequence of operators consistent with a walk on the graph \mathcal{F}_3 under the vertex map until we either reach an operator with an associated $2 \times 2 \times 2$ row pairing or a 2×4 row pairing. Regardless of which, there is an $s' \in V_{k-1}$ such that $s' \in \underline{\mathcal{E}}_3$ which implies consistency with Eq. (5.26). As our initial generator \hat{G}_k may be any of the first 3 elements of (\mathcal{S}) , using Lemmas 5.4.8 and 5.4.9 we conclude that the output of the synthesis algorithm must be some operator of the form in Eq. (5.25) but not Eqs. (5.19) and (5.20).

Now, suppose it is known that \hat{U} has the form of Eq. (5.25) and is a 2×4 pairing with some $s_3 \in V_k$ such that $s_3 \in \mathcal{A}|\mathcal{B}$. Furthermore, let $s_1 \in \mathcal{B}|\mathcal{A}$ and $s_2 \in \overline{\mathcal{E}}_3$ such that $\{s_1, s_2\} \subset W_k$ and which form a 4-pairing. Consider left multiplication of this operator by some \hat{G}_{k+1} such that (x, y) is an edge on the graph \mathcal{F}_3 . By Remark 5.4.2, we know that $W_{k+1} \cap W_k = \emptyset$. There must be two pairs $\{s'_2, s'_3\} \subset W_{k+1}$ that are not paired in V_k and one pair $s'_1 \in W_{k+1}$ that is part of the 4-pairing in V_k . Restricted to the set of operators $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$, by inspection the only possibility is $s'_1 \in \mathcal{B}|\mathcal{A}$.

Using these facts and similar methods to Eq. (5.17) we have

$$\left(\rho_k(\hat{U})\right) [s'_j; [[1..6]]] = \begin{bmatrix} \vec{r}_{j_1} \\ \vec{r}_{j_2} \end{bmatrix} \forall j \neq 1 \quad (5.27)$$

$$\left(\rho_{k+1}(\hat{G}_{k+1}\hat{U})\right) [s'_j; [[1..6]]] = \begin{bmatrix} \vec{r}_{j_1} + \vec{r}_{j_2} \\ \vec{r}_{j_1} + \vec{r}_{j_2} \end{bmatrix} \forall j \neq 1 \quad (5.28)$$

$$\left(\rho_{k+1}(\hat{G}_{k+1}\hat{U})\right) [s'_1; [[1..6]]] = \begin{bmatrix} \vec{0} \\ \vec{0} \end{bmatrix} \quad (5.29)$$

with $\vec{r}_{j_1} \neq \vec{r}_{j_2}$. Thus, the rows of s'_j are paired in our new operator which must have lde $k + 1$. We see that $s'_1 \in V_{k+1} \cap W_{k+1}$ and that V_{k+1} must be a 2×4 pairing (see Fig. 5.7). This implies that if \hat{U} has the form of Eq. (5.25), is a 2×4 pairing with some $s \in V_k$ where $s \in \mathcal{A}|\mathcal{B}$, and is as a sequence the output of our synthesis algorithm, then so must be $\hat{G}_{k+1}\hat{U}$.

Let \hat{U} have pairing V_k such that $s_3 \in V_k$ with $s_3 \in \mathcal{E}_3$. By Lemmas 5.4.8 and 5.4.9 we know $G_k \in \{\mathcal{S}_4, \dots, \mathcal{S}_9\}$. Consider left-multiplication by a generator $\hat{G}_{k+1} \in \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$ such that $s_1 = s'_1 \in W_{k+1} \cap W_k$ and where we have two pairs $\{s'_2, s'_3\} \in W_{k+1} \setminus W_k$. Our set restrictions imply that $s'_1 \in \mathcal{A}|\mathcal{B}$. Then we see Eqs. (5.27) to (5.29) hold in this case and so by the same reasoning V_{k+1} must be a 2×4 pairing with $s'_1 \in V_{k+1}$ (see Fig. 5.8). Thus, any operator of the form Eq. (5.25) but not Eq. (5.20) nor Eq. (5.19) where the $\mathcal{F}_{1,3}$ automaton walk is of length one must be such that it has an associated 2×4 pairing V_k with $s \in V_k$ where $s \in \mathcal{A}|\mathcal{B}$. By induction, we conclude that every sequence of the form Eq. (5.25) must be a

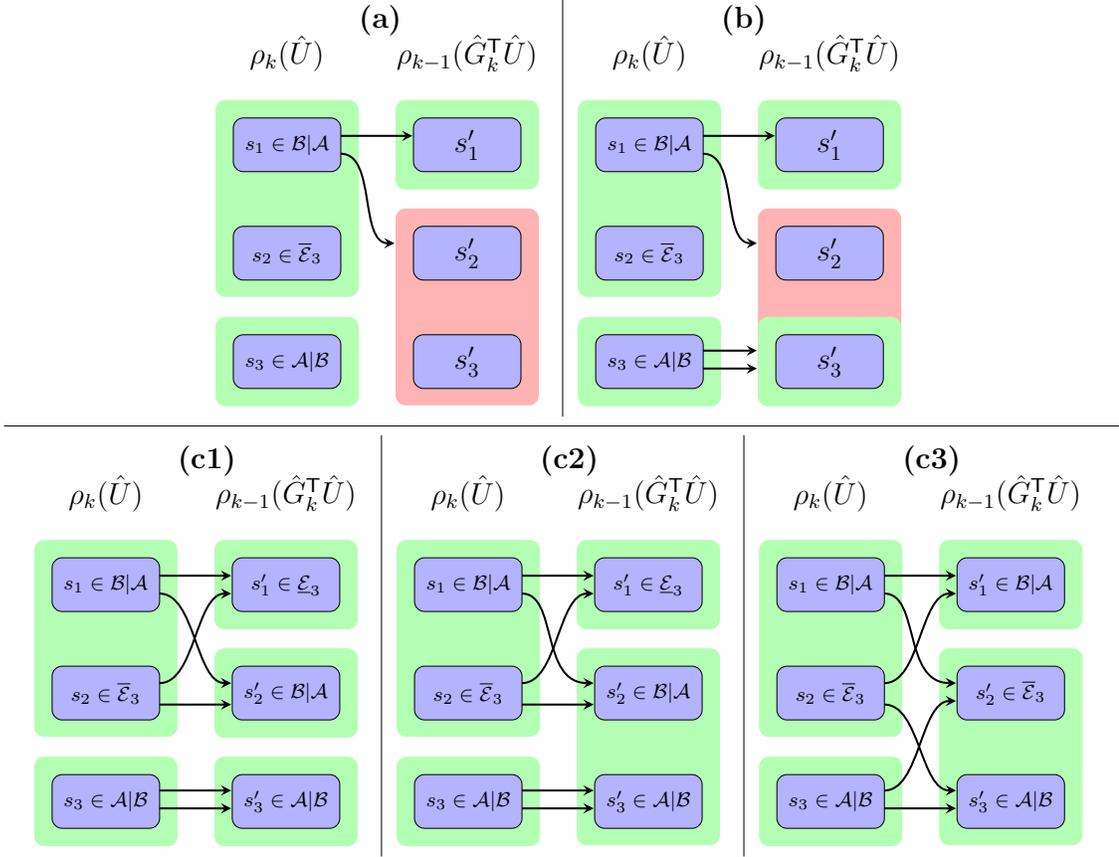


Figure 5.6: Proof diagram for the “only if” direction of Lemma 5.4.10. In pane (a), we observe that $s_1 \in \mathcal{B}|\mathcal{A}$ must not be paired in V_{k-1} . In pane (b), we see that $s_3 \in \mathcal{A}|\mathcal{B}$ must remain paired in V_{k-1} . Pane (c1) establishes that in the case where V_{k-1} is a $2 \times 2 \times 2$ pairing, then W_{k-1} must correspond to one of $\{\mathcal{S}_4, \dots, \mathcal{S}_9\}$. In pane (c2), we see that there can be instances where V_{k-1} is a 2×4 pairing such that W_{k-1} must likewise correspond to one of $\{\mathcal{S}_4, \dots, \mathcal{S}_9\}$. Finally, in pane (c2), we show that there can likewise be instances where V_{k-1} is a 2×4 pairing such that W_{k-1} must correspond to one of $\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$ with $W_k \cap W_{k-1} = \emptyset$.

valid output of the synthesis algorithm, and in turn can conclude Lemma 5.4.10

holds. □

Proof of Proposition 5.4.7. We first note the following inclusions for our automaton

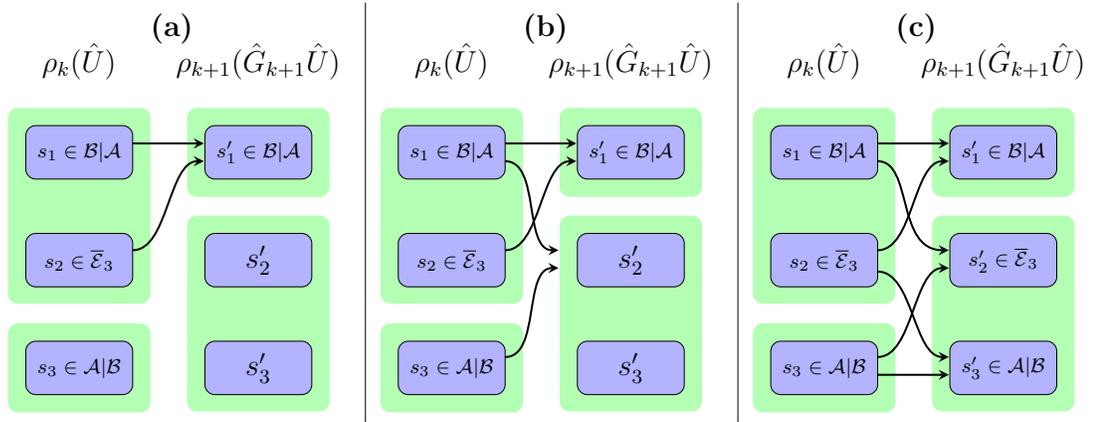


Figure 5.7: Proof diagram for the induction hypothesis of the “if” direction of Lemma 5.4.10. In pane (a), we observe that $s'_1 \in \mathcal{B}|\mathcal{A}$ which must form the 2-pair in the 2×4 pairing V_{k+1} must come from the 4-pairing $\in V_k$. In panes (b) and (c), we form the 4-pairing of V_{k+1} from the remaining elements. Under EGO, $s'_2 \in \bar{\mathcal{E}}_3$ and $s'_3 \in \mathcal{A}|\mathcal{B}$.

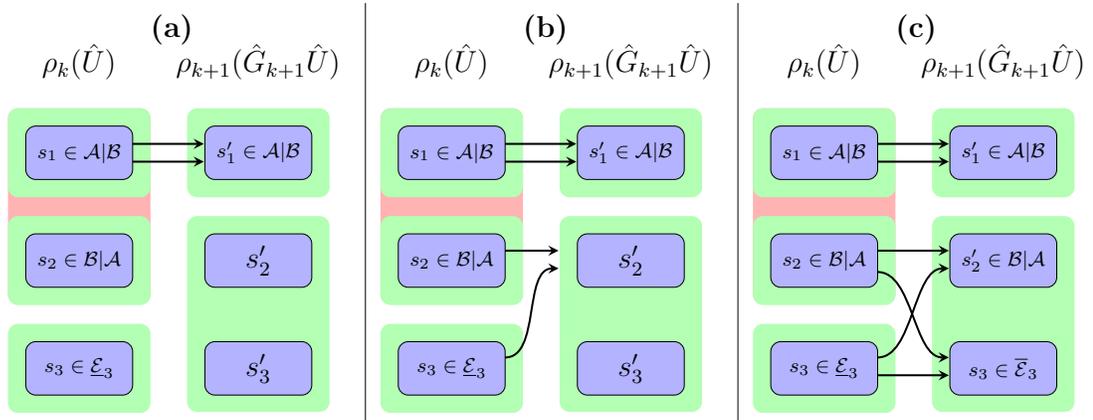


Figure 5.8: Proof diagram for the base case of the “if” direction of Lemma 5.4.10. In pane (a), we observe that $s_1 \in \mathcal{A}|\mathcal{B}$ must form the 2-pair s'_1 in the 2×4 pairing V_{k+1} . In panes (b) and (c), we form the 4-pairing of V_{k+1} from the remaining elements. Under EGO, $s'_2 \in \mathcal{B}|\mathcal{A}$ and $s'_3 \in \bar{\mathcal{E}}_3$.

walks:

$$\begin{aligned}
& (\mathcal{F}_{1,15} \text{ Automaton Walk}) \\
& \subset (\mathcal{F}_{1,9} \text{ Automaton Walk})(\mathcal{F}_{10,15} \text{ Automaton Walk}) \\
& \subset (\mathcal{F}_{1,3} \text{ Automaton Walk})(\mathcal{F}_{4,9} \text{ Automaton Walk})(\mathcal{F}_{10,15} \text{ Automaton Walk})
\end{aligned}$$

By these inclusions, Theorems 5.3.17 and 5.3.20 and Lemmas 5.4.8 to 5.4.10 we can conclude that a circuit is in Clifford+Controlled-Phase gate normal form if and only if it has the form

$$(\mathcal{F}_{1,3} \text{ Automaton Walk})(\mathcal{F}_{4,9} \text{ Automaton Walk})(\mathcal{F}_{10,15} \text{ Automaton Walk}) \cdot \mathcal{C}$$

□

Lemma 5.4.11. *There are*

$$86400(3 \cdot 8^n - 2 \cdot 4^n)$$

Clifford+Controlled-Phase operators of Controlled-Phase gate count precisely $n \geq 1$.

Proof. We can use the internal structure of the normal form from Proposition 5.4.7

to count the number of operators for a given Controlled-Phase gate count n . Ex-

plicity, this is

$$\left[6 \cdot 8^{n-1} + 6 \cdot 4^{n-1} + 3 \cdot 2^{n-1} + \sum_{0 < l < n} 18 \cdot 2^{2n-3-l} + \sum_{0 < l < n} 18 \cdot 2^{3n-4-2l} \right. \\ \left. + \sum_{0 < j < n} 36 \cdot 2^{3n-5-j} + \sum_{0 < l < n-j} \sum_{0 < j < n} 108 \cdot 2^{3n-6-j-2l} \right] \cdot |\mathcal{C}|$$

These terms represent, in order, the number of length n sequences from the: purely $\mathcal{F}_{10,15}$ walk automaton, purely $\mathcal{F}_{4,9}$ walk automaton, purely $\mathcal{F}_{1,3}$ walk automaton, partly $\mathcal{F}_{1,3}$ and $\mathcal{F}_{4,9}$ walk automata, partly $\mathcal{F}_{1,3}$ and $\mathcal{F}_{10,15}$ walk automata, partly $\mathcal{F}_{4,9}$ and $\mathcal{F}_{10,15}$ walk automata, and partly $\mathcal{F}_{1,3}$, $\mathcal{F}_{4,9}$, and $\mathcal{F}_{10,15}$ walk automata. After applying the geometric series formula a few times and substituting in $|\mathcal{C}| = 92160$, we arrive at the desired result. \square

Corollary 5.4.12. *There are*

$$\frac{46080}{7} (45 \cdot 8^n - 35 \cdot 4^n + 4)$$

Clifford+Controlled-Phase operators of Controlled-Phase gate count $\leq n$.

Proof. Use the geometric series formula in conjunction with Lemma 5.4.11 and a value of 92160 for the Controlled-Phase gate count $n = 0$ operators consistent with the cardinality of the Clifford group. \square

Lemma 5.4.13. *In order to ε -approximate any two-qubit special unitary operator,*

there are circuits that will require at least

$$n \gtrsim 5 \log_2 \left(\frac{1}{\varepsilon} \right) - 0.67$$

Controlled-Phase gates.

Proof. By volume counting argument as in Eq. (2.3). Each operator must occupy an ε -ball worth of volume in 15-dimensional SU(4) space, and the sum of all these volumes must add to the total SU(4) volume of $\frac{\sqrt{2}\pi^9}{3}$. The number of circuits up to Controlled-Phase gate count n is taken from Corollary 5.4.12 (we must divide the result by two to account for the absence of overall phase ω in the special unitary group) and a 15-dimensional ε -ball has a volume of $\frac{\pi^{\frac{15}{2}}}{\Gamma(\frac{15}{2}+1)}\varepsilon^{15}$. \square

Remark 5.4.14. In the case of $\mathcal{C} + T$ circuits, it was established [96] that for single-qubit unitaries of determinant one, the lde k in the SU(2) representation was related to the T -count, which was one of $2k - 2$ or $2k$. Interestingly, this is *not* the case for the SU(4) representation of determinant one $\mathcal{C}_2 + CS$ operators. Indeed, no such simple relationship holds between the CS -count and the lde in the SU(4) representation in the general case. This is easy to check by generating random $\mathcal{C}_2 + CS$ circuits with determinant one and then checking the lde in both the SU(4) and SO(6) representations.

That being said, we can still determine bounds for the CS count using the lde in an operator's SU(4) representation. Examination of Eq. (5.12) implies that the

the k' of an $\text{SO}(6)$ representation for an lde k $\text{SU}(4)$ operator must be such that

$$k' \leq 2k + 2.$$

Likewise, close examination of Proposition 5.4.7 shows that every CS operator must be separated from one another by a Clifford with an lde of at most 2 in its unitary representation. Combining with the fact that the largest lde of an operator in \mathcal{C}_2 is 3, we see that

$$k' \geq \frac{k - 3}{2}.$$

Combining our inequalities, we have that the CS count k' for a special unitary operator with an lde k is bounded by

$$\frac{k - 3}{2} \leq k' \leq 2k + 2. \tag{5.30}$$

This means that the CS count of an operator always scales linearly with the lde of its unitary representation. For large k , most operators seem to be such that $\frac{5}{4}k \lesssim k' \lesssim \frac{4}{3}k$, though there are examples of operators with $k \approx k'$ or $2k \approx k'$.

5.5 Conclusion

We have described the first provably optimal compilation algorithm in terms of non-Clifford count for a fault-tolerant multi-qubit gate set. This establishes the existence of a unique normal form for $\mathcal{C}_2 + CS$ circuits. We show that this synthesis algorithm is computable in a time logarithmic in the gate-count of the original

circuit. Finally, we use a volume counting argument to show that ε -approximation of two-qubit unitaries will take at least $5 \log_2(1/\varepsilon)$ CS gates in the typical case. These results will form the basis of an *inexact* synthesis algorithm using this gate set for two-qubit circuits, which shall be developed in Chapter 6.

Looking ahead, the techniques used in this work can hopefully be used to develop optimal multi-qubit normal forms for other two-qubit gate sets such as two-qubit $\mathcal{C} + T$. Indeed, it can be shown using similar techniques to that of Chapter 4 that the $SO(6)$ representation of $\mathcal{C} + T$ operators are exactly the set of $SO(6)$ matrices with entries in the ring $\mathbb{Z}[1/\sqrt{2}]$. Looking further forward, there exist another exceptional isomorphism for $SU(8)$, which could prove useful in establishing a synthesis algorithm for 3-qubit circuits. Long term, these types of algorithms may well form the basis of quantum compilers.

Chapter 6: Clifford + Controlled Phase Inexact Synthesis ¹

6.1 Introduction

As discussed previously, the $\mathcal{C} + CS$ gate set is a fault-tolerant multi-qubit gate set. In Chapter 4, we demonstrated that any n -qubit unitary which is exactly expressible over this gate set is in the matrix group $\mathcal{U}_{2^n \times 2^n}(\mathbb{D}[i])$ up to a factor of ω . In Chapter 5, we then described a Controlled-Phase-count-optimal synthesis algorithm for the gate set. In analogue with past work [90, 92, 96–98, 100], we would now like to develop an algorithm for approximating *any* 2-qubit unitary using this gate set. In particular, we will show that we can synthesize any approximation for Pauli rotations, as was done in [99, 100]. We shall also comment on the synthesis of unitaries which are not Pauli-rotations.

6.2 Overview of Approximation Scheme

Initially, we restrict our attention to approximating Pauli rotations of the form

$$e^{-\frac{i\phi}{2}P}$$

¹This chapter is the preliminary workings of a forthcoming manuscript [147]

where P is any traceless, Hermitian two-qubit Pauli Operator. Up to an irrelevant phase of -1 , this set consists of 15 operators. Every element of this set is actually expressible as some Clifford conjugation of $Z \otimes \mathbb{1}$, $P = C(Z \otimes \mathbb{1})C^\dagger$, for some Clifford C , and so we can actually restrict this discussion to rotations about $Z \otimes \mathbb{1}$. This means we need to consider how to approximate the operator

$$U = e^{-\frac{i\phi}{2}Z \otimes \mathbb{1}} = \begin{bmatrix} e^{-\frac{i\phi}{2}} & 0 & 0 & 0 \\ 0 & e^{-\frac{i\phi}{2}} & 0 & 0 \\ 0 & 0 & e^{\frac{i\phi}{2}} & 0 \\ 0 & 0 & 0 & e^{\frac{i\phi}{2}} \end{bmatrix}. \quad (6.1)$$

By our result in Chapter 4, this means looking for some

$$\tilde{U} = \frac{1}{\sqrt{2^k}} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ u_{41} & u_{42} & u_{43} & u_{44} \end{bmatrix}$$

where $u_{jm} \in \mathbb{Z}[i]$ such that

$$\tilde{U}_{jm} = \frac{u_{jm}}{\sqrt{2^k}} \approx \begin{cases} e^{-\frac{i\phi}{2}} & j = m \leq 2 \\ e^{\frac{i\phi}{2}} & j = m > 2 \\ 0 & \text{otherwise} \end{cases}.$$

By our approximation sign here, we really mean that we need $\mathcal{N}(U - \tilde{U}) \leq \varepsilon$ for some matrix norm \mathcal{N} and $\varepsilon \in \mathbb{R}^+$ (strictly positive Reals). For simplicity, we shall choose \mathcal{N} to be the Frobenius norm, and so we have

$$\begin{aligned} \mathcal{N}(U - \tilde{U}) &= \sqrt{\text{Tr} \left[(U - \tilde{U})(U - \tilde{U})^\dagger \right]} \\ &= \sqrt{8 - \text{Tr} \left[\tilde{U}U^\dagger + U\tilde{U}^\dagger \right]} \\ &= \sqrt{8 - 2 \text{Re Tr} \left[\tilde{U}U^\dagger \right]}. \end{aligned}$$

Solving for $\text{Re Tr} \left[\tilde{U}U^\dagger \right]$ and bounding the result by the fact that $\varepsilon \mathcal{N}(U - \tilde{U}) \geq 0$, we have

$$1 \geq \frac{1}{4} \text{Re Tr} \left[\tilde{U}U^\dagger \right] \geq 1 - \frac{\varepsilon^2}{8}. \quad (6.2)$$

By virtue of choosing U as a rotation about $Z \otimes \mathbb{1}$, we see that the only contribution to Eq. (6.2) ends up being from the diagonal elements of \tilde{U} . Approximate synthesis of U will thus consist of two aspects:

1. Find a set of \tilde{U}_{jj} such that Eq. (6.2) is satisfied where the resulting matrix will use as few CS gates as possible.
2. Find a set of suitable \tilde{U}_{jl} for $j \neq l$ to make \tilde{U} unitary.

We begin by tackling Item 2 first, as it is the simpler of the two problems.

6.3 Unitary Templates and Lagrange Four-Squares

As in previous work [99,100], we will work in the special unitary representation when looking for approximations. Unlike these schemes, the lde in our representation is not a perfect metric for the final CS count – ultimately, this will cost our scheme optimality. Regardless, we still produce an asymptotically optimal inexact synthesis algorithm by using what we call a template.

Definition 6.3.1 (Template). Let $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$ for complex α , β , and γ .

Consider the operator

$$V = \begin{bmatrix} \alpha & 0 & -\beta^* & -\gamma^* \\ 0 & \alpha & \gamma & -\beta \\ \beta & -\gamma^* & \alpha^* & 0 \\ \gamma & \beta^* & 0 & \alpha^* \end{bmatrix} \quad (6.3)$$

which is unitary by construction. We call V a $Z \otimes \mathbb{1}$ rotation *template*.

Let us consider ε -approximation using our template for a $Z \otimes \mathbb{1}$ rotation. By Eqs. (6.1) to (6.3), we have the constraints

$$1 \geq \operatorname{Re} \left[\alpha e^{\frac{i\phi}{2}} \right] \geq 1 - \frac{\varepsilon^2}{8} \quad \text{and} \quad |\alpha|^2 \leq 1$$

which are pictorially represented in Fig. 6.1. Suppose we can find such an α with

$$\alpha = \frac{a + ib}{\sqrt{2}^k} \tag{6.4}$$

for $a, b \in \mathbb{Z}$ and $k \in \mathbb{N}$. Then finding an appropriate β and γ is easy as we can make use of Lagrange’s four-squares theorem [148]. This theorem guarantees a solution to the equation

$$w^2 + x^2 + y^2 + z^2 = n$$

for integral w, x, y, z for all integers $n \in \mathbb{N}$. In our case, setting $n = 2^k - a^2 - b^2$ and finding such a set of integers means we can set $\beta = \frac{w+ix}{\sqrt{2}^k}$ and $\gamma = \frac{y+iz}{\sqrt{2}^k}$ to ensure

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1.$$

Solutions to this so-called *Diophantine* equation can be computed using a randomized algorithm in a time $\mathcal{O}(\log^2(n))$ [148]. All that remains is to develop an approximation scheme for α .

6.4 Finding Approximations with Small Least Denominator Exponent

While it is true that the lde in the $SU(4)$ representation does not have a simple expression connecting it to the CS count of the respective operator, we can still bound its CS -count. In particular, using Eq. (5.30) we know that it should

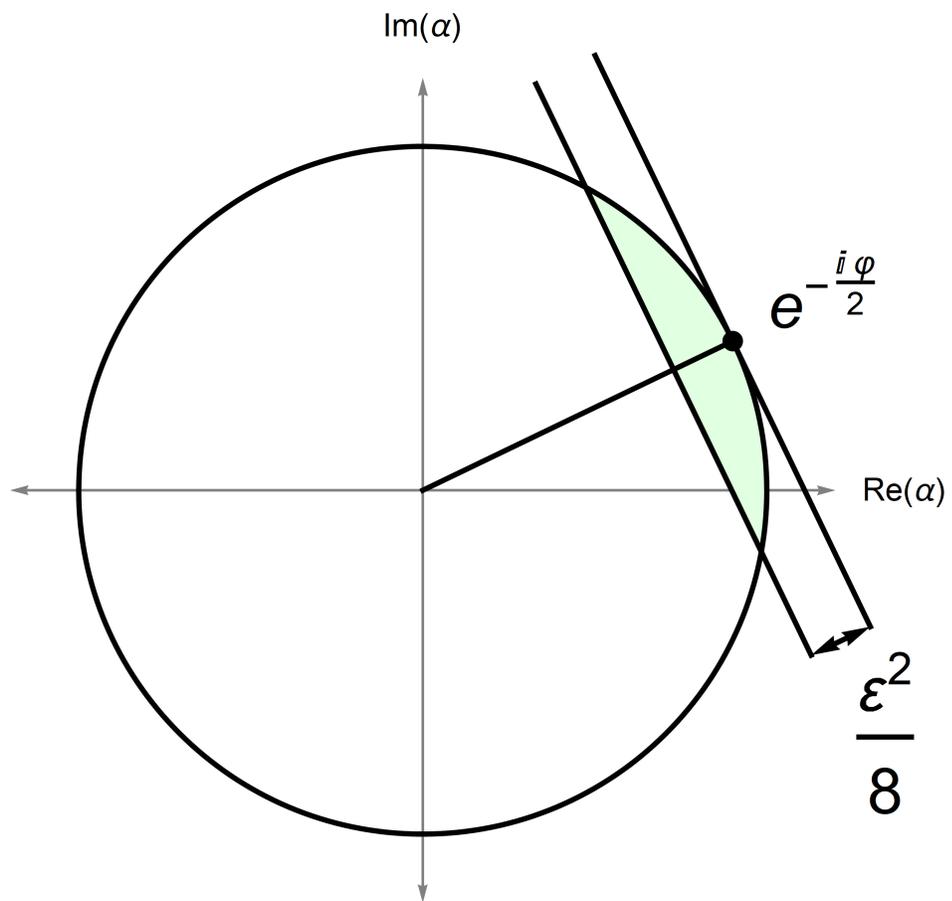


Figure 6.1: The acceptable values of α fall in the green region which is a segment of the unit disk. This region can be contained within a rotated rectangle which has a width of $\frac{\varepsilon^2}{8}$ and a height of approximately ε .

scale linearly with the lde in the unitary representation – hence we should look for a candidate α of the form in Eq. (6.4) that has the smallest lde possible. In particular, we can use the Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm [101] for ever-increasing k until we find some candidate for α . Given inputs ϕ and ε , we search for a 2-component vector of integers $\vec{\alpha} = [a, b]$ for which $\alpha = \frac{a+ib}{\sqrt{2}^k}$ is a valid solution using the following algorithm:

1. Let $k = -1$ and $\vec{\alpha}$ be unset
2. Compute the vector $\vec{t} = [\cos(\frac{i\phi}{2}), -\sin(\frac{i\phi}{2})]$
3. While $\vec{\alpha}$ is unset:
 - I. Increment k
 - II. Define $\mathcal{R}_\varepsilon^k = \left\{ \vec{u} \mid \vec{u} \cdot \vec{t} \geq \sqrt{2}^k \left(1 - \frac{\varepsilon^2}{8}\right) \text{ and } |\vec{u}| \leq \sqrt{2}^k \right\}$
 - III. Find a bounding parallelogram of integer vertex coordinates that contains $\mathcal{R}_\varepsilon^k$, call it $\mathcal{B}_\varepsilon^k$ and let the vectors which define two non-parallel sides of $\mathcal{B}_\varepsilon^k$ be \vec{v}_1 and \vec{v}_2
 - IV. Run the LLL lattice basis reduction algorithm on $[\vec{v}_1, \vec{v}_2]$ and let the result be $[\vec{v}'_1, \vec{v}'_2]$ which produces the invertible linear transformation \mathcal{M}
 - V. Compute $\mathcal{T}_\varepsilon^k = \mathcal{M}(\mathcal{B}_\varepsilon^k)$ which has the x domain $x_0 \leq x \leq x_1$.
 - VI. For each integer a'_j such that $x_0 \leq a'_j \leq x_1$:
 - i. Compute the smallest b'_j which could be in $\mathcal{T}_\varepsilon^k$ and define the vector $\vec{\lambda}' = [a'_j, b'_j]$

- ii. Compute $\vec{\lambda} = \mathcal{M}^{-1}\vec{\lambda}'$
 - iii. If $\vec{\lambda} \in \mathcal{R}_\varepsilon^k$, set $\vec{\alpha} = \vec{\lambda}$
4. Return $(\vec{\alpha}, k)$

The LLL lattice basis reduction algorithm effectively transforms the bounding region $\mathcal{B}_\varepsilon^k$, a long and skinny rotated parallelogram, to be a relatively “upright” parallelogram. By making the region upright, the number of values a_j which need to be tested is $\mathcal{O}(1)$, and so the performance of the algorithm scales roughly linearly with k . Rather than dive more into the details of this scheme, we present an alternate interpretation that shall make studying performance a little easier.

6.4.1 Alternate Algorithm for Finding Approximations

The major difficulty in finding approximations to α lies in the fact that the region in which acceptable solutions exist is much longer than it is wide, and is generally oriented at an angle. For fixed lde k , the total area of this region is $\mathcal{O}(2^k \varepsilon^3)$ as there is a tightly-bounding rectangle of width $\frac{\sqrt{2}^k \varepsilon^2}{8}$ and a height of $\sqrt{2}^k \varepsilon$. Consider the 2-dimensional grid of integers \mathbb{Z}^2 . Naively, we expect a solution when the area of our region is $\mathcal{O}(1)$ as that is the area of a unit cell of \mathbb{Z}^2 ; this corresponds to an expected $k \sim 3 \log_2 \left(\frac{1}{\varepsilon}\right)$. However, in the worst case we may not find a solution until the width of the rectangle is of size $\mathcal{O}(1)$, corresponding to $k \sim 4 \log_2 \left(\frac{1}{\varepsilon}\right)$. It shall turn out that our intuition holds in both cases. Indeed, using the same techniques as [104] would yield a valid α approximation with $k \sim 4 \log_2 \left(\frac{1}{\varepsilon}\right)$, and so we know this is an upper bound on our lde.

Because the rectangle is skinny (at most a width of $\mathcal{O}(1)$ before a solution is found), we only need to check a single y grid point for every x grid point which falls within the x -domain of the rectangle. However, the total number of x grid points to check is approximately $\sqrt{2}^k \varepsilon \sin\left(\frac{\phi}{2}\right)$ which is $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ in the worst case and $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}}\right)$ in the typical case. If we were able to re-orient our bounding rectangle such that each side is approximately perpendicular to the coordinate axes of the grid \mathbb{Z}^2 , it would be straightforward to identify points inside the region. Ideally, we'd be able to perform a pure rotation that orients this rectangle upright. However, this is impossible due to the fact that the only pure rotations that preserve \mathbb{Z}^2 grid points are those for $\frac{\pi}{2}$ rotations. We therefore need to consider more general transformations which *do* preserve our grid.

Affine transformations are transformation which preserve colinearity and distance ratios. Any affine transformation A with integer entries maps the grid \mathbb{Z}^2 onto itself. Furthermore, we can ensure this mapping is one-to-one if A is invertible over the integers – this corresponds only to those affine transformations of determinant ± 1 . Our goal then is as follows: find an invertible integral affine transformation that minimizes the x -domain of the bounding rectangle after the transformation.

The initial bounding rectangle can be described by two vectors which give the orientation of its sides. From Fig. 6.1, we can use geometry to determine that these

are

$$\begin{aligned}\vec{v}_1 &= \sqrt{2^k} \varepsilon \sqrt{1 - \frac{\varepsilon^2}{16}} \cdot \left[-\sin\left(\frac{\phi}{2}\right), \cos\left(\frac{\phi}{2}\right) \right] \approx \sqrt{2^k} \varepsilon \cdot \left[-\sin\left(\frac{\phi}{2}\right), \cos\left(\frac{\phi}{2}\right) \right] \\ \vec{v}_2 &= \frac{\sqrt{2^k} \varepsilon^2}{8} \cdot \left[\cos\left(\frac{\phi}{2}\right), \sin\left(\frac{\phi}{2}\right) \right]\end{aligned}$$

We will not consider translations of the grid \mathbb{Z}^2 , and so our affine transformations are representable as 2×2 matrices over the integers:

$$A = \begin{bmatrix} q & r \\ s & t \end{bmatrix}$$

with $\det A = \pm 1$. The width \mathcal{W} of the x -domain of our bounding parallelogram post-transformation is the maximum x -component of $D = \{|A(\vec{v}_1 - \vec{v}_2)|, |A(\vec{v}_1 + \vec{v}_2)|\}$. \mathcal{W} is upper-bounded by the x -component of

$$|A\vec{v}_1| + |A\vec{v}_2|$$

and so we can determine

$$\mathcal{W} \leq \sqrt{2^k} \varepsilon \left(\left| r \cos\left(\frac{\phi}{2}\right) - q \sin\left(\frac{\phi}{2}\right) \right| + \frac{\varepsilon}{8} \left| q \cos\left(\frac{\phi}{2}\right) + r \sin\left(\frac{\phi}{2}\right) \right| \right). \quad (6.5)$$

Thus, if we are able to find q, r such that $\frac{r}{q} \approx \tan\left(\frac{\phi}{2}\right)$ with $q \lesssim \frac{8}{\sqrt{2^k} \varepsilon^2}$ then we can ensure $\mathcal{W} \sim \mathcal{O}(1)$. The following discussion shall be restricted $0 \leq \frac{\phi}{2} \leq \frac{\pi}{4}$ as we can use the $\frac{\pi}{2}$ -rotation affine transformation and the $x \leftrightarrow y$ reflection operation to map

every angle into this domain.

By Dirichlet's approximation theorem [149], we know that for real numbers κ and $N \geq 1$, we can find integers q and $1 \leq \ell \leq N$ such that

$$|\kappa\ell - m| \leq \frac{1}{N}.$$

Observing that this roughly matches the form of Eq. (6.5), we can equate $\kappa \rightarrow \tan\left(\frac{\phi}{2}\right)$, $\ell \rightarrow r$, and $m \rightarrow q$ to show

$$\left| r \cos\left(\frac{\phi}{2}\right) - q \sin\left(\frac{\phi}{2}\right) \right| \leq \frac{\cos\left(\frac{\phi}{2}\right)}{N} \quad (6.6)$$

for some integral q and $r \leq q$ with $1 \leq q \leq N$. Such a q and r can be found using, for example, continued fractions or the aforementioned LLL lattice basis reduction scheme [149]. We then must choose N such that the second term of Eq. (6.5) is not too large. In particular, we have

$$\left| q \cos\left(\frac{\phi}{2}\right) + r \sin\left(\frac{\phi}{2}\right) \right| \leq q \cos\left(\frac{\phi}{2}\right) \left[1 + \frac{r}{q} \tan\left(\frac{\phi}{2}\right) \right]. \quad (6.7)$$

We can use Eq. (6.6) to show

$$\frac{r}{q} \leq \tan\left(\frac{\phi}{2}\right) + \frac{1}{qN}$$

and so we can simplify Eq. (6.7) to

$$\begin{aligned}
q \cos\left(\frac{\phi}{2}\right) \left[1 + \frac{r}{q} \tan\left(\frac{\phi}{2}\right)\right] &\leq q \cos\left(\frac{\phi}{2}\right) \left[1 + \tan^2\left(\frac{\phi}{2}\right) + \frac{\tan\left(\frac{\phi}{2}\right)}{qN}\right] \\
&\leq q \cos\left(\frac{\phi}{2}\right) \left[\sec^2\left(\frac{\phi}{2}\right) + \frac{\tan\left(\frac{\phi}{2}\right)}{qN}\right] \\
&\leq q \sec\left(\frac{\phi}{2}\right) + \frac{\sin\left(\frac{\phi}{2}\right)}{N}
\end{aligned} \tag{6.8}$$

We now consider two separate cases: when $\phi = 2 \arctan v$ for $v \in \mathbb{Q}$, and otherwise.

Suppose $\phi = 2 \arctan v$ for $v \in \mathbb{Q}$. Then we have $\frac{r}{q} = v$ and $\frac{1}{N} \rightarrow 0$. Using Eqs. (6.6) and (6.8) with these substitutions reduces Eq. (6.5) to

$$\mathcal{W} \leq \frac{\sqrt{2}^k \varepsilon^2 q}{8} \tag{6.9}$$

For a ϕ that does not have this property, we instead have upon rearrangement of Eq. (6.5)

$$\mathcal{W} \leq \frac{\sqrt{2}^k \varepsilon^{\frac{3}{2}} \cos\left(\frac{\phi}{2}\right)}{8} \left[\frac{8 + \varepsilon \tan\left(\frac{\phi}{2}\right)}{N\sqrt{\varepsilon}} + q\sqrt{\varepsilon} \sec^2\left(\frac{\phi}{2}\right) \right]. \tag{6.10}$$

Using that $q \leq N$, we can set $q \rightarrow N$ in Eq. (6.10) and minimize with respect to $N\sqrt{\varepsilon}$. This yields

$$\mathcal{W} \leq \frac{\sqrt{2}^k \varepsilon^{\frac{3}{2}} \sqrt{8 + \varepsilon \tan\left(\frac{\phi}{2}\right)}}{4} \leq \frac{3\sqrt{2}^k \varepsilon^{\frac{3}{2}}}{4} \text{ for } N = \sqrt{\frac{8}{\varepsilon} + \tan\left(\frac{\phi}{2}\right)} \cos\left(\frac{\phi}{2}\right). \tag{6.11}$$

In the case where our rational approximation of $\tan\left(\frac{\phi}{2}\right)$ produces some q such that

$q\sqrt{\varepsilon} \sim 1$, then Eq. (6.11) is a relatively good approximation for the width. However, if $q\sqrt{\varepsilon} \ll 1$ (alternatively, $q \ll N$), the width is much closer to the expression of Eq. (6.9). Regardless of the scaling of \mathcal{W} , we know that solutions will show up when $\mathcal{W} \gtrsim 1$ and $\frac{A}{\mathcal{W}} \gtrsim 1$. When Eq. (6.9) is the better approximation for the width \mathcal{W} , then \mathcal{W} is the limiting factor, with $\mathcal{W} \sim 1$ for $k \sim 4 \log_2 \left(\frac{1}{\varepsilon}\right)$. When Eq. (6.11) is on the other hand, both the width and area are $\mathcal{O}(1)$ when $k \sim 3 \log_2 \left(\frac{1}{\varepsilon}\right)$.

While we have described a method for finding a suitable q, r in our affine transformation A , we have said nothing of s, t . We know that A must have a determinant ± 1 , meaning we need

$$qt - rs = \pm 1.$$

Selecting the positive variant of this equation, we can always find such an s, t by using the extended Euclidean algorithm [150], which bounds $|s| \leq \frac{q}{2}$ and $|t| \leq \frac{r}{2}$. We can try and likewise bound the height \mathcal{H} of the resulting bounding parallelogram. We see that Eq. (6.5) holds under $\mathcal{W} \rightarrow \mathcal{H}$, $q \rightarrow s$, and $r \rightarrow t$. Again, we tackle these problems in the case when $\tan\left(\frac{\phi}{2}\right)$ is exactly rational and when it is not.

Beginning with the rational case with $\tan\left(\frac{\phi}{2}\right) = \frac{r}{q}$, we can exactly bound

$$\left|t \cos\left(\frac{\phi}{2}\right) - s \sin\left(\frac{\phi}{2}\right)\right| = \cos\left(\frac{\phi}{2}\right) \left|t - \frac{rs}{q}\right| = \frac{\cos\left(\frac{\phi}{2}\right)}{q}. \quad (6.12)$$

Evidently, t is an approximation of $s \tan\left(\frac{\phi}{2}\right)$ to $\frac{1}{q}$. Bounding the second term of our

equation, we have

$$\begin{aligned}
\left| s \cos\left(\frac{\phi}{2}\right) + t \sin\left(\frac{\phi}{2}\right) \right| &\leq \frac{\cos\left(\frac{\phi}{2}\right)}{q} |sq + tr| \\
&\leq \frac{\cos\left(\frac{\phi}{2}\right)}{2q} (q^2 + r^2) \\
&\leq \frac{\cos\left(\frac{\phi}{2}\right)}{2q} q^2 \sec^2\left(\frac{\phi}{2}\right) \\
&\leq \frac{q \sec\left(\frac{\phi}{2}\right)}{2}.
\end{aligned} \tag{6.13}$$

Combining Eqs. (6.12) and (6.13), we arrive at

$$\mathcal{H} \leq \frac{\sqrt{2}^k \varepsilon^{\frac{3}{2}}}{4} \left[\frac{4}{q\sqrt{\varepsilon} \sec\left(\frac{\phi}{2}\right)} + \frac{q\sqrt{\varepsilon} \sec\left(\frac{\phi}{2}\right)}{4} \right]. \tag{6.14}$$

In the irrational case (or more generally when $q\sqrt{\varepsilon} \sim 1$), we instead bound

$$\begin{aligned}
\left| s \tan\left(\frac{\phi}{2}\right) - t \right| &\leq \left| s \left(\frac{r}{q} - \frac{1}{qN} \right) - \left(\frac{sr+1}{q} \right) \right| \\
&\leq \frac{1}{q} \left| \frac{s}{N} + 1 \right| \\
&\leq \frac{1}{q} \left| \frac{q}{2N} + 1 \right| \\
&\leq \frac{3}{2q},
\end{aligned} \tag{6.15}$$

implying that t is an approximation to $s \tan\left(\frac{\phi}{2}\right)$ within $\frac{3}{2q}$. We can use the same analysis as the width under $\mathcal{W} \rightarrow \mathcal{H}$, $N \rightarrow \frac{2q}{3}$, and $q \rightarrow s \leq \frac{q}{2}$ in Eq. (6.10) to arrive

at

$$\mathcal{H} \leq \frac{\sqrt{2}^k \varepsilon^{\frac{3}{2}} \cos\left(\frac{\phi}{2}\right)}{16} \left[\frac{24 + 3\varepsilon \tan\left(\frac{\phi}{2}\right)}{q\sqrt{\varepsilon}} + q\sqrt{\varepsilon} \sec^2\left(\frac{\phi}{2}\right) \right]. \quad (6.16)$$

As we have that $q\sqrt{\varepsilon} \sim 1$ for this analysis, we can call $q = \lambda N$ for $\lambda \sim 1$ and further reduce this equation using our N value from Eq. (6.11) to

$$\mathcal{H} \leq \frac{\sqrt{3}\sqrt{2}^k \varepsilon^{\frac{3}{2}} \sqrt{8 + \varepsilon \tan\left(\frac{\phi}{2}\right)}}{16} \left[\frac{\sqrt{3}}{\lambda} + \frac{\lambda}{\sqrt{3}} \right]. \quad (6.17)$$

Using Eqs. (6.9), (6.11), (6.14) and (6.17) we can bound the ratio between the area of the transformed bounding parallelogram and the area of the smallest completely upright rectangle that encloses it. In both instances, we let $q = \lambda N$ for the N value in Eq. (6.11). For the rational case, we have

$$\frac{\mathcal{A}}{\mathcal{HW}} \geq \frac{1}{\cos\left(\frac{\phi}{2}\right) + \lambda^2 \left(\frac{\cos\left(\frac{\phi}{2}\right)}{2} + \varepsilon^2 \sin\left(\frac{\phi}{2}\right) \right)} \quad (6.18)$$

and for the irrational case, we have

$$\frac{\mathcal{A}}{\mathcal{HW}} \geq \frac{8}{8 + \varepsilon \tan\left(\frac{\phi}{2}\right)} \frac{\lambda}{3 + \sqrt{3}\lambda^2}. \quad (6.19)$$

When $\lambda \ll 1$, corresponding to an exact (or nearly-exact) rational approximation of $\tan\left(\frac{\phi}{2}\right)$, then Eq. (6.18) is close to one. Likewise when λ is not very close to 0, Eq. (6.19) is close to one. Thus, this affine transformation should produce a transformed bounding parallelogram that is not too dissimilar from an upright rectangle.

Now that we have a method for finding an affine transformation A that maps long and skinny angled rectangles onto relatively upright parallelograms, we can use it to find approximations for α . To summarize, our method for finding A given some angle ϕ and error ε is as follows:

Lemma 6.4.1. *There exists an algorithm to find the affine transformation A which has been described in Section 6.4.1.*

Proof. The algorithm is

1. Find a pure rotation/reflection in the integers that maps $-\frac{\phi}{2} \rightarrow \frac{\phi'}{2} \in [0, \frac{\pi}{4})$, call it A_R
2. Find integers q and r such that $\left| q \tan\left(\frac{\phi'}{2}\right) - r \right| \leq \frac{\sqrt{\varepsilon} \sec\left(\frac{\phi'}{2}\right)}{\sqrt{8 + \varepsilon \tan\left(\frac{\phi'}{2}\right)}}$ using, e.g., continued fractions
3. Solve $qt - rs = 1$ for integral s and t using the extended Euclidean algorithm
4. Set $A_V = \begin{bmatrix} q & r \\ s & t \end{bmatrix}$
5. Return $A_V A_R$

□

We can use this subroutine in a slightly amended algorithm for finding $\vec{\alpha} = [a, b]$ for which $\alpha = \frac{a+ib}{\sqrt{2}^k}$ is a valid approximation of $e^{-\frac{i\phi}{2}}$ for error ε :

Proposition 6.4.2. *There exists an algorithm to compute an approximation $\alpha = \frac{a+ib}{\sqrt{2}^k}$ to $e^{-\frac{i\phi}{2}}$ for $a, b \in \mathbb{Z}$ and $k \in \mathbb{N}$ such that $\operatorname{Re} \left[\alpha e^{-\frac{i\phi}{2}} \right] \geq 1 - \frac{\varepsilon^2}{8}$ in $\mathcal{O}(\log(\frac{1}{\varepsilon}))$*

arithmetic operations. When $\tan\left(\frac{\phi}{2}\right) \in \mathbb{Q}$, this yields an lde of $k \approx 4 \log_2\left(\frac{1}{\varepsilon}\right)$ and otherwise gives an lde of $k \approx 3 \log_2\left(\frac{1}{\varepsilon}\right)$

Proof. We amend our earlier algorithm using Lemma 6.4.1 to propose the following algorithm:

1. Let $k = -1$ and $\vec{\alpha}$ be unset
2. Compute the vector $\vec{t} = \left[\cos\left(\frac{i\phi}{2}\right), -\sin\left(\frac{i\phi}{2}\right)\right]$
3. Compute the affine transformation A for input angle $-\frac{\phi}{2}$ and error ε .
4. While $\vec{\alpha}$ is unset:
 - I. Increment k
 - II. Define $\mathcal{R}_\varepsilon^k = \left\{ \vec{u} \mid \vec{u} \cdot \vec{t} \geq \sqrt{2}^k \left(1 - \frac{\varepsilon^2}{8}\right) \text{ and } |\vec{u}| \leq \sqrt{2}^k \right\}$
 - III. Find the bounding rectangle of $\mathcal{R}_\varepsilon^k$, call it $\mathcal{B}_\varepsilon^k$
 - IV. Apply A to the vertices of $\mathcal{B}_\varepsilon^k$, yielding the transformed parallelogram $\mathcal{T}_\varepsilon^k = A(\mathcal{B}_\varepsilon^k)$ with x domain $x_0 \leq x \leq x_1$
 - V. For each integer a'_j such that $x_0 \leq a'_j \leq x_1$:
 - i. Compute the smallest b'_j which could be in $\mathcal{T}_\varepsilon^k$ and define the vector $\vec{\lambda}' = [a'_j, b'_j]$
 - ii. Compute $\vec{\lambda} = A^{-1}\vec{\lambda}'$
 - iii. If $\vec{\lambda} \in \mathcal{R}_\varepsilon^k$, set $\vec{\alpha} = \vec{\lambda}$
5. Return $(\vec{\alpha}, k)$

This algorithm is effectively the same as earlier, except with a more explicit construction of the transformation used. We note that the expected run-time of the algorithm is $\mathcal{O}\left(\log\left(\frac{1}{\varepsilon}\right)\right)$ as this is both the number of arithmetic operations required to compute the transformation A as well as the number of iterations of k , with a constant number of arithmetic operators required per k value. \square

6.5 Pauli-Rotation Approximations

We can now use our approximation algorithm in Proposition 6.4.2 along with our $Z \otimes \mathbb{1}$ rotation template and Lagrange four-squares algorithm covered in Section 6.3 to provide an approximation scheme for Pauli-rotations. Consider operators of the form Eq. (6.3) as an ε -approximation to a $Z \otimes \mathbb{1}$ rotation of ϕ . We can use the algorithm in Proposition 6.4.2 to find an approximation for α with the smallest lde, yielding

$$\alpha = \frac{a + ib}{\sqrt{2}^k}$$

for $k \approx 3 \log_2\left(\frac{1}{\varepsilon}\right)$ in the typical case and $k \approx 4 \log_2\left(\frac{1}{\varepsilon}\right)$ when $\tan\left(\frac{\phi}{2}\right) \in \mathbb{Q}$ with a small denominator. Then we can solve the Lagrange four-squares problem to find suitable β and γ , which likewise have an lde no more than k . We can then try and determine the lde of the resulting operator in the $SO(6)$ representation.

Lemma 6.5.1. *Every operator that is a $Z \otimes \mathbb{1}$ rotation template with lde $k \geq 2$ in the $SU(4)$ representation has an lde $2k - 2$ in the $SO(6)$ representation.*

Proof. Direct calculation for the $SO(6)$ representation of V from Eq. (6.3) using the

substitutions

$$\alpha = \frac{a + ib}{\sqrt{2^k}}, \quad \beta = \frac{c + id}{\sqrt{2^k}}, \quad \gamma = \frac{e + if}{\sqrt{2^k}}$$

yields an $\text{SO}(6)$ representation with entries from the following set \mathcal{S} :

$$\mathcal{S}_1 = \left\{ \pm \frac{xy}{\sqrt{2^{2k-2}}} \mid x \neq y \text{ and } x, y, \in \{a, b, c, d, e, f\} \right\},$$

$$\mathcal{S}_2 = \left\{ \frac{a^2 - b^2 - c^2 - d^2 - e^2 - f^2}{\sqrt{2^{2k}}}, \frac{a^2 - b^2 + c^2 + d^2 + e^2 + f^2}{\sqrt{2^{2k}}}, \right. \\ \frac{a^2 + b^2 - c^2 + d^2 + e^2 + f^2}{\sqrt{2^{2k}}}, \frac{a^2 + b^2 + c^2 + d^2 + e^2 - f^2}{\sqrt{2^{2k}}}, \\ \left. \frac{a^2 + b^2 + c^2 + d^2 - e^2 + f^2}{\sqrt{2^{2k}}}, \frac{a^2 + b^2 + c^2 - d^2 + e^2 + f^2}{\sqrt{2^{2k}}} \right\},$$

$$\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2.$$

Exactly four of $\{a, b, c, d, e, f\}$ must be odd for the $\text{SU}(4)$ to have an lde of $k \geq 2$.

This in turn implies that exactly 12 of the elements of \mathcal{S}_1 have an lde of $2k - 2$.

Furthermore, the elements of set \mathcal{S}_2 can have lde at most $2k - 2$, and therefore the

lde of the overall operator must be $2k - 2$. \square

Combining Proposition 6.4.2 and Lemma 6.5.1, we arrive at the following result.

Theorem 6.5.2. *There exists an algorithm to compute an ε approximation for any Pauli-rotation by angle ϕ . This algorithm uses $\mathcal{O}(\log(\frac{1}{\varepsilon}))$ arithmetic operations and yields approximations which use $6 \log(\frac{1}{\varepsilon}) + \mathcal{O}(1)$ CS operators in the typical case and $8 \log(\frac{1}{\varepsilon}) + \mathcal{O}(1)$ when $\tan(\frac{\phi}{2}) \in \mathbb{Q}$.*

Proof. By Proposition 6.4.2 and Lemma 6.5.1 along with our exact synthesis algorithm of Chapter 5. Note that these are written for the $Z \otimes \mathbb{1}$ Pauli-rotation; this may be extended to all possible Pauli-rotations by Clifford conjugation. \square

We can use this algorithm to likewise approximate any operator in $SU(4)$ by using the Pauli-rotation decomposition of $SU(4)$ [151]. Such a decomposition uses 15 Pauli rotations, giving us the following decomposition algorithm.

Theorem 6.5.3. *There exists an algorithm which computes an ε -approximation to any $U \in SU(4)$ that requires $90 \log\left(\frac{1}{\varepsilon}\right) + \mathcal{O}(1)$ CS operators in the typical case and $120 \log\left(\frac{1}{\varepsilon}\right) + \mathcal{O}(1)$ in the worst case. The run-time of this algorithm is $\mathcal{O}\left(\log\left(\frac{1}{\varepsilon}\right)\right)$.*

Proof. We will use Theorem 6.5.2 a total of (up to) 15 times in accordance with the Pauli-rotation decomposition of $SU(4)$. To explicitly construct this decomposition and find the 15 angles for the necessary rotations, we give the following decomposition algorithm.

Direct computation of the $SO(6)$ representation of the $Z \otimes \mathbb{1}$ Pauli-rotation by angle ϕ yields the matrix

$$\begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & \mathbb{1}_4 \end{bmatrix}.$$

Every other Pauli-rotation is of a similar form, a two-level matrix

$$V_{[\ell,m]}(\phi) = \begin{bmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{bmatrix}_{[\ell,m]}$$

for rows/columns ℓ and m . These operators generate $\text{SO}(6)$, with one such decomposition being

$$H_{1,6}H_{1,5}H_{1,4}H_{1,3}H_{1,2} \text{ for } H_{1,\ell} = V_{[\ell-1,\ell]}(\phi_{\ell,\ell})V_{[\ell-2,\ell-1]}(\phi_{\ell-1,\ell}) \cdots V_{[1,2]}(\phi_{1,\ell}),$$

effectively a Hausholder decomposition of U . Each angle is straightforward to compute, as the inverse of this decomposition gives a prescription for how to send any element of $\text{SO}(6)$ back to the identity. Each application of a V should then map the bottom-most entry of the leftmost non-unit-vector column to zero. Once our fifteen angles are found, we can approximate each V using Theorem 6.5.2 and an approximating error of $\frac{\varepsilon}{15}$. Multiplying our approximations together, we can then run the exact synthesis algorithm from Chapter 5 on our result to produce the ε -approximation to U . The scaling of both the CS -count and the run-time then come from both Theorem 6.5.2 and the run-time of our exact synthesis algorithm. \square

6.6 Conclusion

In Chapter 6, we have developed an inexact synthesis algorithm for the $\mathcal{C} + CS$ gate set. In the case of Pauli-rotations, we developed an algorithm that requires

$6 \log \left(\frac{1}{\varepsilon}\right) + \mathcal{O}(1)$ CS operators in the typical case and $8 \log \left(\frac{1}{\varepsilon}\right) + \mathcal{O}(1)$ in the worst case. We then described a synthesis algorithm which uses these Pauli-rotations as subroutines, achieving a CS -count of $90 \log \left(\frac{1}{\varepsilon}\right) + \mathcal{O}(1)$ in the typical case and $120 \log \left(\frac{1}{\varepsilon}\right) + \mathcal{O}(1)$ in the worst case.

It is not clear if the algorithm outlined here is truly optimal – indeed, it is likely not. The number-theoretic bound from Chapter 5 seems to suggest that the constant could be 5 instead of 6 in the typical case. Moreover, as we have restricted to Pauli-rotations, it may be possible to push this constant even lower. For example, using the same approximation outlined here, the constant can be lower-bounded to 3, which would match the performance of the Clifford+T gate set in the single-qubit case. Regardless, we know that this algorithm is at least asymptotically optimal.

Chapter 7: Conclusion

In this dissertation, we have developed improved quantum compiling techniques for both qudit and multi-qubit circuits, with a particular emphasis on the Clifford + CS gate set. In Chapter 3 we provide an algorithm which computes the analogue of a Matsumoto-Amano normal form for single-qudit $\mathcal{C} + T$ circuits, in turn showing it is both T -count optimal and unique. We also provide as an addendum that such Matsumoto-Amano normal forms exist in higher prime dimensions as well, though we do not prove uniqueness.

In Chapter 4, we move to characterizing circuits which correspond to the obvious subrings of $\mathcal{C} + T$ operators. We show that up to one ancilla, $2^n \times 2^n$ unitaries with entries in the rings \mathbb{D} , $\mathbb{D}[\sqrt{2}]$, $\mathbb{D}[\sqrt{-2}]$, and $\mathbb{D}[i]$ correspond to circuits over the gate set $\{X, CX, CCX\}$ appended with analogues of the Hadamard gate and an optional phase gate which both live in their respective rings. Encouraged by our characterization of $\mathbb{D}[i]$ which up to a phase is equivalent to the $\mathcal{C} + CS$ gate set, in Chapter 5 we develop a CS -count-optimal normal form and exact synthesis algorithm for two-qubit circuits over this gate set, the first optimal synthesis algorithm for a multi-qubit fault tolerant gate set. In Chapter 6, we show how this algorithm can be used to find asymptotically optimal approximations for both Pauli-rotations

and generic two-qubit unitaries.

We would like to highlight that a version of our rewrite algorithm for single-qutrit $\mathcal{C}+T$ operators has been implemented by Xiaoning Bian of Dalhousie University, which can be found on his homepage [126]. We also include in Appendix A the text version of some Mathematica software which we have provided at our github repository. This software provides tools for applying all of the algorithms described in Chapters 5 and 6.

Looking forward, we hope that the work contained in this dissertation encourages readers to continue pushing the limits of algorithms for quantum compiling. Many techniques in classical compiling such as peephole optimization rely on optimal compiling for smaller circuits. Expanding our ability to optimally compile circuits for few-qubit unitaries would be a boon to both near- and far-term application. With the advent of quantum supremacy [11], we hope to see wider adoption of these synthesis techniques to improve the circuits which our near-term devices use.

Appendix A: Software Package for the Clifford + Controlled-Phase Gate Set

In this appendix, we supply the printed version of our software package available from our github repository. This package is suitable for performing both exact and inexact synthesis of two-qubit Clifford+ CS operators.

GuassSynth.m -- The Two-Qubit Clifford + CS Circuit Synthesis Package

Written and Maintained by Andrew Glaudell

The GuassSynth.m package is a package for quantum compiling on two qubits using the Clifford group and the Controlled-Phase gate CS. The circuits which are exactly expressible over this gate set constitute every 4x4 unitary matrix which can be written as a matrix of Gaussian integers divided by some non-negative integer power of $2^{1/2}$ -- hence the package name. In this package, we supply a number of functions for performing quantum circuit synthesis on this gate set, both in the exact and approximate case. The algorithms in this package are based off of the work of Andrew Glaudell, Julien Ross, Matthew Amy, and Jake Taylor, and for details related to how these algorithms were developed, I suggest reading the articles [1-3] in the sources section below.

Package Details

Copyright © 2019 Andrew Glaudell

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Package Version: 1.0

Written for Mathematica Version: 12.0

History:

1.0 - Initial version, completed 11/4/2019

Keywords: Quantum Compiling, Quantum Circuit Synthesis, Clifford Group, Controlled Phase Gate, Normal Forms, Exact Synthesis, Approximate Synthesis

Sources:

[1] Matt Amy, Andrew Glaudell, and Neil J. Ross. Number-theoretic characterizations of some restricted clifford+t circuits. Upcoming publication, preprint available from arXiv:1908.06076, 2019.

[2] Andrew Glaudell, Neil J. Ross, and Jacob M. Taylor. Optimal two-qubit circuits for universal fault-tolerant quantum computation. Upcoming publication, 2019.

[3] Andrew Glaudell. Inexact synthesis of pauli-rotations with the fault-tolerant clifford + controlled-phase gate set. Upcoming publication, in preparation, 2019.

Warnings:

I have used a fair amount of input checking so that functions only accept inputs of the appropriate form. This comes at the cost of some speed -- that being said, these checks cause constant overhead, and so their performance impact is worth it to prevent some erroneous calculation from being carried out. If you don't care about this input checking, one could relatively easily define their own functions from my own internal ones to slightly speed up their performance.

Limitations:

This package is only intended for usage on two-qubit circuits. To perform circuit synthesis on larger circuits, I suggest loading this package and using these functions as subroutines.

Discussion:

Rather than describe these algorithms in detail here, I differ to the sources [1-3] listed above or the function descriptions.

Requirements:

None

```
BeginPackage["GaussSynth"];
```

Function Usage

```
GaussSynth::usage = "GaussSynth is a package for quantum compiling for two-qubit circuits";

Id::usage = "Id is the 4x4 Identity Matrix.";
X1::usage = "X1 is the unitary representation of the X⊗I gate.";
X2::usage = "X2 is the unitary representation of the I⊗X gate.";
Z1::usage = "Z1 is the unitary representation of the Z⊗I gate.";
Z2::usage = "Z2 is the unitary representation of the I⊗Z gate.";
W::usage = "W is the unitary representation of the primitive 8th root of unity ω.";
H1::usage = "H1 is the unitary representation of the H⊗I gate.";
H2::usage = "H2 is the unitary representation of the I⊗H gate.";
S1::usage = "S1 is the unitary representation of the S⊗I gate.";
S2::usage = "S2 is the unitary representation of the I⊗S gate.";
CZ::usage = "CZ is the unitary representation of the CZ gate.";
CNOT12::usage = "CNOT12 is the unitary representation of the CNOT gate with control qubit 1 and target qubit 2";
CNOT21::usage = "CNOT21 is the unitary representation of the CNOT gate with control qubit 2 and target qubit 1";
EX::usage = "EX is the unitary representation of the SWAP (Exchange) gate.";
CS::usage = "CS is the unitary representation of the CS gate.";

U4ToS06::usage = "U4ToS06[U] maps the 4x4 unitary U to the equivalent SO(6) representation";

IdS06::usage = "Id is the 6x6 Identity Matrix.";
X1S06::usage = "X1S06 is the SO(6) representation of the X⊗I gate.";
X2S06::usage = "X2S06 is the SO(6) representation of the I⊗X gate.";
Z1S06::usage = "Z1S06 is the SO(6) representation of the Z⊗I gate.";
Z2S06::usage = "Z2S06 is the SO(6) representation of the I⊗Z gate.";
IS06::usage = "IS06 is the SO(6) representation of the complex phase I. As SU(4) is a double cover of SO(6), we have  $I^2 = IdS06$ .";
H1S06::usage = "H1S06 is the SO(6) representation of the H⊗I gate.";
H2S06::usage = "H2S06 is the SO(6) representation of the I⊗H gate.";
S1S06::usage = "S1S06 is the SO(6) representation of the S⊗I gate.";
S2S06::usage = "S2S06 is the SO(6) representation of the I⊗S gate.";
CZS06::usage = "CZS06 is the SO(6) representation of the CZ gate. Note that this gate is not special unitary, but is by a primitive 8th root of unity ω before performing the transformation.";
CNOT12S06::usage = "CNOT12S06 is the SO(6) representation of the CNOT gate with control qubit 1 and target qubit 2. Note that this gate is not special unitary, and so we multiply by a primitive 8th root of unity ω before performing the transformation.";
CNOT21S06::usage = "CNOT21S06 is the SO(6) representation of the CNOT gate with control qubit 2 and target qubit 1. Note that this gate is not special unitary, and so we multiply by a primitive 8th root of unity ω before performing the transformation.";
```


If U is an element of $U(4)$, the result is a $U(4)$ representation of a Clifford + CS circuit. Note that the Frobenius distance between U and the representation of a Clifford + CS circuit. Note that the Frobenius distance between U and the representation of a Clifford + CS circuit. Note that the Frobenius distance between U and the representation of a Clifford + CS circuit.

`ApproximateSequence::usage = "ApproximateSequence[U,ε,options] finds a normalized sequence (in the Unitary representation and up to an irrelevant phase) for the input U. U may be a string unless otherwise specified in the options. The options for the \"OutputType\" or \"SyllableType\" are \"Normal\" or \"Asymmetric\", and the options for \"IfGaussianDecomposition\" are \"Normal\" or \"Asymmetric\".`

Possible Errors

```
General::invldopt = "Option `2` for function `1` received invalid value `3`";
U4ToS06::notunitary = "The argument must be a 4x4 unitary matrix.";
FromSequence::notstring = "You have not entered a string.";
FromList::notagate = "The string `1` is not one of {\"W\", \"S1\", \"S2\", \"H1\", \"H2\"},
  You may have forgotten a space between gate names or used a name for a gate which is
  not a gate name";
FromHexDec::invalidnumber = "The string `1` is not a valid hexadecimal representation. Make
  sure the string is a valid hexadecimal representation, and that your syllables only take
  characters from the set of {\"0\"-\"9\", \"A\"-\"F\"}";
CliffordQ::notacircuit = "You have not entered a string of operators, a valid hexadecimal
  representation of a Clifford operator";
CliffordSynth::notaclifford = "Your input is not a Clifford operator in string form, a valid
  hexadecimal representation of a Clifford operator";
GaussianQ::notacircuit = "You have not entered a string of operators, a valid hexadecimal
  representation of a Clifford operator";
NormIt::badopt = "The option `1` is not valid for `2`.";
NormIt::notacircuit = "You have not entered a string of operators, a valid hexadecimal, a
  valid hexadecimal representation of a Clifford operator";
FrobeniusDistance::notequidimensionalmatrices = "Your inputs are not two matrices of equal
  size";
CandidateFinder::notreals = "Your inputs are not two real numbers";
PauliRotation::notreals = "Your input does not include two real numbers";
PauliRotation::invldstring = "Your input does not include one string from the set of {\"XI\",
  \"YX\", \"ZX\", \"XY\", \"YY\", \"ZY\", \"XZ\", \"YZ\", or \"ZZ\"}.";
PauliDecomposition::notanoperator = "Your input is neither an element of U(4) or SO(6) or
  a Clifford operator";
ApproximateOp::notreal = "Your error tolerance is not a real number.";
ApproximateOp::notanoperator = "Your input is neither an element of U(4) or SO(6) and so
  is not a Clifford operator";

Begin["`Private`"];

```

Function Definitions

Functions for option checking

These functions will be used to check options for functions which accept them. For each such function, we must define a test[f,op] function for a particular option type op of function f. Credit for this code snippet goes to Mr. Wizard in the Stack Overflow post <https://mathematica.stackexchange.com/questions/116623>.

```

optsMsg[f_][op_, val_] :=
  test[f, op][val] || Message[General::invldopt, f, op, val];

Attributes[optsCheck] = {HoldFirst};

optsCheck @ head_[___, opts : OptionsPattern[]] :=
  And @@ optsMsg[head] @@@ FilterRules[{opts}, Options @ head];

```

Constants and Single-Qubit operators

For internal use only.

```

 $\omega$  = (1+I)/Sqrt[2];
 $\zeta$  = Exp[I*Pi/8];
s = DiagonalMatrix[{1, I}];
h = 1/Sqrt[2]*{{1, 1}, {1, -1}};
x = PauliMatrix[1];
z = PauliMatrix[3];

```

Unitary Representations of Two-qubit Clifford + CS operators

These operators are exported to the user as 4x4 matrices in the standard Mathematica format.

```

Id = IdentityMatrix[4];
W =  $\omega$ *Id;
S1 = KroneckerProduct[s,IdentityMatrix[2]];
S2 = KroneckerProduct[IdentityMatrix[2],s];
H1 = KroneckerProduct[h,IdentityMatrix[2]];
H2 = KroneckerProduct[IdentityMatrix[2],h];
CZ = DiagonalMatrix[{1,1,1,-1}];
CNOT12 = {
  {1,0,0,0},
  {0,1,0,0},
  {0,0,0,1},
  {0,0,1,0}
};
CNOT21 = {
  {1,0,0,0},
  {0,0,0,1},
  {0,0,1,0},
  {0,1,0,0}
};
EX = {
  {1,0,0,0},
  {0,0,1,0},
  {0,1,0,0},
  {0,0,0,1}
};
CS = DiagonalMatrix[{1,1,1,I}];
X1 = KroneckerProduct[x,IdentityMatrix[2]];
X2 = KroneckerProduct[IdentityMatrix[2],x];
Z1 = KroneckerProduct[z,IdentityMatrix[2]];
Z2 = KroneckerProduct[IdentityMatrix[2],z];

```

Checks For U(4) and SO(6)

These Boolean functions determine whether an operator is an element of U(4) or SO(6), respectively.

```

U4Q[U_] := UnitaryMatrixQ[U] && (Dimensions[U] == {4,4});
SO6Q[O_] := OrthogonalMatrixQ[O] && (Dimensions[O] == {6,6}) && (Det[O] == 1);

```

The $SU(4) \cong SO(6)$ Isomorphism

These definitions and functions allow one to compute the SO(6) representation of an element of U(4) (up to a phase).

Rules for Inner Products of Wedge Products

Defined using the unassigned Mathematica symbols of \wedge , \langle , and \rangle .

```

⟨a_,0_,b_*c_⟩:=b*⟨a,0,c⟩;
⟨a_*b_,0_,c_⟩ := Conjugate[a]*⟨b,0,c⟩;
⟨a_+b_,0_,c_⟩:=⟨a,0,c⟩+⟨b,0,c⟩;
⟨a_,0_,b_+c_⟩:=⟨a,0,b⟩+⟨a,0,c⟩;
⟨x_∧y_,0_,u_∧v_⟩ := (Conjugate[x].0.u)*(Conjugate[y].0.v) - (Conjugate[x].0.v)*(Conjugate[y].0.u);

```

Orthonormal Basis for Subscript[C, 6]

This basis is such that computing the above inner products for an element of $U(4, \mathbb{C})$ will produce an element of $SO(6, \mathbb{R})$. Moreover, the representations for Clifford + CS operators are easy to work with in this basis.

```

Basis6 = {
  (1/Sqrt[2]) * (UnitVector[4,1]^UnitVector[4,2] - UnitVector[4,3]^UnitVector[4,4]),
  (1/Sqrt[2]) * (UnitVector[4,1]^UnitVector[4,2] + UnitVector[4,3]^UnitVector[4,4]),
  (1/Sqrt[2]) * (UnitVector[4,2]^UnitVector[4,3] - UnitVector[4,1]^UnitVector[4,4]),
  (1/Sqrt[2]) * (UnitVector[4,2]^UnitVector[4,3] + UnitVector[4,1]^UnitVector[4,4]),
  (1/Sqrt[2]) * (UnitVector[4,2]^UnitVector[4,4] - UnitVector[4,3]^UnitVector[4,1]),
  (1/Sqrt[2]) * (UnitVector[4,2]^UnitVector[4,4] + UnitVector[4,3]^UnitVector[4,1])
};

```

Calculating the SO(6) Representation for an element of SU(4) (and from U(4) up to a phase)

Functions for mapping elements of SU(4) to SO(6) and U(4) to SO(6) (by converting that element of U(4) to an element of SU(4)).

```

SU4ToSO6[U_] := Table[Simplify[⟨Basis6[[i]],U,Basis6[[j]]⟩],{i,1,6},{j,1,6}];
U4ToSO6[U_];U4Q[U] := SU4ToSO6[1/(Det[U])^(1/4)*U];
U4ToSO6[U_] := (
  Message[U4ToSO6::notunitary];
  $Failed
);

```

SO(6) Representations of Two-qubit Clifford + CS operators

Calculated using our transformations. These operators are exported to the user as 6x6 matrices in the standard Mathematica format. Note that we have to multiply by overall phases to ensure that the transformation uses an element of SU(4).

```

IdS06 = SU4ToS06[Id];
IS06 = SU4ToS06[W.W];
H1S06 = SU4ToS06[H1];
H2S06 = SU4ToS06[H2];
S1S06 = SU4ToS06[ $\omega^{-1}$ *S1];
S2S06 = SU4ToS06[ $\omega^{-1}$ *S2];
CZS06 = SU4ToS06[ $\omega^{-1}$ *CZ];
CNOT12S06 = SU4ToS06[ $\omega^{-1}$ *CNOT12];
CNOT21S06 = SU4ToS06[ $\omega^{-1}$ *CNOT21];
EXS06 = SU4ToS06[ $\omega^{-3}$ *EX];
CSS06 = SU4ToS06[ $\zeta^{-1}$ *CS];
X1S06 = SU4ToS06[X1];
X2S06 = SU4ToS06[X2];
Z1S06 = SU4ToS06[Z1];
Z2S06 = SU4ToS06[Z2];

```

Custom Representations of SO(6) Clifford + CS operators

Our synthesis algorithms will use a custom data type for the SO(6) representation of a Clifford + CS operator. The basic data structure of this special representation is as follows:

$$\{k, M\} := 2^{-(k/2)} \cdot M$$

This allows easy tracking of the lde. They are packed in SparseArrays to help make things even a little faster, as every Clifford is just a permutation matrix. These representations are for internal use only.

Switching between the standard representation for a 6x6 matrix and a representation specifically for Clifford + CS operators.

Functions for converting to and from the special representation.

```

SO6ToSpecialRep[M_] := Module[{LDE, IntegerMat},
  LDE = Max[Map[Simplify[Log[Sqrt[2], Denominator[#]]]&, Flatten[Simplify[M]]]];
  IntegerMat = Simplify[(Sqrt[2]^LDE)*M];
  {LDE, SparseArray[IntegerMat]}
];
SpecialRepToSO6[{k_, M_}] := Simplify[1/Sqrt[2]^k*Normal[M]];

```

Special Representations for SO(6) Clifford + CS operators

```

IdSp = S06ToSpecialRep[IdS06];
ISp = S06ToSpecialRep[IS06];
H1Sp = S06ToSpecialRep[H1S06];
H2Sp = S06ToSpecialRep[H2S06];
S1Sp = S06ToSpecialRep[S1S06];
S2Sp = S06ToSpecialRep[S2S06];
CZSp = S06ToSpecialRep[CZS06];
CNOT12Sp = S06ToSpecialRep[CNOT12S06];
CNOT21Sp = S06ToSpecialRep[CNOT21S06];
EXSp = S06ToSpecialRep[EXS06];
CSSp = S06ToSpecialRep[CSS06];
X1Sp = S06ToSpecialRep[X1S06];
X2Sp = S06ToSpecialRep[X2S06];
Z1Sp = S06ToSpecialRep[Z1S06];
Z2Sp = S06ToSpecialRep[Z2S06];

```

Basic Matrix Operations for the Special Representation

These internal functions are used to reduce the denominator exponent to the lde, multiply operators in the special representation, and invert the special representation.

```

KReduceOnce[{{k_,a_}}]:=If[AllTrue[a,EvenQ,2] && k>1,{k-2,a/2},{k,a}];
KReduce[o_]:=FixedPoint[KReduceOnce,o];

Dot2Sp[{{k1_,a1_},{k2_,a2_}}]:=KReduce[{{k1+k2,a1.a2}}];
DotSp[x_]:=Fold[Dot2Sp,IdSp,{x}];

InvSp[{{k_,a_}}]:={k,Transpose[a]};

```

Functions for Residues Modulo 2 and Finding Paired Matrix Rows

These functions are used for finding paired rows in operators in the special representation.

```

PatternMats[{{k_,a_}}]:=Mod[a,2];

MatrixRowPairs[list_]:=GatherBy[
  Range@Length[Normal[list]],
  Normal[list][[#]]&
];

RowPairs[x_]:=Sort@MatrixRowPairs@PatternMats@x

```

String Reading

Functions for reading in a string of operators. The string is always read in as an element of U(4).

```

FromSequence[str_String] := Module[{strlist},
  strlist = StringSplit[str];
  FromList[strlist]
];
FromSequence[x_] := (
  Message[FromSequence::notstring];
  $Failed
);

FromList[{}] = Id;
FromList[{"W"}] = W;
FromList[{"S1"}] = S1;
FromList[{"S2"}] = S2;
FromList[{"H1"}] = H1;
FromList[{"H2"}] = H2;
FromList[{"CZ"}] = CZ;
FromList[{"CS"}] = CS;
FromList[{"EX"}] = EX;
FromList[{"X1"}] = X1;
FromList[{"X2"}] = X2;
FromList[{"Z1"}] = Z1;
FromList[{"Z2"}] = Z2;
FromList[{str_}] := (
  Message[FromList::notagate,str];
  $Failed
);
FromList[{h_,t_}] := FromList[{h}] . FromList[{t}];

```

Hexadecimal Representations

We shall use strings of signed hexadecimal integers to represent Clifford + CS operators. This form is much more compact than, for example, the string form of an operator. The string must be of the following form:

$$\backslash(\epsilon|-)(1-9|a-f)^* 00000 C \backslash$$

where C is the hexadecimal representation of an integer from 1 to 92160.

```

ValidHexDecQ[num_String] := Module[{separator,typenum,syllabletype,syllablescliffordok,s}
  separator = StringCount[num,"00000"] == 1;
  typenum = StringCount[num,"-"];
  syllabletype = (typenum == 0) || ((typenum == 1) && StringStartsQ[num,"-"]);
  syllablescliffordok = If[
    separator && syllabletype,
    split = StringSplit[num,{"00000","-"}];
    {syllables,clifford} = If[Length[split] > 1,split,{"",split[[1]]}];
    syllablescharacterlist = Union[Characters[syllables]];
    syllablesok = SubsetQ[{"1","2","3","4","5","6","7","8","9","a","b","c","d","e","f"}

```

```

cliffordcharacterlist = Union[Characters[clifford]];
cliffordpossible = (StringLength[clifford]) <= 5 && SubsetQ[{"0","1","2","3"}
cliffordok = If[
  cliffordpossible,
    1 <= FromDigits[clifford,16] <= 92160,
    False
];
syllablesok && cliffordok,
False
];
syllablescliffordok
];
ValidHexDecQ[U_] := False

PhaseSet = {Id,MatrixPower[W,4],W,MatrixPower[W,5]};
L1Set = {Id,H1,S1.H1.MatrixPower[W,7]};
L2Set = {Id,H2,S2.H2.MatrixPower[W,7]};
CZSet = {Id,CZ.MatrixPower[W,7]};
S1Set = {Id,S1.MatrixPower[W,7]};
P1Set = {Id,X1,X1.Z1,Z1};
S2Set = {Id,S2.MatrixPower[W,7]};
P2Set = {Id,X2,X2.Z2,Z2};
SWAPSet = {Id,EX.MatrixPower[W,5]};
ISet = {Id,MatrixPower[W,2]};

CliffordFromNumber[cliffnum_] := Module[{digits,cz,l1,l2,index},
  digits = PadLeft[IntegerDigits[cliffnum-1,MixedRadix[{{4,10,3,3,2,4,2,4,2,2}}],10];
  cz = Unitize[digits[[2]]];
  l2 = Mod[digits[[2]]-cz,3];
  l1 = (digits[[2]]-l2-cz)/3;
  index = Join[{digits[[1]],l1,l2,cz},digits[[3;]-1]] + 1;
  PhaseSet[[index[[1]]] . L1Set[[index[[2]]] . L2Set[[index[[3]]] . CZSet[[index[[4]]]
  L1Set[[index[[5]]] . L2Set[[index[[6]]] . S1Set[[index[[7]]] . P1Set[[index[[8]]]
  S2Set[[index[[9]]] . P2Set[[index[[10]]] . SWAPSet[[index[[11]]] . ISet[[index[[12]]]
];

FromHexDec[str_String;/ValidHexDecQ[str]] := Module[{syllablelist,split,syllablestr,cliff},
  syllablelist = If[StringStartsQ[str,"-"],SyllableListAsymmetric[All,1],SyllableList];
  split = StringSplit[str,{"0000","-"}];
  {syllablestr,cliffstr} = If[Length[split] > 1,split,{"",split[[1]]}];
  cliffnum = FromDigits[cliffstr,16];
  cliff = CliffordFromNumber[cliffnum];
  syllables = Map[syllablelist[FromDigits[#,16]]&,Characters[syllablestr]];
  Simplify[Dot @@ (Append[syllables,cliff])]
];
FromHexDec[str_] := (
  Message[FromHexDec::invalidnumber,str];
  $Failed
);

```

The Two-Qubit Clifford Group

Here we develop some basic functions for the two-qubit Clifford group.

Identifying if an operator is a Clifford and if two operators are Clifford-similar

Boolean functions for checking if an operator is a Clifford or if two operators are Clifford-similar.

```
CliffordQ[U_;/;U4Q[U]] := (Det[U]^2 == 1) && AllTrue[Flatten[U4ToS06[U]], IntegerQ];
CliffordQ[U_;/;S06Q[U]] := AllTrue[Flatten[U], IntegerQ];
CliffordQ[U_String;/;ValidHexDecQ[U]] := CliffordQ[FromHexDec[U]];
CliffordQ[U_String] := CliffordQ[FromSequence[U]];
CliffordQ[U_] := (
  Message[CliffordQ::notacircuit];
  $Failed
);

RightCliffordSimilar[U_String;/;ValidHexDecQ[U], V_String;/;ValidHexDecQ[V]] := RightCliffordSimilar[FromSequence[U], FromSequence[V]];
RightCliffordSimilar[U_String, V_String] := RightCliffordSimilar[FromSequence[U], FromSequence[V]];
RightCliffordSimilar[U_, V_] := CliffordQ[ConjugateTranspose[U].V];

LeftCliffordSimilar[U_String;/;ValidHexDecQ[U], V_String;/;ValidHexDecQ[V]] := LeftCliffordSimilar[FromSequence[U], FromSequence[V]];
LeftCliffordSimilar[U_String, V_String] := LeftCliffordSimilar[FromSequence[U], FromSequence[V]];
LeftCliffordSimilar[U_, V_] := CliffordQ[V.ConjugateTranspose[U]];
```

Synthesis of Clifford Circuits (with at most 1 CZ gate and 1 SWAP gate)

Rather than carry around an explicit lookup table with 92160 elements in the $U(4)$ case and 23040 elements in the $SO(6)$ case, we instead will use a synthesis algorithm based on the special representation; this takes advantage of the sparsity of Cliffords in this representation. We implicitly define our regular Clifford synthesis algorithm, `CliffordSynth`, in terms of this other algorithm, to be defined below.

```

PhaseFixU4[{str_, power_, SUcliffnum_}, U_] := Module[{unitary, phase, Wpower, diff, Wcosetnumb
  unitary = FromSequence[str];
  phase = Simplify[(U. ConjugateTranspose[unitary])[[1,1]]];
  Wpower = Mod[Log[(1+I)/Sqrt[2], phase], 8];
  diff = Mod[Wpower - power, 8];
  Wcosetnumber = Which[
    (diff == 0) || (diff == 2), 0,
    (diff == 4) || (diff == 6), 1,
    (diff == 1) || (diff == 3), 2,
    True, 3
  ];
  {IntegerString[23040*Wcosetnumber + FromDigits[SUcliffnum, 16], 16], str<>StringRepeat[
];
PhaseFixS06[{str_, power_, SUcliffnum_} := {SUcliffnum, str<>StringRepeat[" W", power]};

CliffordSynth[U_String;/;ValidHexDecQ[U]] := Module[{unitary},
  unitary = FromSequence[U];
  CliffordSynth[unitary]
];
CliffordSynth[U_String] := Module[{unitary},
  unitary = FromSequence[U];
  CliffordSynth[unitary]
];
CliffordSynth[U_;/; (U4Q[U] && CliffordQ[U])] := Module[{orthogonal, synth},
  orthogonal = U4ToS06[U];
  synth = CliffordSynthSp[{0, SparseArray[orthogonal]}];
  PhaseFixU4[synth, U]
];
CliffordSynth[U_;/; CliffordQ[U]] := PhaseFixS06[CliffordSynthSp[{0, SparseArray[U]}]];
CliffordSynth[U_] := (
  Message[CliffordSynth::notaclifford];
  $Failed
);

```

Clifford Group in the Special representation

Explicit CliffordSynthSp algorithm. We perform the synthesis by decomposing an element of the signed permutation group in terms of a generating set which is constructed from basic Clifford operators in the SO(6) representation.

```

L1Cosets = {
  {IdSp, "", 0, 0},
  {H1Sp, "H1 ", 0, 1},
  {DotSp[S1Sp, H1Sp], "S1 H1 ", 7, 2}
};
L2Cosets = {
  {IdSp, "", 0, 0},
  {H2Sp, "H2 ", 0, 1},
  {DotSp[S2Sp, H2Sp], "S2 H2 ", 7, 2}
};

```

```

};
CZSets = {
  {IdSp, "", 0, 0},
  {CZSp, "CZ ", 7, 1}
};
S1Sets = {
  {IdSp, "", 0, 0},
  {S1Sp, "S1 ", 7, 1},
};
Pauli1Sets = {
  {IdSp, "", 0, 0},
  {X1Sp, "X1 ", 0, 1},
  {DotSp[X1Sp, Z1Sp], "X1 Z1 ", 0, 2},
  {Z1Sp, "Z1 ", 0, 3}
};
S2Sets = {
  {IdSp, "", 0, 0},
  {S2Sp, "S2 ", 7, 1},
};
Pauli2Sets = {
  {IdSp, "", 0, 0},
  {X2Sp, "X2 ", 0, 1},
  {DotSp[X2Sp, Z2Sp], "X2 Z2 ", 0, 2},
  {Z2Sp, "Z2 ", 0, 3}
};
SwapSets = {
  {IdSp, "", 0, 0},
  {EXSp, "EX ", 5, 1}
};
ISets = {
  {IdSp, "", 0, 0},
  {ISp, "", 2, 1}
};

CliffordQSp[U_] := Module[{k, M},
  {k, M} = KReduce[U];
  {k == 0} && OrthogonalMatrixQ[M] && (Det[M] == 1)
];

CliffordSynthSp[[_ , M_]] := Module[
  {
    CliffordList, SwapTest, MUnSwapped, UpperLeft, LowerRight, RemovedLeftCosets, CZRows, Re
    UpperLeftCol, LowerRightCol, RemovedLeftCosetsNew, S1Test, S2Test, RemovedSSets, d1, d2,
    RemovedPauli1Sets, d3new, d4, d5, d6, RemovedPauli2Sets, ITest, str, phase, list, firstdigi
  },

  CliffordList = ConstantArray[0, 11];

  SwapTest = Total[Abs[M[[1;;3, 1;;3]]], 2];
  CliffordList[[10]] = If[SwapTest > 1, SwapSets[[1]], SwapSets[[2]];
  MUnSwapped = M.Transpose[CliffordList[[10, 1, 2]]];

```

```

UpperLeft = Total[Abs[MUnSwapped[[1;;3,1;;3]],{2}];
LowerRight = Total[Abs[MUnSwapped[[4;;6,4;;6]],{2}];
CliffordList[[1]] = Which[
  UpperLeft == {0,1,1},L1Cosets[[2]],
  UpperLeft == {1,0,1},L1Cosets[[3]],
  True,L1Cosets[[1]]
];
CliffordList[[2]] = Which[
  LowerRight == {0,1,1},L2Cosets[[2]],
  LowerRight == {1,0,1},L2Cosets[[3]],
  True,L2Cosets[[1]]
];
RemovedLeftCosets = Transpose[CliffordList[[1,1,2]].CliffordList[[2,1,2]].MUnSwapped];

CZRows = Total[Abs[RemovedLeftCosets[[3,1;;3]],2];
CliffordList[[3]] = If[CZRows == 1,CZSets[[1]],CZSets[[2]]];
RemovedCZSet = Transpose[CliffordList[[3,1,2]].RemovedLeftCosets];

UpperLeftCol = Abs[RemovedCZSet[[1;;3,3]]];
LowerRightCol = Abs[RemovedCZSet[[4;;6,6]]];
CliffordList[[4]] = Which[
  UpperLeftCol == {0,0,1},L1Cosets[[1]],
  UpperLeftCol == {0,1,0},L1Cosets[[3]],
  True,L1Cosets[[2]]
];
CliffordList[[5]] = Which[
  LowerRightCol == {0,0,1},L2Cosets[[1]],
  LowerRightCol == {0,1,0},L2Cosets[[3]],
  True,L2Cosets[[2]]
];
RemovedLeftCosetsNew = Transpose[CliffordList[[4,1,2]].CliffordList[[5,1,2]].RemovedLeftCosets];

S1Test = Abs[RemovedLeftCosetsNew[[1,1]]];
S2Test = Abs[RemovedLeftCosetsNew[[4,4]]];
CliffordList[[6]] = If[S1Test == 1,S1Sets[[1]],S1Sets[[2]]];
CliffordList[[8]] = If[S2Test == 1,S2Sets[[1]],S2Sets[[2]]];
RemovedSSETS = Transpose[CliffordList[[6,1,2]].CliffordList[[8,1,2]].RemovedLeftCosets];

{d1,d2,d3} = Diagonal[RemovedSSETS[[1;;3,1;;3]]];
CliffordList[[7]] = Which[
  (d1 == d2) && (d2 == d3),Pauli1Sets[[1]],
  (d1 != d2) && (d2 == d3),Pauli1Sets[[2]],
  (d1 != d2) && (d2 != d3),Pauli1Sets[[3]],
  True,Pauli1Sets[[4]]
];
RemovedPauli1Sets = Transpose[CliffordList[[7,1,2]].RemovedSSETS];

{d3new,d4,d5,d6} = Diagonal[RemovedPauli1Sets[[3;;6,3;;6]]];
CliffordList[[9]] = Which[
  (d4 == d5) && (d5 == d6) && (d3new*d4 == 1),Pauli2Sets[[1]],

```

```

      (d4 != d5) && (d5 == d6), Pauli2Sets[[2]],
      (d4 != d5) && (d5 != d6), Pauli2Sets[[3]],
      True, Pauli2Sets[[4]]
    ];
    RemovedPauli2Sets = Transpose[CliffordList[[9,1,2]]].RemovedPauli1Sets;

    ITest = RemovedPauli2Sets[[1,1]];
    CliffordList[[11]] = If[ITest == 1, ISets[[1]], ISets[[2]]];

    {str, phase, list} = Fold[{
      #1[[1]] <> #2[[2]],
      Mod[#1[[2]] + #2[[3]], 8],
      Append[#1[[3]], #2[[4]]]
    } &,
    {"", 0, {}},
    CliffordList
  ];
  firstdigit = list[[3]] * (list[[1]]*3 + list[[2]] + 1);
  cliffordnumber = FromDigits[Prepend[list[[4]; -1], firstdigit], MixedRadix[{10, 3, 3, 2, 4}]];
  {str, phase, IntegerString[cliffordnumber, 16]}
];

```

The Two-Qubit Clifford + CS Group

Here we develop some basic constructors for Clifford + CS circuits. We also provide a function for checking if an operator corresponds to a Clifford + CS circuit.

Check If an Operator Is a Gaussian Clifford + T Matrix

We define a function for determining if an operator is a representation for a Gaussian Clifford + T matrix, i.e. a Clifford + CS operator.

```

DyadicQ[n_] := Module[{numerator, denominator},
  {numerator, denominator} = NumeratorDenominator[n];
  IntegerQ[numerator] && IntegerQ[Log2[denominator]]
];
GaussianDyadicQ[n_] := AllTrue[ReIm[n], DyadicQ];

GaussianQ[U_/?U4Q[U]] := Module[{UGaussDyadicQ, UWGuassDyadicQ},
  UGaussDyadicQ = AllTrue[Flatten[U], GaussianDyadicQ];
  UWGuassDyadicQ = AllTrue[Flatten[W.U], GaussianDyadicQ];
  UGaussDyadicQ || UWGuassDyadicQ
];
GaussianQ[U_/?S06Q[U]] := Module[{sp},
  sp = S06ToSpecialRep[U];
  (Det[U] == 1) && IntegerQ[sp[[1]]] && AllTrue[Flatten[sp[[2]]], IntegerQ]
];
GaussianQ[U_/?ValidHexDecQ[U]] := True;
GaussianQ[U_String] := GaussianQ[FromSequence[U]];
GaussianQ[U_] := (Message[GaussianQ::notacircuit];
  $Failed
);

```

Syllable Lists

For the exported versions:

```

PrePostfixList = {
  {H1.H2,H1S06.H2S06,"H1 H2 ","H2 H1 "},
  {S1.H1.S2.H2,S1S06.H1S06.S2S06.H2S06,"S1 H1 S2 H2 ","H1 S1 S1 S1 H2 S2 S2 S2 "},
  {Id,IdS06,"",""},
  {S1.H1,S1S06.H1S06,"S1 H1 ","H1 S1 S1 S1"},
  {S2.H2,S2S06.H2S06,"S2 H2 ","H2 S2 S2 S2 "},
  {H2,H2S06,"H2 ","H2 "},
  {H1,H1S06,"H1 ","H1 "},
  {H1.S2.H2,H1S06.S2S06.H2S06,"H1 S2 H2 ","H1 H2 S2 S2 S2"},
  {S1.H1.H2,S1S06.H1S06.H2S06,"S1 H1 H2 ","H1 S1 S1 S1 H2"},
  {CNOT12.H1,CNOT12S06.H1S06,"H2 CZ H1 H2 ","H1 H2 CZ H2"},
  {CZ.S1.H1.S2.H2,CZS06.S1S06.H1S06.S2S06.H2S06,"CZ S1 H1 S2 H2 ","H1 S1 S1 S1 H2 S2 S2 S2"},
  {CZ.H1.H2,CZS06.H1S06.H2S06,"CZ H1 H2 ","H1 H2 CZ "},
  {CNOT12.S1.H1,CNOT12S06.S1S06.H1S06,"H2 CZ S1 H1 H2 ","H1 S1 S1 S1 H2 CZ H2"},
  {CZ.S1.H1.H2,CZS06.S1S06.H1S06.H2S06,"CZ S1 H1 H2 ","H1 S1 S1 S1 H2 CZ"},
  {CZ.H1.S2.H2,CZS06.H1S06.S2S06.H2S06,"CZ H1 S2 H2 ","H1 H2 S2 S2 S2 CZ"}
};

SyllableList = Map[
  {#[[1]].CS.ConjugateTranspose[#[[1]],#[[2]].CSS06.Transpose[#[[2]],#[[3]]]<>"CS "
  PrePostfixList
];
SyllableListAsymmetric = Map[
  {#[[1]].CS,#[[2]].CSS06,#[[3]]<>"CS "}&,
  PrePostfixList
];

```

And the internal data structure:

```

SyllableListSp = MapIndexed[
  {S06ToSpecialRep[#[[2]],#[[3]],IntegerString[#[2,16]}]&,
  SyllableList
];
SyllableListAsymmetricSp = MapIndexed[
  {S06ToSpecialRep[#[[2]],#[[3]],IntegerString[#[2,16]}]&,
  SyllableListAsymmetric
];

```

Normalization

In this section we develop a function for normalizing a circuit in any of our representations. This algorithm is described in detail in [2].

Earliest Generator Ordering and associated row pairings

We develop here an association which matches up each syllable to their row pairings under EGO.

```

TwoFourPaired = Map[
  {#, Complement[Range[1,6],#]}&,
  Subsets[Range[1,6],{4}]
];
TwoTwoTwoPaired = Flatten[Map[
  Table[{#[[1]],#[[2,1]],#[[2,k]]},Complement[#[[2]],{#[[2,1]],#[[2,k]]}],{k,2,4}
  Table[{1,j},Complement[Range[1,6],{1,j}]],{j,2,6}
],1];
PairList = Map[Sort,Join[TwoTwoTwoPaired,TwoFourPaired]];

MatchingPairings[list_] := {
  list,
  {list[[3]],Union[list[[1]],list[[2]]}},
  {list[[2]],Union[list[[1]],list[[3]]}},
  {list[[1]],Union[list[[2]],list[[3]]}}
};
PossibleSyllableListPairings = Map[
  MatchingPairings[RowPairs[First[#]]]&,
  SyllableListSp
];
SyllableListPairings = Map[
  # -> First[FirstPosition[PossibleSyllableListPairings,#]]&,
  PairList
];
PairingKey = Association @@ SyllableListPairings;

```

Finding a leftmost syllable and normalizing in the Special representation

We develop separate synthesis algorithms here based on whether we want to synthesize a circuit using the symmetric or asymmetric syllables.

```

LeftmostSyllable[U_] := SyllableListSp[[PairingKey[RowPairs[U]]]];
RemoveLeftmost[{{k_,M_},str_,numberstr_} := Module[{leftmost,clifford},
  Which[
    k > 0,
      leftmost = LeftmostSyllable[{k,M}];
      {Dot2Sp[InvSp[leftmost[[1]]],{k,M}],str<>leftmost[[2]],numberstr<>leftmost
    k == 0,
      clifford = CliffordSynthSp[{k,M}];
      {{-1,SparseArray[ConstantArray[0,{6,6}]],str<>clifford[[1]]<>StringRepeat[
  True,
    {k,M},str,numberstr}
  ]
];

LeftmostSyllableAsymmetric[U_] := SyllableListAsymmetricSp[[PairingKey[RowPairs[U]]]];
RemoveLeftmostAsymmetric[{{k_,M_},str_,numberstr_} := Module[{leftmost,clifford},
  Which[
    k > 0,
      leftmost = LeftmostSyllableAsymmetric[{k,M}];
      {Dot2Sp[InvSp[leftmost[[1]]],{k,M}],str<>leftmost[[2]],numberstr<>leftmost
    k == 0,
      clifford = CliffordSynthSp[{k,M}];
      {{-1,SparseArray[ConstantArray[0,{6,6}]],str<>clifford[[1]]<>StringRepeat[
  True,
    {k,M},str,numberstr}
  ]
];

test[NormItSp,"SyllableType"] := MemberQ[{"Normal","Asymmetric"},#]&;

Options[NormItSp] = {"SyllableType" -> "Normal"};
NormItSp[U_,OptionsPattern[]]?optsCheck := Module[{type},
  type = If[OptionValue["SyllableType"] == "Normal",RemoveLeftmost,RemoveLeftmostAsymm
  FixedPoint[type,{U,"",If[OptionValue["SyllableType"] == "Normal","", "-"]}] [[2];3]]
];

```

Normalizing from a string, a hexadecimal, U(4), or SO(6).

We provide the function NormIt for normalizing operators in any of our forms.

```

Options[FixPhase] = {"UpToPhase" -> True};
FixPhase[{str_, numberstr_}, U_, OptionsPattern[]] := Module[{syllablesplit, syllablenum, syl
  If[
    OptionValue["UpToPhase"] == True,
      {str, numberstr},
      syllablesplit = StringSplit[numberstr, "0000"];
      syllablenum = If[Length[syllablesplit] > 1, syllablesplit[[1]], ""];
      syllablepart = FromHexDec[syllablenum<>"000001"];
      clifford = Simplify[ConjugateTranspose[syllablepart] . U];
      {cliffnum, cliffstr} = CliffordSynth[clifford];
      numberstrfixed = syllablenum<>"00000"<>cliffnum;
      nophasestr = StringDelete[str, "W "];
      phasecount = StringCount[cliffstr, "W "];
      strfixed = nophasestr<>StringRepeat["W ", phasecount];
      {strfixed, numberstrfixed}
    ]
  ];

test[NormIt, "SyllableType"] := MemberQ[{"Normal", "Asymmetric"}, #] &;
test[NormIt, "OutputType"] := MemberQ[{"String", "HexDec"}, #] &;
test[NormIt, "UpToPhase"] := BooleanQ;

Options[NormIt] = {"OutputType" -> "String", "SyllableType" -> "Normal", "UpToPhase" -> Tr
NormIt[U_ /; (GaussianQ[U] && U4Q[U]), OptionsPattern[]]?optsCheck := Module[{index},
  index = If[OptionValue["OutputType"] == "String", 1, 2];
  FixPhase[NormItSp[S06ToSpecialRep[U4ToS06[U]], "SyllableType" -> OptionValue["Syllabl
];
NormIt[U_ /; (GaussianQ[U] && S06Q[U]), OptionsPattern[]]?optsCheck := Module[{index},
  index = If[OptionValue["OutputType"] == "String", 1, 2];
  NormItSp[S06ToSpecialRep[U], "SyllableType" -> OptionValue["SyllableType"]][[index]]
];
NormIt[hexdec_String /; ValidHexDecQ[hexdec], opts:OptionsPattern[]] := NormIt[FromHexDec[h
NormIt[str_String, opts:OptionsPattern[]] := NormIt[FromSequence[str], opts];
NormIt[U_] := (
  Message[NormIt::notacircuit, U];
  $Failed
);

```

ϵ -Approximating Pauli Rotations

This section describes algorithms used for finding approximations to Pauli Rotations

Frobenius Distance of Matrices

Computes the Frobenius distance between two matrices A and B (i.e. the Frobenius norm of the difference between A and B)

```
FrobeniusDistance[A_;/;MatrixQ[A],B_;/;MatrixQ[B]]/;(Dimensions[A] == Dimensions[B]) := S
FrobeniusDistance[_,_] := (
  Message[FrobeniusDistance::notequidimensionalmatrices];
  $Failed
);
```

Continued fractions and affine transformation

We describe a scheme for finding an appropriate affine transformation.

```
NextIterate[{aN_,rN_,pN_,qN_,pNm1_,qNm1_}] := Module[{aNp1},
  aNp1 = IntegerPart[1/rN];
  {aNp1,1/rN - aNp1,aNp1*pN + pNm1,aNp1*qN + qNm1,pN,qN}
];
ContinuedFractionToPrecision[α_,tol_] := Module[{a0,iterate},
  a0 = IntegerPart[α];
  iterate = NestWhile[NextIterate,{a0,N[α - a0,2*Log10[1/tol]],a0,1,1,0},#[[4]] <= 1/
  If[iterate[[2]] == 0,
    iterate[[3;4]],
    iterate[[5;6]]
  ]
];
AffineMatrix[φ_,ε_] := Module[{α,q,r,s,t},
  α = Tan[φ/2];
  {r,q} = ContinuedFractionToPrecision[α,Sqrt[ε/(8+ε*α)]]*Sec[φ/2];
  {t,s} = ExtendedGCD[q,-r][[2]];
  {{q,r},{s,t}}
];
```

Finding angle candidates for bounded and unbounded angles

For calculating a candidate solution, we first map every angle into the interval $[0, \pi/2]$. We then use our affine transformation to find solutions to the integer programming problem.

```
RealQ[x_] := Element[x,Reals];
CandidateFinder[φ_?RealQ,ε_?RealQ] := Module[{modφ},
  modφ=Mod[φ,4*Pi];
  Which[
    0<=modφ<=Pi/2,
      CandidateFinderBounded[modφ,ε],
    Pi/2<modφ<=Pi,
      MapAt[Reverse,CandidateFinderBounded[Pi-modφ,ε],2],
    Pi<modφ<=2*Pi,
      MapAt[{-#[[2]],#[[1]]}&,CandidateFinder[modφ-Pi,ε],2],
    True,
      MapAt[{-#[[1]],-#[[2]]}&,CandidateFinder[modφ-2*Pi,ε],2]
```

```

];
CandidateFinder[_,_] := (
  Message[CandidateFinder::notreals];
  $Failed
);

CandidateFinderBounded[φ_,ε_] := Module[{root2k,c,s,α,p,Δ,δ,A,invA,scaledp',scaledΔ',scaledδ'},
  root2k = 1;
  c = Cos[φ/2];
  s = Sin[φ/2];
  α = 1-ε^2/8;
  p = {c*α+s*Sqrt[1-α^2],s*α-c*Sqrt[1-α^2]};
  Δ = 2*Sqrt[1-α^2]*{-s,c};
  δ = ε^2/8*{c,s};
  A = AffineMatrix[φ,ε];
  invA = Inverse[A];
  scaledp' = A.p;
  scaledΔ' = A.Δ;
  scaledδ' = A.δ;
  θΔ' = VectorAngle[scaledΔ',{1,0}];
  Which[
    θΔ' > Pi/2,
    m1 = scaledΔ'[[2]] / scaledΔ'[[1]];
    m2 = scaledδ'[[2]] / scaledδ'[[1]];
    valid = Catch[Do[
      {x1,x2,x3,x4} = {
        Ceiling[scaledp'[[1]]+scaledΔ'[[1]]],
        Floor[scaledp'[[1]]],
        Floor[scaledp'[[1]]+scaledΔ'[[1]]+scaledδ'[[1]]],
        Floor[scaledp'[[1]]+scaledδ'[[1]]]
      };
    Do[
      y0 = If[x <= x2,
        Ceiling[m1*(x - scaledp'[[1])] + scaledp'[[2]],
        Ceiling[m2*(x - scaledp'[[1])] + scaledp'[[2]]]
      ];
      y1 = If[x <= x3,
        Floor[m2*(x - scaledp'[[1]]-scaledΔ'[[1])] + scaledp'[[2]] + scaledδ'[[2]],
        Floor[m1*(x - scaledp'[[1]]-scaledδ'[[1])] + scaledp'[[2]] + scaledδ'[[2]]
      ];
      Do[
        transformed = invA.{x,y};
        If[Norm[transformed] <= Sqrt[2]^k,Throw[{k,transformed}]],
        {y,y0,y1}
      ];
      {x,x1,x4}
    ];
  ];
];

```

```

        {scaledp', scaledΔ', scaledδ'} *= Sqrt[2];,
        {k, 0, Ceiling[4*Log2[1/ε]+6]}
    ];];,
    θΔ' == Pi/2,
    m1 = scaledδ'[[2]] / scaledδ'[[1]];
    valid = Catch[Do[
        {x1, x2} = {
            Ceiling[scaledp'[[1]]],
            Floor[scaledp'[[1]]+scaledδ'[[1]]]
        };
        Do[
            y0 = Ceiling[m1*(x - scaledp'[[1]]) + scaledp'[[2]]];
            y1 = Floor[m1*(x - scaledp'[[1]]-scaledΔ'[[1]]) + scaledp'[[2]] + s
            Do[
                transformed = invA.{x,y};
                If[Norm[transformed] <= Sqrt[2]^k, Throw[{k, transformed}]],
                {y, Floor[(y0+y1)/2], y1}
            ];,
            {x, x1, x2}
        ];
        {scaledp', scaledΔ', scaledδ'} *= Sqrt[2];,
        {k, 0, Ceiling[4*Log2[1/ε]+6]}
    ];];,
    True,
    m1 = scaledδ'[[2]] / scaledδ'[[1]];
    m2 = scaledΔ'[[2]] / scaledΔ'[[1]];
    valid = Catch[Do[
        {x1, x2, x3, x4} = {
            Ceiling[scaledp'[[1]]],
            Floor[scaledp'[[1]] + scaledδ'[[1]]],
            Floor[scaledp'[[1]] + scaledΔ'[[1]]],
            Floor[scaledp'[[1]] + scaledδ'[[1]] + scaledΔ'[[1]]]
        };
        Do[
            y0 = If[x <= x2,
                Ceiling[m1*(x - scaledp'[[1]]) + scaledp'[[2]]],
                Ceiling[m2*(x - scaledp'[[1]] - scaledδ'[[1]]) + scaledp'[[2]]
            ];
            y1 = If[x <= x3,
                Floor[m2*(x - scaledp'[[1]]) + scaledp'[[2]]],
                Floor[m1*(x - scaledp'[[1]] - scaledΔ'[[1]]) + scaledp'[[2]] +
            ];
            Do[
                transformed = invA.{x,y};
                If[Norm[transformed] <= Sqrt[2]^k, Throw[{k, transformed}]],
                {y, y0, y1}
            ];,
            {x, x1, x4}
        ];
    ];

```

```

        {scaledp, scaledΔ, scaledδ} *= Sqrt[2];,
        {k, 0, Ceiling[4*Log2[1/ε]+6]}
    ];];
];
valid
];

```

Solving Lagrange Four-Squares

Rather than implement our own solver based off of well known algorithms, we instead use a basic Mathematica function as the inputs for this problem never get too big.

```
Lagrange4[n_] := Module[{x1, x2, x3, x4}, {x1, x2, x3, x4} /. FindInstance[x1^2 + x2^2 + x3^2 + x4^2 == n
```

SU(4) Approximations Using a Candidate Solution

We provide an algorithm for finding a rotation by angle φ up to error ϵ for any of the 15 Pauli matrices.

```

SU4Z1Finder[φ_,ε_] := Module[{k,x,y,a,b,c,d,Invroot2k,α,β,χ},
  {k,{x,y}} = CandidateFinder[φ,ε];
  {a,b,c,d} = Lagrange4[2^k-x^2-y^2];
  Invroot2k = 1/Sqrt[2]^k;
  α = Invroot2k*(x+I*y);
  β = Invroot2k*(a+I*b);
  χ = Invroot2k*(c+I*d);
  {
    {α,0,-Conjugate[β],-Conjugate[χ]},
    {0,α,χ,-β},
    {β,-Conjugate[χ],Conjugate[α],0},
    {χ,Conjugate[β],0,Conjugate[α]}
  }
];

PauliRotation[φ_;/;Not[RealQ[φ]],ε_;/;Not[RealQ[ε]],_] := (
  Message[PauliRotation::notreals];
  $Failed
);

PauliRotation[φ_,ε_, "ZI"] := SU4Z1Finder[φ,ε];
PauliRotation[φ_,ε_, "XI"] := H1 . SU4Z1Finder[φ,ε] . H1;
PauliRotation[φ_,ε_, "YI"] := S1 . H1 . SU4Z1Finder[φ,ε] . H1 . ConjugateTranspose[S1];
PauliRotation[φ_,ε_, "IZ"] := EX . SU4Z1Finder[φ,ε] . EX;
PauliRotation[φ_,ε_, "IX"] := EX . H1 . SU4Z1Finder[φ,ε] . H1 . EX;
PauliRotation[φ_,ε_, "IY"] := EX . S1 . H1 . SU4Z1Finder[φ,ε] . H1 . ConjugateTranspose[S1];
PauliRotation[φ_,ε_, "ZZ"] := CNOT21 . SU4Z1Finder[φ,ε] . CNOT21;
PauliRotation[φ_,ε_, "XZ"] := H1 . CNOT21 . SU4Z1Finder[φ,ε] . CNOT21 . H1;
PauliRotation[φ_,ε_, "YZ"] := S1 . H1 . CNOT21 . SU4Z1Finder[φ,ε] . CNOT21 . H1 . ConjugateTranspose[S1];
PauliRotation[φ_,ε_, "ZX"] := H2 . CNOT21 . SU4Z1Finder[φ,ε] . CNOT21 . H2;
PauliRotation[φ_,ε_, "XX"] := H1 . H2 . CNOT21 . SU4Z1Finder[φ,ε] . CNOT21 . H2 . H1;
PauliRotation[φ_,ε_, "YX"] := S1 . H1 . H2 . CNOT21 . SU4Z1Finder[φ,ε] . CNOT21 . H2 . H1;
PauliRotation[φ_,ε_, "ZY"] := S2 . H2 . CNOT21 . SU4Z1Finder[φ,ε] . CNOT21 . H2 . ConjugateTranspose[S2];
PauliRotation[φ_,ε_, "XY"] := H1 . S2 . H2 . CNOT21 . SU4Z1Finder[φ,ε] . CNOT21 . H2 . Cc;
PauliRotation[φ_,ε_, "YY"] := S1 . H1 . S2 . H2 . CNOT21 . SU4Z1Finder[φ,ε] . CNOT21 . H2;
PauliRotation[_,_,_] := (
  Message[PauliRotation::invldstring];
  $Failed
);

test[PauliRotationSequence,"SyllableType"] := MemberQ[{"Normal","Asymmetric"},#]&;
test[PauliRotationSequence,"OutputType"] := MemberQ[{"String","HexDec"},#]&;

Options[PauliRotationSequence] = {"OutputType" -> "String","SyllableType" -> "Normal"};

PauliRotationSequence[φ_,ε_,pauli_,opts:OptionsPattern[]]?optsCheck := NormIt[PauliRotat

```

General Unitary Decomposition

Here we develop an algorithm for the approximation of any $U(4)$ matrix using the Clifford + CS gate set.

Angles of rotation in the Pauli decomposition

First, we develop a method for finding the Pauli decomposition of an operator in terms of its 15 Pauli-rotation angles.

```

SafeArcTan[a_,b_] := If[(a == 0) && (b == 0),0,ArcTan[a,b]];
AngleFind[{{a_,_},{b_,_}}] := SafeArcTan[a,b];
SmallPauliDecomposition[o_] := Module[{v1,v2,v3,φ1,φ2,φ3,M1,M2,or,orr},
  v1 = o[[1;;2,3]];
  φ1 = -SafeArcTan @@ Reverse[v1];
  M1 = {{Cos[φ1],Sin[φ1],0},{-Sin[φ1],Cos[φ1],0},{0,0,1}};
  or = Simplify[M1 . o];
  v2 = or[[2;;3,3]];
  φ2 = -SafeArcTan @@ Reverse[v2];
  M2 = {{1,0,0},{0,Cos[φ2],Sin[φ2]},{0,-Sin[φ2],Cos[φ2]}};
  orr = Simplify[M2 . or];
  v3 = orr[[1;;2,1]];
  φ3 = SafeArcTan @@ v3;
  {φ1,φ2,φ3}
];

PauliDecomposition[U_?U4Q] := PauliDecomposition @ U4ToS06 @ U
PauliDecomposition[U_?S06Q] := Module[{a,b,c,d,U1a',Da,U2a',U1d',Dd,U2d',U1a,U2a,U1d,U2d,
  a = U[[1;;3,1;3]];
  b = U[[4;;6,1;3]];
  c = U[[1;;3,4;6]];
  d = U[[4;;6,4;6]];
  {U1a',Da,U2a'} = SingularValueDecomposition[a];
  {U1d',Dd,U2d'} = SingularValueDecomposition[d];
  {U1a,U2a,U1d,U2d} = {
    FullSimplify[Det[U1a'] * U1a',
    FullSimplify[Det[U2a'] * U2a',
    FullSimplify[Det[U1d'] * U1d',
    FullSimplify[Det[U2d'] * U2d'
  ];
  Db = Simplify[Transpose[U1d] . b . U2a];
  Dc = Simplify[Transpose[U1a] . c . U2d];
  blocks = Map[{{Da[[#,#]],Db[[#,#]]},{Dc[[#,#]],Dd[[#,#]]}&,Range[1,3]];
  {φXX,φYY,φZZ} = -Map[AngleFind,blocks];
  {φZI1,φXI1,φZI2} = -SmallPauliDecomposition[U1a];
  {φZI3,φXI2,φZI4} = -SmallPauliDecomposition[Transpose[U2a]];
  {φIZ1,φIX1,φIZ2} = -SmallPauliDecomposition[U1d];
  {φIZ3,φIX2,φIZ4} = -SmallPauliDecomposition[Transpose[U2d]];
  {
    {φZI1,"ZI"},{φXI1,"XI"},{φZI2,"ZI"},
    {φIZ1,"IZ"},{φIX1,"IX"},{φIZ2,"IZ"},
    {φXX,"XX"},{φYY,"YY"},{φZZ,"ZZ"},
    {φZI3,"ZI"},{φXI2,"XI"},{φZI4,"ZI"},
    {φIZ3,"IZ"},{φIX2,"IX"},{φIZ4,"IZ"}
  }
];
PauliDecomposition[_] := (
  Message[PauliDecomposition::notanoperator];
  $Failed
);

```

Reconstruction using Pauli-rotation angles

```

ApproximateOp[_ , ε_ /; Not[RealQ[ε]]] := (
  Message[Approximate::notreal];
  $Failed
);
ApproximateOp[U_?U4Q, ε_] := Module[{anglesaxes, pauliapproximations},
  anglesaxes = PauliDecomposition[U];
  pauliapproximations = Map[PauliRotation[#[[1]], ε/15, #[[2]]]&, anglesaxes];
  Simplify[Dot @@ pauliapproximations]
];
ApproximateOp[U_?S06Q, ε_] := Module[{anglesaxes, pauliapproximations},
  anglesaxes = PauliDecomposition[U];
  pauliapproximations = Map[PauliRotation[#[[1]], ε/15, #[[2]]]&, anglesaxes];
  U4ToS06 @ Simplify @ Dot @@ pauliapproximations
];
ApproximateOp[_ , _] := (
  Message[Approximate::notanoperator];
  $Failed
);

test[ApproximateSequence, "IfGaussianDoExact"] := BooleanQ;
test[ApproximateSequence, "SyllableType"] := MemberQ[{"Normal", "Asymmetric"}, #]&;
test[ApproximateSequence, "OutputType"] := MemberQ[{"String", "HexDec"}, #]&;

Options[ApproximateSequence] = {"IfGaussianDoExact" -> True, "OutputType" -> "String", "Sy

ApproximateSequence[_ , ε_ /; Not[RealQ[ε]], OptionsPattern[]] := (
  Message[ApproximateSequence::notreal];
  $Failed
);
ApproximateSequence[U_ , ε_ , opts:OptionsPattern[]] := Module[{normitops},
  normitops = FilterRules[{opts}, Options[NormIt]];
  If[
    OptionValue["IfGaussianDoExact"] && GaussianQ[U],
    NormIt[U, Append[normitops, "UpToPhase" -> False]],
    NormIt[ApproximateOp[U, ε], normitops]
  ]
];

```

End of functions in the private context

```
End[];
```

Exported Functions

```
Protect [Id, X1, X2, Z1, Z2, W, H1, H2, S1, S2, CZ, CNOT12, CNOT21, EX, CS, U4ToS06,  
        IdS06, X1S06, X2S06, Z1S06, Z2S06, IS06, H1S06, H2S06, S1S06, S2S06, CZS06, CNOT12S06, CNOT21S06,  
        FromSequence, FromHexDec, CliffordQ, CliffordSynth, RightCliffordSimilar, LeftCliffordSimi  
        CandidateFinder, PauliRotation, PauliDecomposition, ApproximateOp, ApproximateSequence];
```

```
EndPackage[];
```

Bibliography

- [1] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of statistical physics*, 22(5):563–591, 1980.
- [2] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.
- [3] Andrew Steane. Quantum computing. *Reports on Progress in Physics*, 61(2):117, 1998.
- [4] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.
- [5] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.
- [6] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.
- [7] Seth Lloyd. Universal quantum simulators. *Science*, pages 1073–1078, 1996.
- [8] John Preskill. Quantum computing and the entanglement frontier. Rapporteur talk at the 25th Solvay Conference on Physics, 2012.
- [9] Aram W Harrow and Ashley Montanaro. Quantum computational supremacy. *Nature*, 549(7671):203–209, 2017.
- [10] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

- [11] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.
- [12] David P DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik: Progress of Physics*, 48(9-11):771–783, 2000.
- [13] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 333–342. ACM, 2011.
- [14] Peter W Shor. Scheme for reducing decoherence in quantum computer memory. *Physical review A*, 52(4):R2493, 1995.
- [15] Daniel Eric Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, 1997.
- [16] Juan I Cirac and Peter Zoller. Quantum computations with cold trapped ions. *Physical review letters*, 74(20):4091, 1995.
- [17] Neil A Gershenfeld and Isaac L Chuang. Bulk spin-resonance quantum computation. *science*, 275(5298):350–356, 1997.
- [18] David G Cory, Amr F Fahmy, and Timothy F Havel. Ensemble quantum computing by nmr spectroscopy. *Proceedings of the National Academy of Sciences*, 94(5):1634–1639, 1997.
- [19] Daniel Loss and David P DiVincenzo. Quantum computation with quantum dots. *Physical Review A*, 57(1):120, 1998.
- [20] Emanuel Knill, Raymond Laflamme, and Gerald J Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409(6816):46, 2001.
- [21] Alfred V Aho, Ravi Sethi, and Jeffrey D Ullman. Compilers: Principles, techniques, and tools. *Addison wesley*, 7(8):9, 1986.
- [22] Prakash Murali, Norbert Matthias Linke, Margaret Martonosi, Ali Javadi Abhari, Nhung Hong Nguyen, and Cinthia Huerta Alderete. Full-stack, real-system quantum computer studies: architectural comparisons and design insights. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 527–540. ACM, 2019.
- [23] Aram Harrow. *Quantum Compiling*. PhD thesis, Massachusetts Institute of Technology, 2001.
- [24] Ashley Montanaro. Quantum algorithms: an overview. *npj Quantum Information*, 2:15023, 2016.

- [25] Mark Ettinger, Peter Høyer, and Emanuel Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters*, 91(1):43–48, 2004.
- [26] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proc. of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, 1996*, 1996.
- [27] Lov K Grover. From schrödinger’s equation to the quantum search algorithm. *American Journal of Physics*, 69(7):769–777, 2001.
- [28] Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.
- [29] Dominic W Berry, Andrew M Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809. IEEE, 2015.
- [30] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. Simulating hamiltonian dynamics with a truncated taylor series. *Physical review letters*, 114(9):090502, 2015.
- [31] Andrew M Childs. Universal computation by quantum walk. *Physical review letters*, 102(18):180501, 2009.
- [32] Andrew M Childs, Dmitri Maslov, Yunseong Nam, Neil J Ross, and Yuan Su. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences*, 115(38):9456–9461, 2018.
- [33] Guang Hao Low and Isaac L Chuang. Optimal hamiltonian simulation by quantum signal processing. *Physical review letters*, 118(1):010501, 2017.
- [34] Guang Hao Low and Isaac L Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, 2019.
- [35] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.
- [36] Andris Ambainis. Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. Preprint available from [arXiv:1010.4458](https://arxiv.org/abs/1010.4458), 2010.
- [37] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. Quantum linear system algorithm for dense matrices. *Physical review letters*, 120(5):050502, 2018.
- [38] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

- [39] David Beckman, Amalavoyal N Chari, Srikrishna Devabhaktuni, and John Preskill. Efficient networks for quantum factoring. *Physical Review A*, 54(2):1034, 1996.
- [40] Johannes Buchmann, Jiirgen Loho, and Jorg Zayer. An implementation of the general number field sieve. In *Annual International Cryptology Conference*, pages 159–165. Springer, 1993.
- [41] Aleksei Yur’evich Kitaev. Quantum computations: algorithms and error correction. *Uspekhi Matematicheskikh Nauk*, 52(6):53–112, 1997.
- [42] GM D’Ariano, C Macchiavello, and MF Sacchi. On the general problem of quantum phase estimation. *Physics Letters A*, 248(2-4):103–108, 1998.
- [43] H Dieter Zeh. On the interpretation of measurement in quantum theory. *Foundations of Physics*, 1(1):69–76, 1970.
- [44] Maximilian Schlosshauer. Decoherence, the measurement problem, and interpretations of quantum mechanics. *Reviews of Modern physics*, 76(4):1267, 2005.
- [45] Andrew M Steane. Efficient fault-tolerant quantum computing. *Nature*, 399(6732):124, 1999.
- [46] Peter W Shor. Fault-tolerant quantum computation. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 56–65. IEEE, 1996.
- [47] Emanuel Knill, Raymond Laflamme, and Wojciech H Zurek. Resilient quantum computation. *Science*, 279(5349):342–345, 1998.
- [48] Ben W Reichardt. Quantum universality from magic states distillation applied to css codes. *Quantum Information Processing*, 4(3):251–264, 2005.
- [49] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 71(2):022316, 2005.
- [50] Daniel Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. In *Quantum information science and its contributions to mathematics, Proceedings of Symposia in Applied Mathematics*, volume 68, pages 13–58, 2010.
- [51] Earl T Campbell, Barbara M Terhal, and Christophe Vuillot. Roads towards fault-tolerant universal quantum computation. *Nature*, 549(7671):172–179, 2017.
- [52] Bryan Eastin and Emanuel Knill. Restrictions on transversal encoded quantum gate sets. *Physical review letters*, 102(11):110502, 2009.
- [53] Sergey Bravyi and Jeongwan Haah. Magic-state distillation with low overhead. *Physical Review A*, 86(5):052329, 2012.

- [54] Andrew M Steane. Overhead and noise threshold of fault-tolerant quantum error correction. *Physical Review A*, 68(4):042322, 2003.
- [55] Robert Raussendorf and Jim Harrington. Fault-tolerant quantum computation with high threshold in two dimensions. *Physical review letters*, 98(19):190504, 2007.
- [56] Austin G Fowler, Ashley M Stephens, and Peter Groszkowski. High-threshold universal quantum computation on the surface code. *Physical Review A*, 80(5):052312, 2009.
- [57] Daniel Gottesman. Fault-tolerant quantum computation with local gates. *Journal of Modern Optics*, 47(2-3):333–345, 2000.
- [58] Adam Paetznick and Ben W Reichardt. Universal fault-tolerant quantum computation with only transversal gates and error correction. *Physical review letters*, 111(9):090505, 2013.
- [59] Robert Raussendorf. Key ideas in quantum error correction. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1975):4541–4565, 2012.
- [60] Jonas T Anderson and Tomas Jochym-O’Connor. Classification of transversal gates in qubit stabilizer codes. *Quantum Information & Computation*, 16(9-10):771–802, 2016.
- [61] Daniel Gottesman and Isaac L Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402(6760):390, 1999.
- [62] Bei Zeng, Xie Chen, and Isaac L Chuang. Semi-clifford operations, structure of c k hierarchy, and gate complexity for fault-tolerant quantum computation. *Physical Review A*, 77(4):042313, 2008.
- [63] Ingemar Bengtsson, Kate Blanchfield, Earl Campbell, and Mark Howard. Order 3 symmetry in the clifford hierarchy. *Journal of Physics A: Mathematical and Theoretical*, 47(45):455302, 2014.
- [64] Shawn X Cui, Daniel Gottesman, and Anirudh Krishna. Diagonal gates in the clifford hierarchy. *Physical Review A*, 95(1):012329, 2017.
- [65] Alex Bocharov and Krysta M Svore. Resource-optimal single-qubit quantum circuits. *Physical Review Letters*, 109(19):190501, 2012.
- [66] Sergey Bravyi and David Gosset. Improved classical simulation of quantum circuits dominated by clifford gates. *Physical review letters*, 116(25):250501, 2016.

- [67] Matthew Amy, Dmitri Maslov, and Michele Mosca. Polynomial-time t -depth optimization of clifford+ t circuits via matroid partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(10):1476–1489, 2014.
- [68] David Steven Dummit and Richard M Foote. *Abstract algebra*, volume 3. Wiley Hoboken, 2004.
- [69] Lawrence C Washington. *Introduction to cyclotomic fields*, volume 83. Springer Science & Business Media, 1997.
- [70] Michael A Nielsen and IL Chuang. Quantum computation. *Quantum Information. Cambridge University Press, Cambridge*, 2000.
- [71] Phillip Kaye, Raymond Laflamme, Michele Mosca, et al. *An introduction to quantum computing*. Oxford University Press, 2007.
- [72] Daniel Gottesman. Fault-tolerant quantum computation with higher-dimensional systems. In *NASA International Conference on Quantum Computing and Quantum Communications*, pages 302–313. Springer, 1998.
- [73] E. Knill. Non-binary unitary error bases and quantum codes. Technical report, Los Alamos National Laboratory, 1996. quant-ph/9608048.
- [74] E. Knill. Group representations, error bases and quantum codes. Technical report, Los Alamos National Laboratory, 1996. quant-ph/9608049.
- [75] Bradley Dickinson and Kenneth Steiglitz. Eigenvectors and functions of the discrete fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 30(1):25–31, 1982.
- [76] Maris Ozols. Clifford group. *Essays at University of Waterloo, Spring*, 2008.
- [77] JM Farinholt. An ideal characterization of the clifford operators. *Journal of Physics A: Mathematical and Theoretical*, 47(30):305303, 2014.
- [78] Jeroen Dehaene and Bart De Moor. Clifford group, stabilizer states, and linear and quadratic operations over $\text{gf}(2)$. *Physical Review A*, 68(4):042318, 2003.
- [79] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.
- [80] D GOTTESMAN. The heisenberg representation of quantum computers. In *Proc. XXII International Colloquium on Group Theoretical Methods in Physics, 1998*, pages 32–43, 1998.
- [81] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical review A*, 52(5):3457, 1995.

- [82] Mark Howard and Jiri Vala. Qudit versions of the qubit $\pi/8$ gate. *Phys. Rev. A*, 86:022316, Aug 2012.
- [83] Fern H. E. Watson, Earl T. Campbell, Hussain Anwar, and Dan E. Browne. Qudit color codes and gauge color codes in all spatial dimensions. *Phys. Rev. A*, 92:022312, Aug 2015.
- [84] Vivek V Shende, Aditya K Prasad, Igor L Markov, and John P Hayes. Synthesis of reversible logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(6):710–722, 2003.
- [85] Scott Aaronson, Daniel Grier, and Luke Schaeffer. The classification of reversible bit operations. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference*, volume 67 of *LIPICs*, pages 23:1–23:34, 2017. Also available from [arXiv:1504.05155](https://arxiv.org/abs/1504.05155).
- [86] Christopher M. Dawson and Michael A. Nielsen. The Solovay-Kitaev algorithm. *Quantum Information and Computation*, 6(1):81–95, January 2006.
- [87] Aram W Harrow, Benjamin Recht, and Isaac L Chuang. Efficient discrete approximations of quantum gates. *Journal of Mathematical Physics*, 43(9):4445–4451, 2002.
- [88] MS Marinov. Invariant volumes of compact groups. *Journal of Physics A: Mathematical and General*, 13(11):3357, 1980.
- [89] MS Marinov. Correction to invariant volumes of compact groups'. *Journal of Physics A: Mathematical and General*, 14(2):543, 1981.
- [90] Ken Matsumoto and Kazuyuki Amano. Representation of quantum circuits with clifford and $\pi/8$ gates. Preprint available from [arXiv:0806.3834](https://arxiv.org/abs/0806.3834), 2008.
- [91] Brett Giles and Peter Selinger. Remarks on matsumoto and amano's normal form for single-qubit clifford+ t operators. Preprint available from [arXiv:1312.6584](https://arxiv.org/abs/1312.6584), 2013.
- [92] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single-qubit unitaries generated by clifford and t gates. *Quantum Info. Comput.*, 13(7-8):607–630, July 2013.
- [93] Andreas Blass, Alex Bocharov, and Yuri Gurevich. Optimal ancilla-free pauli+ v circuits for axial rotations. *Journal of Mathematical Physics*, 56(12):122201, 2015.
- [94] Alex Bocharov, Yuri Gurevich, and Krysta M. Svore. Efficient decomposition of single-qubit gates into V basis circuits. *Phys. Rev. A*, 88:012313 (13 pages), 2013.

- [95] Vadym Kliuchnikov, Alex Bocharov, Martin Roetteler, and Jon Yard. A framework for approximating qubit unitaries. Preprint available from [arXiv:1510.03888](https://arxiv.org/abs/1510.03888), 2015.
- [96] Brett Giles and Peter Selinger. Exact synthesis of multiqubit Clifford+ T circuits. *Physical Review A*, 87:032332, 2013.
- [97] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Asymptotically optimal approximation of single qubit unitaries by Clifford and T circuits using a constant number of ancillary qubits. *Phys. Rev. Lett.*, 110:190502 (5 pages), 2013.
- [98] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Practical approximation of single-qubit unitaries by single-qubit quantum clifford and t circuits. *IEEE Transactions on Computers*, 65(1):161–172, 2016.
- [99] Peter Selinger. Efficient Clifford+ T approximation of single-qubit operators. *Quantum Information and Computation*, 2014.
- [100] Neil J. Ross and Peter Selinger. Optimal ancilla-free clifford+ T approximation of z -rotations. *Quantum Information & Computation*, 16(11&12):901–953, 2016.
- [101] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
- [102] PP Vaidyanathan. Unitary and paraunitary systems in finite fields. In *IEEE International Symposium on Circuits and Systems*, pages 1189–1192. IEEE, 1990.
- [103] Todd Tilma and ECG Sudarshan. Generalized euler angle parametrization for $su(n)$. *Journal of Physics A: Mathematical and General*, 35(48):10467, 2002.
- [104] Vadym Kliuchnikov. Synthesis of unitaries with clifford+ t circuits. Preprint available from [arXiv:1306.3200](https://arxiv.org/abs/1306.3200), 2013.
- [105] Earl T Campbell, Hussain Anwar, and Dan E Browne. Magic-state distillation in all prime dimensions using quantum reed-muller codes. *Physical Review X*, 2(4):041021, 2012.
- [106] Matthew Amy, Jianxin Chen, and Neil J. Ross. A finite presentation of CNOT-dihedral operators. In *Proceedings of the 14th International Conference on Quantum Physics and Logic, QPL '17*, pages 84–97, 2017. Also available from [arXiv:1701.00140](https://arxiv.org/abs/1701.00140).
- [107] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, 2013.

- [108] Matthew Amy and Michele Mosca. T-count optimization and reed-muller codes. *IEEE Transactions on Information Theory*, 2019.
- [109] Luke E Heyfron and Earl T Campbell. An efficient quantum compiler that reduces t count. *Quantum Science and Technology*, 4(1):015004, 2018.
- [110] Giulia Meuli, Mathias Soeken, and Giovanni De Micheli. SAT-based {CNOT, T} quantum circuit synthesis. In *Proceedings of the 10th International Conference on Reversible Computation*, RC '17, pages 175–188, 2018.
- [111] Peter Selinger. Generators and relations for n-qubit Clifford operators. *Logical Methods in Computer Science*, 11(10):1–17, Jun 2015.
- [112] Andrew N Claudell, Neil J Ross, and Jacob M Taylor. Canonical forms for single-qutrit clifford+ t operators. *Annals of Physics*, 406:54–70, 2019.
- [113] Vadym Kliuchnikov, Alex Bocharov, and Krysta M Svore. Asymptotically optimal topological quantum compiling. *Physical review letters*, 112(14):140504, 2014.
- [114] Neil J. Ross. Optimal ancilla-free Clifford+V approximation of z-rotations. *Quantum Information and Computation*, 15(11–12):932–950, 2015.
- [115] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. Graduate Studies in Mathematics 47. American Mathematical Society, 2002.
- [116] Alex Bocharov, Xingshan Cui, Vadym Kliuchnikov, and Zhenghan Wang. Efficient topological compilation for a weakly integral anyonic model. *Physical Review A*, 93(1):012313, 2016.
- [117] Victor V Albert, Kyungjoo Noh, Kasper Duivenvoorden, Dylan J Young, RT Brierley, Philip Reinhold, Christophe Vuillot, Linshu Li, Chao Shen, SM Girvin, et al. Performance and structure of single-mode bosonic codes. *Physical Review A*, 97(3):032346, 2018.
- [118] Erik Hostens, Jeroen Dehaene, and Bart De Moor. Stabilizer states and clifford operations for systems of arbitrary dimensions and modular arithmetic. *Physical Review A*, 71(4):042315, 2005.
- [119] Marios H Michael, Matti Silveri, RT Brierley, Victor V Albert, Juha Salmilehto, Liang Jiang, and Steven M Girvin. New class of quantum error-correcting codes for a bosonic mode. *Physical Review X*, 6(3):031006, 2016.
- [120] Murphy Yuezhen Niu, Isaac L Chuang, and Jeffrey H Shapiro. Hardware-efficient bosonic quantum error-correcting codes based on symmetry operators. *Physical Review A*, 97(3):032323, 2018.

- [121] Alex Bocharov. A note on optimality of quantum circuits over metaplectic basis. *Quantum Information and Computation*, 18(1&2):0001–0017, 2018.
- [122] Alex Bocharov, Shawn X Cui, Martin Roetteler, and Krysta M Svore. Improved quantum ternary arithmetic. *Quantum Information & Computation*, 16(9-10):862–884, 2016.
- [123] Alex Bocharov, Martin Roetteler, and Krysta M Svore. Factoring with qutrits: Shor’s algorithm on ternary and metaplectic quantum architectures. *Physical Review A*, 96(1):012306, 2017.
- [124] Simon Forest, David Gosset, Vadym Kliuchnikov, and David McKinnon. Exact synthesis of single-qubit unitaries over clifford-cyclotomic gate sets. *Journal of Mathematical Physics*, 56(8):082201, 2015.
- [125] Shiroman Prakash, Akalank Jain, Bhakti Kapur, and Shubangi Seth. Normal form for single-qutrit clifford+ t operators and synthesis of single-qutrit gates. *Physical Review A*, 98(3):032304, 2018.
- [126] Xiaoning Bian. Qutrit matsumoto-amano normal form. <https://www.mathstat.dal.ca/~xbian/QutritMA/index.php>, 2019. Accessed: 2019-11-3.
- [127] Matt Amy, Andrew Glaudell, and Neil J. Ross. Number-theoretic characterizations of some restricted clifford+t circuits. Upcoming publication, preprint available from [arXiv:1908.06076](https://arxiv.org/abs/1908.06076), 2019.
- [128] Alex Bocharov, Martin Roetteler, and Krysta Marie Svore. Efficient synthesis of probabilistic quantum circuits with fallback. *CoRR*, abs/1409.3552, 2014. Also available from [arXiv:1409.3552](https://arxiv.org/abs/1409.3552).
- [129] Peter Selinger and Xiaoning Bian. Relations for 2-qubit clifford+t operator group. Quantum Programming and Circuits Workshop, 2015.
- [130] David Gosset, Vadym Kliuchnikov, Michele Mosca, and Vincent Russo. An algorithm for the T-count. *Quantum Information & Computation*, 14(15-16):1261–1276, November 2014. Also available from [arXiv:1308.4134](https://arxiv.org/abs/1308.4134).
- [131] Seth Greyllyn. Generators and relations for the group $U_4(\mathbb{Z}[1/\sqrt{2}, i])$. Available from [arXiv:1408.6204](https://arxiv.org/abs/1408.6204), 2014.
- [132] Jonathan Welch, Alex Bocharov, and Krysta Svore. Efficient approximation of diagonal unitaries over the Clifford+T basis. *Quantum Information & Computation*, 16(1-2):87–104, 2016.
- [133] Dorit Aharonov. A simple proof that Toffoli and Hadamard are quantum universal. Preprint available from [arXiv:quant-ph/0301040](https://arxiv.org/abs/quant-ph/0301040), January 2003.

- [134] Yaoyun Shi. Both Toffoli and controlled-NOT need little help to do universal quantum computing. *Quantum Information & Computation*, 3(1):84–92, January 2003. Also available from [arXiv:quant-ph/0205115](https://arxiv.org/abs/quant-ph/0205115).
- [135] Terry Rudolph and Lov Grover. A 2 rebit gate universal for quantum computing. Preprint available from [arXiv:quant-ph/0210187](https://arxiv.org/abs/quant-ph/0210187), nov 2002.
- [136] A. K. Hashagen, S. T. Flammia, D. Gross, and J. J. Wallman. Real randomized benchmarking. *Quantum*, 2:85, August 2018. Also available from [arXiv:1801.06121](https://arxiv.org/abs/1801.06121).
- [137] Miriam Backens and Aleks Kissinger. ZH: A complete graphical calculus for quantum computations involving classical non-linearity. In *Proceedings of the 15th International Conference on Quantum Physics and Logic, QPL '18*, pages 23–42, 2018. Also available from [arXiv:1805.02175](https://arxiv.org/abs/1805.02175).
- [138] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Y-calculus: A language for real matrices derived from the ZX-calculus. In *Proceedings of the 14th International Conference on Quantum Physics and Logic, QPL '17*, pages 23–57, 2017. Also available from [arXiv:1702.00934](https://arxiv.org/abs/1702.00934).
- [139] Renaud Vilmart. A ZX-calculus with triangles for Toffoli-Hadamard, Clifford+T, and beyond. In *Proceedings of the 15th International Conference on Quantum Physics and Logic, QPL '18*, pages 313–344, 2018. Also available from [arXiv:1804.03084](https://arxiv.org/abs/1804.03084).
- [140] Daniel Grier and Luke Schaeffer. The classification of stabilizer operations over qubits. Preprint available from [arXiv:1603.03999](https://arxiv.org/abs/1603.03999), 2016.
- [141] Michael Artin. *Algebra*. Prentice Hall, 1991.
- [142] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52:3457–3467, 1995. Also available from [arXiv:quant-ph/9503016](https://arxiv.org/abs/quant-ph/9503016).
- [143] Xiaoning Bian. Private communication, Jul 2019.
- [144] Andrew Glaudell, Neil J. Ross, and Jacob M. Taylor. Optimal two-qubit circuits for universal fault-tolerant quantum computation. Upcoming publication, 2019.
- [145] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A complete axiomatisation of the zx-calculus for clifford+ t quantum mechanics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 559–568. ACM, 2018.

- [146] Bob Coecke and Quanlong Wang. Zx-rules for 2-qubit clifford+ t quantum circuits. In *International Conference on Reversible Computation*, pages 144–161. Springer, 2018.
- [147] Andrew Glaudell. Inexact synthesis of pauli-rotations with the fault-tolerant clifford + controlled-phase gate set. Upcoming publication, in preparation, 2019.
- [148] Michael O Rabin and Jeffery O Shallit. Randomized algorithms in number theory. *Communications on Pure and Applied Mathematics*, 39(S1):S239–S256, 1986.
- [149] Wolfgang M Schmidt. *Diophantine approximations and Diophantine equations*. Springer, 2006.
- [150] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, 2009.
- [151] Heng Fan, Vwani Roychowdhury, and Thomas Szkopek. Optimal two-qubit quantum circuits using exchange interactions. *Physical Review A*, 72(5):052323, 2005.