Neural and Graphical Methods for Quantum Compilation

Richie Yeung

Wolfson College

A thesis submitted for the degree of

Doctor of Philosophy

Trinity term, 2026

Abstract

This thesis studies quantum circuit compilation, a subject concerned with reducing the resources required to implement a quantum computation. We investigate this task through the ZX-calculus and machine learning, focusing on reducing parameter count in parametrised circuits and entangling-gate count in CNOT and Clifford circuits.

We begin by studying parameter optimisation in parametrised quantum circuits (PQCs) with the ZX-calculus. For circuits built from Clifford gates and adjustable single-qubit phase gates, with each parameter used only once, we prove that the phase teleportation algorithm implemented in PyZX minimises parameter count among affine parsimonious reductions. This result extends to measurement-based computations with gflow and distinct measurement parameters. We also show that the assumptions made in our theorems are important by proving that parameter optimisation becomes NP-hard when they are relaxed to allow fixed T gates or repeated parameters in post-selected Clifford circuits.

We then study ZX rewriting as sequence-to-sequence translation. We train encoder–decoder Transformers to learn diagram simplifications from examples generated by PyZX's `full_reduce` routine, without supplying the models with rewrite rules or the diagrams' underlying semantics. The model achieves 96.4% accuracy on held-out ten-qubit Clifford circuits of depth 15, either producing exactly the same diagram as the target diagram or an even smaller alternative. This suggests that Transformers can capture stabiliser structure and learn complex graph transformations from examples alone. However, the accuracy deteriorates as the sequence length increases, exposing limitations in generalisation to larger quantum circuits.

This led us to reinforcement learning (RL) for CNOT and Clifford synthesis within a structured framework that preserves circuit semantics. Formulating the synthesis problems as games on matrix groups, we develop permutation-equivariant, size-agnostic neural network policies. We train these policies without examples of optimal circuits using a novel two-dimensional curriculum that varies qubit count and circuit difficulty. A single model for each synthesis problem accepts circuits of varying size and depth. On benchmarks with known optimal solutions, our circuits have an average entangling-gate count less than one gate above the optimum. The policies also transfer to 30-qubit random-walk targets without retraining and outperform existing baselines.

This work shows that combining machine learning with diagrammatic techniques can push the frontier of quantum computation, uncovering new mathematical results and enabling new tools.

Publications and authorship

This thesis combines and elaborates on the following publications and manuscripts. My contribution to each is described below.

[1] John van de Wetering, **Richie Yeung**, Tuomas Laakkonen, and Aleks Kissinger, 'Optimal Compilation of Parametrised Quantum Circuits', *Quantum*, vol. 9, p. 1828, 2025.

In Chapter 4, John van de Wetering and I jointly developed the reduction model, parametrised Clifford-completeness results, and main optimality results. Tuomas Laakkonen proved the multi-parameter hardness result (Proposition 43); Aleks Kissinger supplied the stabiliser classification (Lemma 38) used in the optimality proof.

[2] **Richie Yeung**, Konstantinos Meichanetzidis, Alexandre Krajenbrink, and François Charton, 'Teaching Small Transformers to Rewrite ZX Diagrams', in *NeurIPS 2023 Workshop on Mathematical Reasoning and AI*, 2023.

The substantive Transformer work in Chapter 5 is my own, carried out with guidance from Konstantinos Meichanetzidis and Alexandre Krajenbrink. François Charton introduced me to sequence-to-sequence methods, provided the initial codebase, and ran substantial experiments using Meta compute resources. I subsequently implemented the conversion between ZX-diagrams and token sequences and carried out the later training, model development, and experiments presented here.

[3] **Richie Yeung**, Aleks Kissinger, and Rob Cornish, 'Equivariant Reinforcement Learning for Clifford Quantum Circuit Synthesis', arXiv:2605.10910, 2026. Under review at NeurIPS.

The Clifford-synthesis study in Chapter 6 is my own research, carried out with guidance from Rob Cornish and Aleks Kissinger.

[4] **Richie Yeung**, Aleks Kissinger, and Rob Cornish, 'Reusable Equivariant Neural Compilers for Matrix-Group Quantum Circuit Synthesis', in *Proceedings of the 2026 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2026. Accepted contribution to the AI for Circuit Synthesis, Optimization, and Discovery Workshop; to appear.

The CNOT-synthesis study in Chapter 6 is also my own research, carried out with guidance from Rob Cornish and Aleks Kissinger.

Contents

Abbreviations and model names

Selected abbreviations and models used in the contribution chapters. Links [§] lead to definitions and explanations.

Abbreviation	**Meaning**
AP	Affine with phases: a representation of stabiliser states using affine binary constraints and a phase polynomial [§]
GSLC	Graph state with local Cliffords: one $\|+\rangle$ qubit per vertex, CZ gates along the edges, then single-qubit Clifford gates [§]
MBQC	Measurement-based quantum computation [§]
MDP	Markov decision process: a mathematical framework for sequential decision-making, defined by states, actions, transitions, and rewards [§]
PCC	Parametrised Clifford circuit: Clifford gates and variable Z rotations, with each parameter used once [§]
PMH	Patel–Markov–Hayes: CNOT synthesis with asymptotically optimal worst-case gate count [§]
PPO	Proximal policy optimisation: a standard policy-gradient RL algorithm [§]
PQC	Parametrised quantum circuit [§]
RL	Reinforcement learning: an agent learns by interacting with an environment to maximise cumulative reward [§]

Model	**Meaning**
BiRel	A neural policy for CNOT synthesis with separate representations of each qubit's control and target roles [§]
RelGNN	Relational graph neural network: a policy for Clifford synthesis using message passing between qubits [§]
RelTransformer	Relation-aware Transformer: a policy for Clifford synthesis using attention informed by relations between qubits [§]

Terminology and notation

Selected concepts and symbols used in the contribution chapters. Links [§] lead to definitions and explanations.

Concept	**Meaning**
Parsimonious	Affine reduction in which each input parameter affects at most one output angle. Parameters may combine, but are not copied to multiple outputs [§]
Permutation-equivariant	Changing the qubit labels changes the policy's gate labels in the same way, preserving their probabilities [§]
Phase teleportation	An algorithm that simplifies phases on a copy of the ZX-diagram, then transfers the resulting angles back to the original circuit without changing its connectivity or non-phase gates [§]
Size-agnostic	An architecture that accepts different qubit counts without retraining [§]

Notation	**Meaning**
$D[\vec{\alpha}]$	Diagram evaluated at phase-parameter values $\vec{\alpha}$ [§]
$M\vec{\alpha} + \vec{c}$	New angles formed from integer-weighted sums of input angles plus fixed offsets, modulo 2π [§]
$\propto$	Equality up to multiplication by a non-zero scalar [§]
$p_\theta(y \mid x)$	Transformer probability of output y given input x [§]
$\mathrm{GL}(n, \mathbb{F}_2)$	Invertible binary $n \times n$ matrices representing CNOT circuits; arithmetic is modulo two [§]
$\mathrm{Sp}(2n, \mathbb{F}_2)$	Binary symplectic matrices representing Clifford operations with Pauli signs omitted [§]
M_U, r_U	Tableau matrix and sign bits recording how Clifford U transforms the Pauli generators [§]
$\mathcal{P}_{n,d}$	RL training distribution: n qubits, mean walk length d [§]
$\pi_{\theta,n}(G \mid M)$	Policy probability of gate G at matrix M on n qubits [§]
$V_\theta(M)$	Value-network estimate of future reward from M, with later rewards given less weight [§]

Chapter I

Introduction

The extended Church–Turing thesis states that every physically reasonable model of computation can be simulated efficiently by a probabilistic Turing machine. Quantum computation challenges this claim: by exploiting the laws of quantum mechanics, quantum algorithms offer efficient solutions to problems for which no efficient classical algorithms are known [5]. A prominent example is Shor's algorithm, which factors integers in polynomial time [6].

The promise of solving hard computational problems motivates the development of quantum computers. Progress towards this goal comes from two directions: researchers are building larger and more reliable quantum computers, while improved algorithms and implementations are reducing the resources required to solve useful problems. For example, the estimated number of physical qubits required to factor a 2048-bit integer has fallen from 20 million [7] to fewer than one million, with an estimated runtime of less than a week [8]. Closing the gap between these resource requirements and available hardware would make such computations possible, opening the way to practical quantum advantage.

Quantum algorithms are commonly expressed in the circuit model, where elementary gates act on qubits and measurements produce classical outcomes. In general, a quantum algorithm can be implemented by many equivalent circuits with substantially different gate counts, depths and resource requirements. Since physical operations are imperfect, each gate or measurement creates another opportunity for error [9]. *Quantum circuit compilation* seeks more efficient implementations, helping close the gap between the resources an algorithm requires and what a quantum computer can provide.

There is, however, a tension at the heart of quantum compilation: it enables us to solve hard computational problems with fewer resources, but doing so requires us to tackle hard optimisation problems in the compilation process itself. Indeed, even basic quantum circuit optimisation problems have been shown to be NP-hard [10]. Researchers therefore translate circuit optimisation into known algebraic problems, borrowing techniques from group theory, tensor decomposition and classical error-correcting codes. Viewing quantum compilation through the lens of these algebraic

structures can yield efficient algorithms for specific circuit families and help us understand the computational difficulty of the underlying optimisation problems.

The ZX-calculus grew out of a similar effort to understand quantum computation through its algebraic structure. It refines the diagrammatic language of categorical quantum mechanics by building diagrams from two complementary observables, Z and X, whose interaction obeys bialgebra and Hopf laws familiar from the study of quantum groups [11]. These algebraic relationships are expressed as graphical rewrite rules, allowing us to reason about quantum computations through the manipulation of diagrams. This exposes structure that can be difficult to see in ordinary circuit notation and provides tools for developing compilation algorithms.

Rewriting a ZX-diagram preserves its meaning. This property is called *soundness*: each rewrite preserves the linear map represented by the diagram. A sequence of rewrites therefore provides a certificate of equality between two diagrams, which can be verified by checking each step. Figure 1 illustrates how such rewrites expose circuit structure: a graph-like ZX-diagram can be viewed as a single instantaneous quantum polynomial (IQP) layer, consisting of commuting Z rotations and controlled-Z gates, followed by postselections. This circuit is an instance of a *parametrised quantum circuit* (PQC).

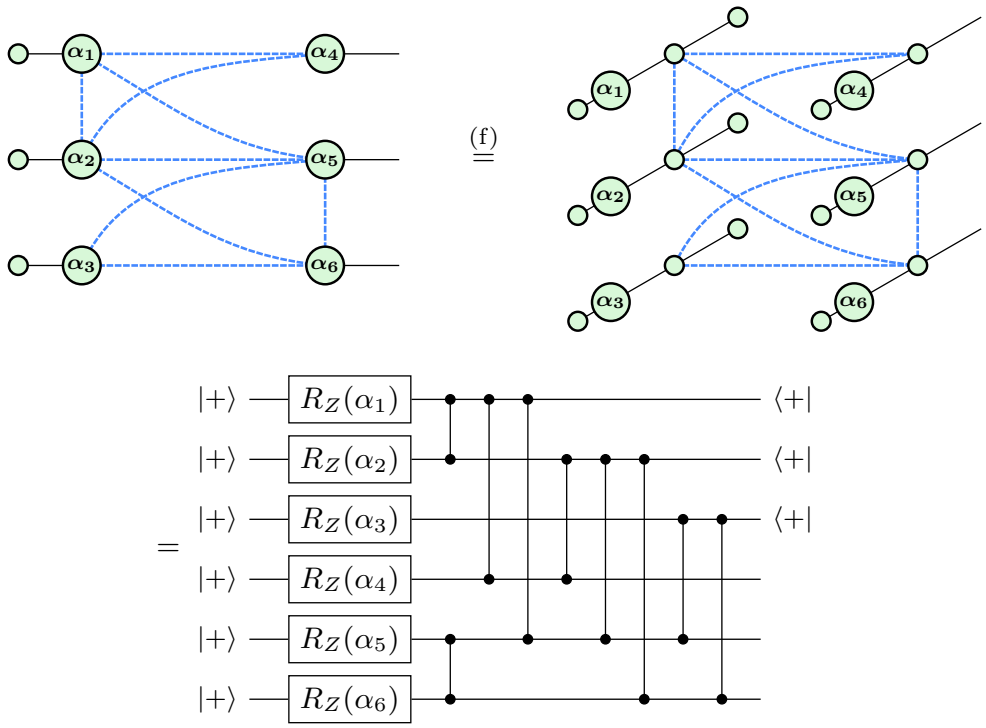


Figure 1: A graph-like ZX-diagram rewritten as a postselected IQP circuit.

The converse of soundness, *completeness*, guarantees that any two diagrams representing the same linear map can be related by a sequence of rewrites. Complete axiomatisations of the ZX-calculus are known, including for diagrams with arbitrary

phases [12]. This guarantees that a derivation exists, but does not tell us how to find one efficiently: the number of possible rewrite sequences grows rapidly with sequence length.

This motivates us to ask whether neural networks can learn the structure of ZX-diagrams and use it to guide simplification. Neural networks have already proved useful in recognising structure in other domains: AlphaFold predicts protein structures from amino-acid sequences [13], while applications in pure mathematics have revealed relationships in knot theory and representation theory that helped guide new conjectures and proofs [14].

Existing quantum compilation methods exploit circuit structure through increasingly sophisticated hand-crafted heuristics, as surveyed in Section 6.3. A natural next step is to ask whether learning and search can discover and improve these strategies automatically. In *The Bitter Lesson*, Sutton [15] argues that general methods such as learning and search are particularly effective at exploiting the growing computational resources made available by Moore's law and the falling cost of computation. As computational resources dedicated to quantum computing continue to grow, our ambition is to use learning and search to develop compilation strategies that outperform hand-crafted heuristics.

Reinforcement learning (RL) allows us to combine learning and search: an agent explores sequences of decisions, learns from the rewards they produce, and uses this experience to guide further exploration. Learning improves the search strategy, while search supplies new experience from which to learn. RL combined with search has achieved superhuman performance in games such as chess, Go, and shogi, as demonstrated by AlphaZero [16]. AlphaTensor-Quantum provides a direct precedent for applying this approach to quantum compilation: it formulates T-count optimisation as a single-player tensor-decomposition game and learns strategies using deep RL [17].

1.1 Thesis overview and contributions

The contributions of this thesis can be organised around PyZX, a software library for compiling quantum circuits and simplifying ZX-diagrams [18] (see Figure 2). Three of its central compilation routines are `full_reduce`, `extract_circuit`, and `teleport_reduce`. The first simplifies a circuit's ZX-diagram, and the second extracts a circuit from the result [19]. The third simplifies phases while preserving the original circuit structure [20].

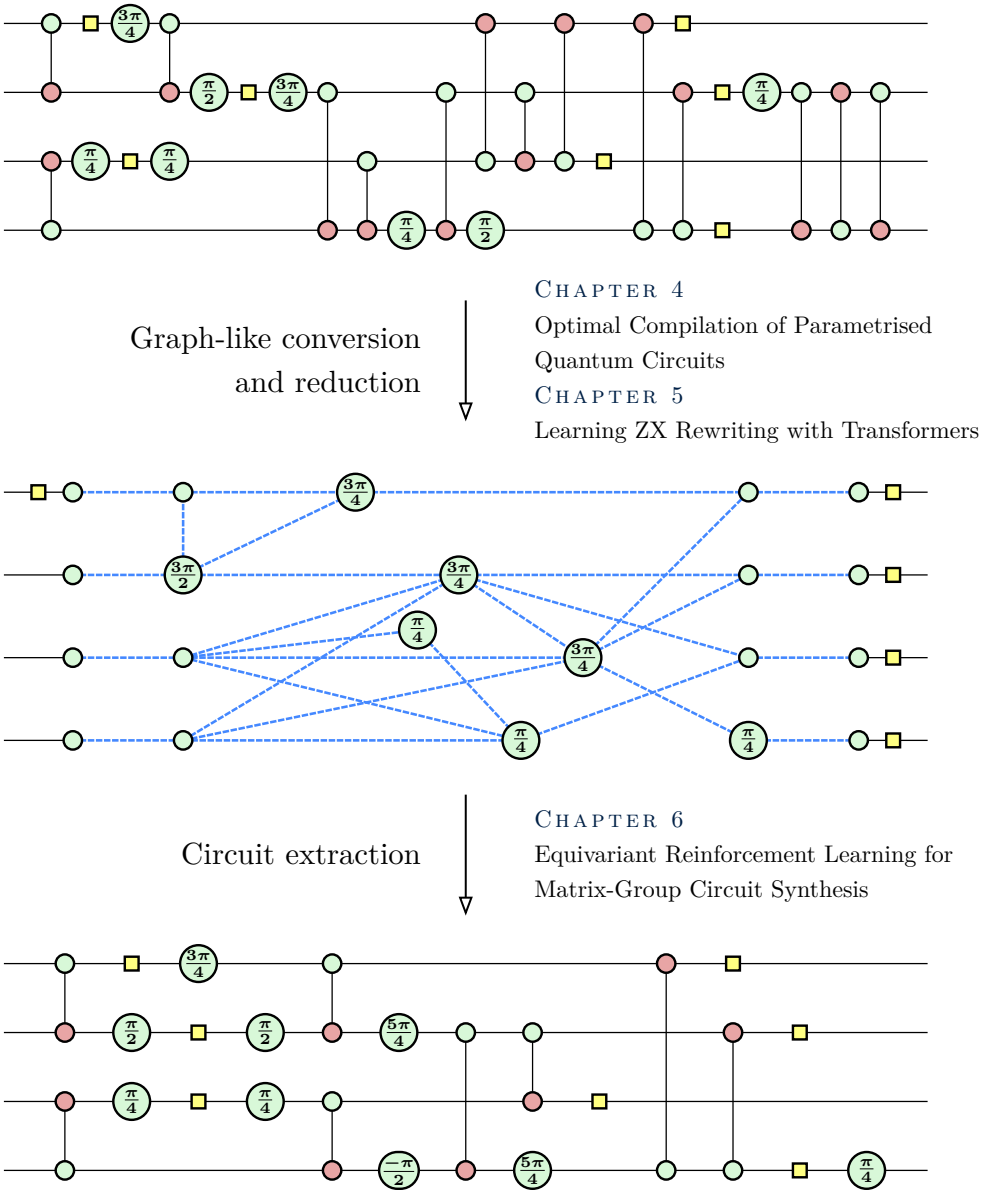


Figure 2: The contributions of this thesis within the PyZX compilation pipeline.

The `full_reduce` procedure simplifies the Clifford structure of a ZX-diagram and fuses or cancels compatible non-Clifford phases [20]. This produces a reduced graphical skeleton, illustrated in the middle panel of Figure 2. The `extract_circuit` routine then constructs an equivalent circuit from this skeleton, with its extraction strategy determining how the remaining structure is realised as gates.

The `teleport_reduce` procedure uses *phase teleportation* to transfer these phase simplifications back to the original circuit. It runs `full_reduce` on a copy of the diagram, tracks how phases combine, and updates the corresponding angles in the original circuit. This preserves the circuit's connectivity and non-phase gates while allowing phase gates to fuse or cancel, avoiding the need to call `extract_circuit` [20].

The two background chapters provide the foundations for the research contributions that follow. Chapter 2 introduces the circuit families and compilation problems

studied in this thesis, while Chapter 3 develops the ZX-calculus and the core rewrite procedures underlying PyZX's simplification algorithms.

In Chapter 4, we prove that, among affine parsimonious reductions, PyZX's `teleport_reduce` procedure minimises the parameter count for parametrised Clifford circuits in which each parameter occurs once. We establish the parametrised Clifford completeness result needed for this proof and extend the optimality result to unitary measurement-based computations with gflow and distinct measurement parameters. We also show why these restrictions matter: parameter optimisation becomes NP-hard when the discrete gate set includes T gates, or when parameters may repeat in post-selected Clifford circuits.

In Chapter 5, we show that Transformers can learn `full_reduce` from input–output examples alone, without rewrite rules or intermediate steps. On held-out ten-qubit Clifford circuits of depth 15, 96.4% of predictions are valid, equivalent to the input, and no larger than the corresponding PyZX `full_reduce` output. The models also produce verified reductions for CNOT and Clifford+T circuits, but their performance declines on circuits deeper than those used for training [2]. This establishes both the feasibility of learning graphical reduction from examples and a limitation of the sequence-to-sequence approach.

In Chapter 6, we study the synthesis problems underlying circuit-extraction routines such as `extract_circuit`. After formulating CNOT and Clifford synthesis as matrix-group reduction games, we introduce permutation-equivariant, size-agnostic policy architectures and a novel two-dimensional curriculum that varies both qubit count and circuit difficulty. Together, these let us train a single policy for each synthesis task and reuse it across qubit counts, without examples of optimal circuits. We achieve near-optimal performance, with entangling-gate counts within one gate of the minimum on average for benchmarks where optimal solutions are known. The CNOT and Clifford policies also generalise to larger circuits and consistently outperform the existing scalable baselines [3, 4].

Chapter II

Algebraic Structure in Quantum Circuit Compilation

Quantum mechanics provides a mathematical framework for describing physical systems and predicting the outcomes of measurements. For example, it captures the behaviour of atoms and photons, including phenomena such as superposition and entanglement that depart from classical intuition. Quantum computing and quantum information theory investigate how these properties can be used to process information.

This chapter introduces quantum compilation, which aims to reduce the cost of executing programs on a quantum computer while preserving their behaviour. We begin with the elements of quantum theory needed to describe quantum circuits, followed by circuit notation and elementary gates. We then introduce circuit rewriting and synthesis, developing the algebraic representations, circuit identities and completeness results for the circuit families studied in this thesis. For CNOT and Clifford synthesis, we present constructive algorithms and counting lower bounds that establish their optimal worst-case scaling, with detailed proofs in Appendix A. Finally, we discuss how hardware connectivity and ancillary qubits affect compilation, and specify the synthesis setting used in Chapter 6.

2.1 Elements of quantum theory

2.1.1 *Quantum states*

A qubit represents a two-level quantum system, such as two energy levels of an atom or two orthogonal polarisation states of a photon. In this thesis, we describe the full system of qubits by a pure state, assuming no interaction with an external environment. A single-qubit state is represented by a unit vector in $\mathbb{C}^2$, called its *state vector*. The computational-basis states are

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1}$$

For two systems A and B, called *subsystems* of the combined system, their joint state space is the *tensor product* $\mathcal{H}_A \otimes \mathcal{H}_B$, in which states $|\psi\rangle_A$ and $|\varphi\rangle_B$ define the *product state* $|\psi\rangle_A \otimes |\varphi\rangle_B$.

Dimensions multiply under tensor products, so an n-qubit system has the 2^n-dimensional state space $\mathcal{H}_n = \left(\mathbb{C}^2\right)^{\otimes n}$, with computational basis

$$|x\rangle = |x_1\rangle \otimes ... \otimes |x_n\rangle, \quad x \in \{0,1\}^n.$$

We abbreviate $|x_1\rangle \otimes |x_2\rangle$ as $|x_1 x_2\rangle$, keeping the order of the subsystems fixed. The tensor product is linear in each argument. For example, the product of the single-qubit states $\alpha|0\rangle + \beta|1\rangle$ and $\gamma|0\rangle + \delta|1\rangle$ expands as

$$(\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle) = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle.$$

General joint states are superpositions of these basis vectors. An n-qubit state has the form

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle, \quad \sum_x |\alpha_x|^2 = 1, \tag{2}$$

where the complex coefficients α_x are *amplitudes*. Pure states that are not product states are *entangled*. The Bell state $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ is an entangled two-qubit state; Figure 3 shows its preparation.

2.1.2 *Unitary evolution*

A closed quantum system evolves reversibly, with its evolution described by a unitary operator U taking $|\psi\rangle$ to $U|\psi\rangle$. An operator U is *unitary* when

$$U^\dagger U = UU^\dagger = I, \tag{3}$$

where the *adjoint* $U^\dagger$ is the conjugate transpose of U and, by this condition, its inverse.

Taking the adjoint of a state vector gives the corresponding bra, $|\psi\rangle^\dagger = \langle\psi|$. Unitarity also preserves inner products:

$$(U|\varphi\rangle)^\dagger (U|\psi\rangle) = \langle\varphi|U^\dagger U\ |\psi\rangle = \langle\varphi|\psi\rangle. \tag{4}$$

Taking $\varphi = \psi$ shows that normalisation is preserved as well.

Operations on separate subsystems combine by tensor product. If U acts on A and V on B, their joint action is

$$(U \otimes V)(|\psi\rangle \otimes |\varphi\rangle) = (U|\psi\rangle) \otimes (V|\varphi\rangle).$$

In a product basis, $U \otimes V$ is represented by the Kronecker product of the two matrices. It is unitary and acts on general joint states by linearity.

2.1.3 *Measurement*

Measurement produces a classical outcome from a quantum system and generally loses information about its original state, so its effect cannot be reversed. In our finite-dimensional qubit setting, these outcomes are discrete: measuring a spin-$\frac{1}{2}$ particle along a chosen axis, for example, gives spin up or spin down.

An *observable* describes the quantity being measured and is represented by a *Hermitian* operator A, satisfying $A = A^\dagger$, with spectral decomposition

$$A = \sum_a a\Pi_a, \tag{5}$$

where the distinct eigenvalues a are the possible outcomes and Π_a projects onto the corresponding eigenspace. The projectors are mutually orthogonal and sum to the identity. For a normalised state $|\psi\rangle$, the *Born rule* gives

$$p(a) = \langle\psi|\Pi_a\ |\psi\rangle. \tag{6}$$

An eigenstate therefore yields its eigenvalue with certainty. Other states generally give probabilistic outcomes. Conditional on obtaining outcome a, an ideal *projective measurement* leaves the state [21]

$$|\psi_a\rangle = \frac{\Pi_a|\psi\rangle}{\sqrt{p(a)}}, \quad p(a) > 0. \tag{7}$$

This projection and normalisation is called *state collapse*. The state now lies in the eigenspace for outcome a, so repeating the same projective measurement with no intervening evolution gives the same outcome with certainty.

For a one-dimensional eigenspace, $\Pi_a = |a\rangle\langle a|$, so the outcome probability is $|\langle a|\psi\rangle|^2$ and the resulting state is $|a\rangle$ up to global phase. In particular, computational-basis measurement of the state in Equation 2 returns x with probability $|\alpha_x|^2$ and leaves the state $|x\rangle$. Multiplying the state by a *global phase* $e^{i\theta}$, with $\theta \in \mathbb{R}$, leaves all measurement probabilities unchanged. For a one-dimensional eigenspace,

$$\left|\langle a|\left(e^{i\theta}|\psi\rangle\right)\right|^2 = \left|e^{i\theta}\langle a|\psi\rangle\right|^2 = |\langle a|\psi\rangle|^2.$$

The three observables used throughout this thesis are the Pauli operators, represented by the matrices

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{8}$$

The Pauli operators have eigenvalues $+1$ and -1. In the spin example, these label spin up and spin down: the physical spin observable is $\left(\frac{\hbar}{2}\right)Z$, with outcomes $+\frac{\hbar}{2}$ and $-\frac{\hbar}{2}$, where $\hbar$ is the reduced Planck constant. The Pauli eigenbases, ordered by outcomes $(+1, -1)$, are

$$\mathcal{B}_Z = \{|0\rangle, |1\rangle\}, \quad \mathcal{B}_X = \{|+\rangle, |-\rangle\}, \quad \mathcal{B}_Y = \{|+i\rangle, |-i\rangle\}, \tag{9}$$

where

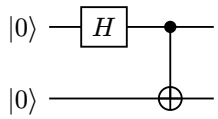
$$|\pm\rangle = \frac{|0\rangle \pm |1\rangle}{\sqrt{2}}, \quad |\pm i\rangle = \frac{|0\rangle \pm i|1\rangle}{\sqrt{2}}.$$

2.2 Quantum circuits

A unitary quantum circuit describes a computation as a sequence of unitary operations, called *gates*, acting on qubits. Each horizontal wire represents a qubit, and the circuit is read from left to right. Labels at the left specify the input state. A box on a wire denotes a single-qubit gate. Gates acting on multiple qubits connect the corresponding wires.

The *gate count* is the number of gates in a circuit. Its *depth* is the fewest time steps needed to run the circuit, counting each gate as one step and allowing gates on disjoint qubits to run in parallel.

For example, Figure 3 prepares the Bell state $|\Phi^+\rangle$ from $|00\rangle$ using a Hadamard gate H followed by a controlled-NOT (CNOT) gate. The state-vector evolution is shown below the circuit.



$$(H \otimes I)|00\rangle = |+\rangle \otimes |0\rangle = \frac{|00\rangle + |10\rangle}{\sqrt{2}},$$

$$\mathrm{CNOT}(H \otimes I)|00\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = |\Phi^+\rangle.$$

Figure 3: Preparation of the Bell state and the corresponding state-vector calculation.

Note that circuits are read from left to right, whereas the rightmost operator in a matrix product acts first: gates $G_1, ..., G_m$ implement $U_C = G_m ... G_1$. A gate on one wire acts as the identity on the others, as $H \otimes I$ illustrates.

Two circuits are *equivalent* when they implement the same unitary, $U_C = U_D$, and *equivalent up to global phase* when $U_C = e^{i\varphi} U_D$ for some $\varphi \in \mathbb{R}$. We specify which convention applies when the distinction matters.

2.2.1 *Elementary gates*

We now introduce the elementary gates used throughout the thesis, together with their matrix representations and circuit symbols.

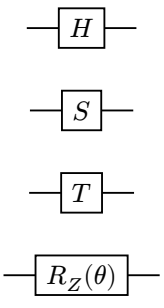
The Pauli operators in Equation 8 are unitary and can therefore act as gates. X flips the computational-basis bit, while Z introduces a relative phase of π between $|0\rangle$ and $|1\rangle$:

$$X|x\rangle = |x \oplus 1\rangle, \quad Z|x\rangle = (-1)^x|x\rangle, \tag{10}$$

where $\oplus$ denotes addition modulo two. Since $Y = iXZ$, a Z gate followed by an X gate implements Y up to global phase.

The identity gate I leaves its input unchanged.
Other standard single-qubit gates include the Hadamard H, the phase gate S, the T gate and the parametrised phase gate $R_Z(\theta)$, with $\theta \in \mathbb{R}$:

$$\begin{aligned} H &= \tfrac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ S &= \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \\ T &= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} \\ R_Z(\theta) &= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix} \end{aligned}$$



In particular, the diagonal gates S, T and Z are special cases of $R_Z(\theta)$:

$$R_Z(\theta)|x\rangle = e^{i\theta x}|x\rangle, \quad S = R_Z\Big(\frac{\pi}{2}\Big), \quad T = R_Z\Big(\frac{\pi}{4}\Big), \quad Z = R_Z(\pi). \tag{11}$$

The Hadamard exchanges the computational and X bases:

$$H|0\rangle = |+\rangle, \quad H|1\rangle = |-\rangle. \tag{12}$$

The S gate maps $|\pm\rangle$ to $|\pm i\rangle$. Its inverse is $S^\dagger = R_Z(-\frac{\pi}{2})$, while $T^\dagger = R_Z(-\frac{\pi}{4})$. The Hadamard and Pauli gates are self-inverse.

2.2.1.1 Controlled gates and swaps

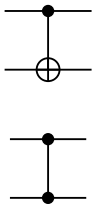
Controlled gates apply an operation to one qubit depending on the state of another. For a single-qubit unitary U, the controlled operation applies U when the control qubit is $|1\rangle$ and the identity when it is $|0\rangle$:

$$\mathrm{C}(U) = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U. \tag{13}$$

In circuit diagrams, a dot on the control wire is connected to the gate on the target wire.

Controlled-NOT (also denoted CX) applies X to its target, while controlled-Z (CZ) applies Z:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \text{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$



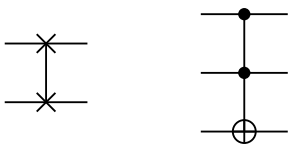
For computational-basis bits $x, y \in \mathbb{F}_2$, with the first qubit controlling the second,

$$\text{CNOT}|x, y\rangle = |x, x \oplus y\rangle, \quad \text{CZ}|x, y\rangle = (-1)^{xy}|x, y\rangle. \tag{14}$$

CZ is symmetric in its two qubits and is drawn with two connected filled dots. The identity $\text{CZ} = (I \otimes H)\,\text{CNOT}(I \otimes H)$ relates the two gates exactly.
The SWAP gate exchanges two qubits, while Toffoli (also denoted CCX) flips its target only when both controls are $|1\rangle$:

$$\text{SWAP}|x, y\rangle = |y, x\rangle, \tag{15}$$

$$\text{Toffoli}|x, y, z\rangle = |x, y, z \oplus (xy)\rangle. \tag{16}$$



SWAP Toffoli

Both are self-inverse. Writing $\text{CNOT}_{a,b}$ for a CNOT with control a and target b, SWAP decomposes as $\text{CNOT}_{1,2}\,\text{CNOT}_{2,1}\,\text{CNOT}_{1,2}$.

Controlled phase gates apply a phase only when every participating bit is one. The singly and doubly controlled versions of $R_Z(\theta)$ are denoted $CR_Z(\theta)$ and $CCR_Z(\theta)$:

$$CR_Z(\theta)|x, y\rangle = e^{i\theta xy}|x, y\rangle, \tag{17}$$

$$CCR_Z(\theta)|x, y, z\rangle = e^{i\theta xyz}|x, y, z\rangle. \tag{18}$$

They are drawn as a phase-gate box with one or two filled controls. In particular, $CR_Z(\pi) = \text{CZ}$; $CCR_Z(\pi)$ is the doubly controlled-Z gate, CCZ.

2.2.1.2 Pauli rotations

Single-qubit rotations extend to rotations about tensor products of Pauli operators. For a signed Hermitian Pauli tensor $Q \in \pm\{I, X, Y, Z\}^{\otimes n}$, define

$$R_Q(\theta) = \frac{I + Q}{2} + e^{i\theta}\frac{I - Q}{2}. \tag{19}$$

For $Q = X, Y, Z$, we obtain the single-qubit gates $R_X(\theta)$, $R_Y(\theta)$, and $R_Z(\theta)$. We use the phase convention in Equation 19 throughout; it differs from the usual rotation

convention only by a global phase. These gates are 2π-periodic and satisfy $R_Q(\theta)^\dagger = R_Q(-\theta)$.

2.3 Circuit compilation, representations, and synthesis

Quantum circuit compilation translates a computation into gates supported by a target device while seeking to reduce its execution cost. This typically involves several stages, from algebraic optimisation to gate decomposition and hardware mapping [22]. This thesis focuses on algebraic optimisation, where representations of the computation expose opportunities to reduce gate or parameter counts.

Circuit compilation at this level commonly uses circuit rewriting and circuit synthesis. Circuit rewriting uses gate identities to transform an existing circuit without changing the computation it performs. Circuit synthesis constructs a circuit from a specification, such as a unitary matrix or Clifford tableau. These approaches can be combined through *resynthesis*: a region of a circuit is replaced by a new implementation of the same operation.

The following sections introduce the algebraic representations, circuit identities, completeness results and synthesis methods for the circuit families studied in this thesis.

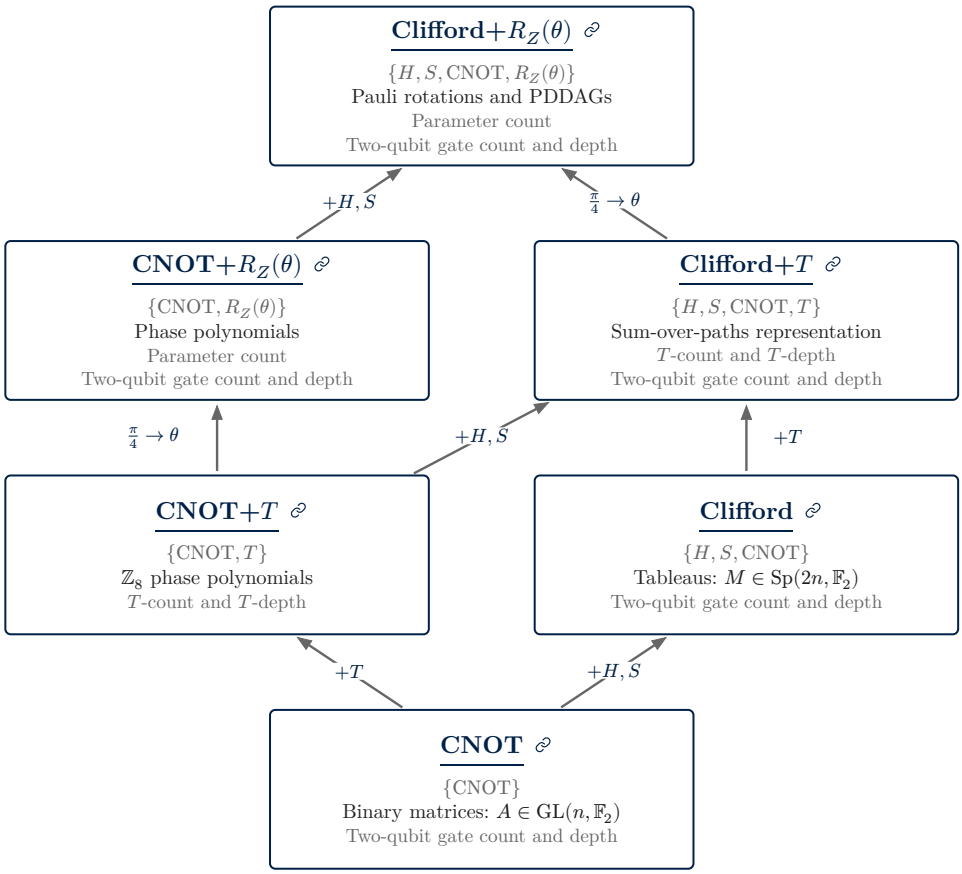


Figure 4: Circuit families of increasing generality, their representations and common compilation costs, as well as links to the corresponding sections.

2.3.1 *CNOT circuits*

A CNOT circuit contains only controlled-NOT gates, each acting on computational-basis states as given in Equation 14.

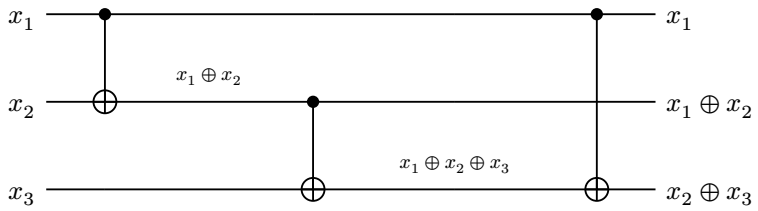
A general n-qubit unitary circuit is represented by a $2^n \times 2^n$ complex unitary matrix, whereas an n-qubit CNOT circuit can be represented by an $n \times n$ invertible binary matrix [23]. Constructing a CNOT circuit from its binary matrix is called *parity synthesis*, typically with the aim of reducing CNOT count or depth.

Let U be the unitary implemented by an n-qubit CNOT circuit. Writing $x \in \mathbb{F}_2^n$ as a column vector, the circuit acts on computational-basis states as

$$U|x\rangle = |Ax\rangle, \quad A \in \mathrm{GL}(n, \mathbb{F}_2). \tag{20}$$

The jth row of A records which input bits are XORed to produce the value on output wire j. This matrix can be read directly from the circuit by propagating symbolic wire values. Initially, wire j carries the symbol x_j. Scanning the circuit diagram from left to right, each $\mathrm{CNOT}_{i,j}$ replaces the symbol on its target wire by its XOR with the symbol on the control wire; all other wire symbols are unchanged. The symbols therefore remain parities of the input bits, and the coefficients of the final parity on output wire j form row j of A. Invertibility follows because every CNOT is self-inverse.

Example 1 Consider the 3-qubit CNOT circuit $\mathrm{CNOT}_{1,3}\,\mathrm{CNOT}_{2,3}\,\mathrm{CNOT}_{1,2}$. To propagate its parities, scan the circuit diagram from left to right and update the target-wire label after each CNOT:



The labels at the right give the overall action on basis states:

$$|x_1, x_2, x_3\rangle \mapsto |x_1, x_1 \oplus x_2, x_2 \oplus x_3\rangle, \quad A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}. \tag{21}$$

The value on the third wire is temporarily $x_1 \oplus x_2 \oplus x_3$; the final CNOT cancels x_1. Thus the circuit and matrix encode the same parity transformation.

The symbolic update for $\mathrm{CNOT}_{i,j}$ corresponds to the row operation

$$r_j \leftarrow r_j \oplus r_i. \tag{22}$$

The synthesis task is therefore to construct a target A from I_n using row operations. Since every update is self-inverse, this is equivalent to reducing A to I_n. Returning to Example 1, the forward construction from I_3 is

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \xrightarrow{r_2 \leftarrow r_2 \oplus r_1} \begin{bmatrix}1&0&0\\1&1&0\\0&0&1\end{bmatrix} \xrightarrow{r_3 \leftarrow r_3 \oplus r_2} \begin{bmatrix}1&0&0\\1&1&0\\1&1&1\end{bmatrix} \xrightarrow{r_3 \leftarrow r_3 \oplus r_1} \begin{bmatrix}1&0&0\\1&1&0\\0&1&1\end{bmatrix}$$

2.3.1.1 Synthesis by Gaussian elimination

The correspondence between CNOT circuits and invertible binary matrices allows us to synthesise circuits using linear algebra over $\mathbb{F}_2$. The underlying decomposition of invertible linear transformations into elementary additions dates back at least as far as Jordan's 1870 work on linear groups [24, §121, pp. 93–94]. Gaussian elimination provides a systematic way to construct such a decomposition.

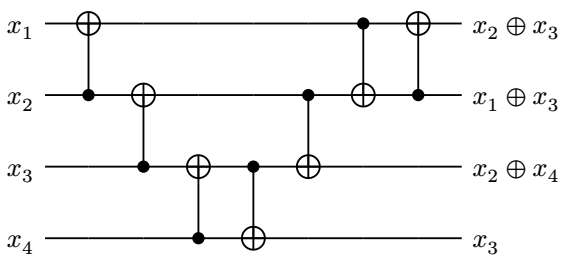
Gaussian elimination reduces the target matrix to the identity in two stages. Forward elimination works column by column, adding a lower row to the pivot row if necessary to make the diagonal entry one, then using the pivot row to clear the entries below it. Since the matrix is invertible, this produces an upper triangular matrix with ones along the diagonal. Backward elimination then clears the entries above the diagonal, working from the last column to the first. Each row addition corresponds to a self-inverse CNOT gate, so reversing the recorded sequence gives a circuit implementing the target.

Gaussian elimination uses $O(n^3)$ classical bit operations and produces a circuit with $O(n^2)$ CNOT gates in the worst case [23].

Example 2 For this four-qubit parity map, forward elimination produces an upper triangular matrix, and backward elimination reduces it to the identity:

$$\begin{bmatrix}0&1&1&0\\1&0&1&0\\0&1&0&1\\0&0&1&0\end{bmatrix} \xrightarrow[\substack{r_2\leftarrow r_2\oplus r_1\\ r_3\leftarrow r_3\oplus r_2\\ r_4\leftarrow r_4\oplus r_3}]{r_1\leftarrow r_1\oplus r_2} \begin{bmatrix}1&1&0&0\\0&1&1&0\\0&0&1&1\\0&0&0&1\end{bmatrix} \xrightarrow[\substack{r_2\leftarrow r_2\oplus r_3\\ r_1\leftarrow r_1\oplus r_2}]{r_3\leftarrow r_3\oplus r_4} \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}.$$

Reversing the seven row additions above gives the following circuit implementing the original parity map, read from left to right:



2.3.1.2 Patel–Markov–Hayes

Patel, Markov and Hayes [23] establish an $\Omega(n^2/\log n)$ lower bound on the worst-case CNOT count for synthesising n-qubit linear reversible transformations without ancillas, assuming all-to-all connectivity and fixed input and output labels. They also give an algorithm that matches this bound asymptotically.

> **Proposition 1** *Let $d_{\mathrm{CNOT}}(n)$ be the maximum, over invertible $n \times n$ binary targets, of the minimum CNOT count for ancilla-free synthesis with all-to-all connectivity and fixed input and output labels. For $n \geq 2$,*
>
> $$d_{\mathrm{CNOT}}(n) \geq \frac{n(n-1)}{\log_2(n(n-1)+1)} = \Omega(n^2/\log n). \tag{23}$$

This lower bound follows from a counting argument. There are $2^{\Theta(n^2)}$ invertible binary targets, whereas each CNOT has only $n(n-1)$ possible placements. Synthesising some targets therefore necessarily requires $\Omega(n^2/\log n)$ CNOT gates. The full argument is given in Section A.2.

Their algorithm divides the columns into blocks and eliminates repeated bit-strings within each block before applying Gaussian elimination.

The algorithm processes blocks of width m from left to right. For each block, consider the rows at or below its first diagonal position. When two rows have the same non-zero pattern within the block, add the earlier row to the later one to cancel that pattern. Each cancellation uses one CNOT. At most $2^m - 1$ distinct non-zero patterns remain for forward Gaussian elimination to clear. Completed columns remain unchanged, so the first pass produces an upper triangular matrix U. Repeating the procedure on U^T yields the identity. To synthesise the target, the second-pass gates are applied in recorded order with controls and targets swapped, followed by the first-pass gates in reverse order. The authors choose $m = \max(1, \lfloor \alpha \log_2 n \rfloor)$ for fixed $0 < \alpha < 1$, giving $O(n^2/\log n)$ CNOTs in the worst case [23].

Example 3 We divide this six-qubit target into three blocks of two columns. Shading marks the active block, with repeated patterns shaded darker. Rows 1, 3 and 5 share $(1,1)$, so we add row 1 to rows 3 and 5, clearing two entries in each. We cancel the other duplicate in the same way, then eliminate below the pivots:

$$\begin{bmatrix} 1&1&0&1&0&0\\ 0&1&1&0&0&0\\ 1&1&1&0&0&0\\ 1&0&0&1&1&1\\ 1&1&0&1&1&0\\ 0&1&0&1&1&1 \end{bmatrix} \xrightarrow[\substack{r_5\leftarrow r_5\oplus r_1\\ r_6\leftarrow r_6\oplus r_2}]{\substack{r_3\leftarrow r_3\oplus r_1\\ \text{cancel}}} \begin{bmatrix} 1&1&0&1&0&0\\ 0&1&1&0&0&0\\ 0&0&1&1&0&0\\ 1&0&0&1&1&1\\ 0&0&0&0&1&0\\ 0&0&1&1&1&1 \end{bmatrix} \xrightarrow[\substack{r_4\leftarrow r_4\oplus r_2\\ \text{eliminate}}]{r_4\leftarrow r_4\oplus r_1} \begin{bmatrix} 1&1&0&1&0&0\\ 0&1&1&0&0&0\\ 0&0&1&1&0&0\\ 0&0&1&0&1&1\\ 0&0&0&0&1&0\\ 0&0&1&1&1&1 \end{bmatrix}.$$

In the second block, rows 3 and 6 share $(1,1)$. We cancel the duplicate, then eliminate below the pivot. The final block needs only ordinary elimination:

$$\begin{bmatrix} 1&1&0&1&0&0\\ 0&1&1&0&0&0\\ 0&0&1&1&0&0\\ 0&0&1&0&1&1\\ 0&0&0&0&1&0\\ 0&0&1&1&1&1 \end{bmatrix} \xrightarrow[\substack{r_4\leftarrow r_4\oplus r_3\\ \text{block 2}}]{r_6\leftarrow r_6\oplus r_3} \begin{bmatrix} 1&1&0&1&0&0\\ 0&1&1&0&0&0\\ 0&0&1&1&0&0\\ 0&0&0&1&1&1\\ 0&0&0&0&1&0\\ 0&0&0&0&1&1 \end{bmatrix} \xrightarrow[\text{block 3}]{r_6\leftarrow r_6\oplus r_5} \begin{bmatrix} 1&1&0&1&0&0\\ 0&1&1&0&0&0\\ 0&0&1&1&0&0\\ 0&0&0&1&1&1\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1 \end{bmatrix}.$$

We then transpose the resulting matrix and repeat the same block-elimination procedure to obtain the identity.

Although PMH is asymptotically optimal, it does not necessarily give the smallest circuit for a given target. In Chapter 6, we show how reinforcement learning can find circuits with fewer CNOTs than PMH.

2.3.1.3 Circuit identities and completeness

CNOT circuits can also be transformed directly using circuit identities. A set of identities is *complete* if every equality between circuits representing the same transformation can be derived from it. Lafont [25] gives such a presentation for invertible binary linear maps, using generators equivalent to CNOT and SWAP gates. The completeness of the rewrite system guarantees that any two equivalent circuits can be related through rewrites, but does not tell us how to choose rewrites that reduce the gate count.

2.3.2 *Clifford circuits*

A *Clifford circuit* is built from H, S, and CNOT gates. Clifford circuits are characterised by their action on Pauli operators, so we first introduce the Pauli group and its binary representation. Recall the Pauli gates introduced in Section 2.2. The operators obtained by freely composing these gates on n qubits form the n-qubit *Pauli group*:

$$\mathcal{P}_n = \{i^k P_1 \otimes ... \otimes P_n \mid k \in \{0,1,2,3\}, P_j \in \{I,X,Y,Z\}\}. \tag{24}$$

The phases arise from Pauli multiplication: $Y = iXZ$, so $XYZ = iI$. Disregarding overall phases gives the *quotient Pauli group*

$$\overline{\mathcal{P}}_n = \mathcal{P}_n / \{\pm I, \pm iI\}.$$

We call a tensor product $P_1 \otimes ... \otimes P_n$, with each $P_j \in \{I,X,Y,Z\}$, a *Pauli string*. It can be encoded by a binary row $(x \quad z) \in \mathbb{F}_2^{2n}$, with the first n entries encoding X and the last n encoding Z. Each qubit uses two bits:

$$I \leftrightarrow (0 \quad 0), \quad X \leftrightarrow (1 \quad 0), \quad Y \leftrightarrow (1 \quad 1), \quad Z \leftrightarrow (0 \quad 1).$$

Multiplication in the quotient Pauli group corresponds to bitwise XOR of the binary rows. Like single-qubit Paulis, two Pauli strings either commute or anticommute, according to whether their single-qubit operators anticommute at an even or odd number of positions. For example, $X \otimes X$ commutes with $Z \otimes Z$, whereas

$$(X \otimes I)(Z \otimes Z) = -(Z \otimes Z)(X \otimes I),$$

so they anticommute.

For binary labels p and q, this commutation condition can be checked using the *symplectic inner product*. Since we need to pair the X component of p with the Z component of q, and vice versa, we define $\langle p, q \rangle_{\text{sp}} = p\Omega q^T$, where Ω swaps the X and Z components. For two qubits, we obtain

$$\begin{aligned} p\Omega q^T &= (x_1 \;\; x_2 \;\; z_1 \;\; z_2) \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1' \\ x_2' \\ z_1' \\ z_2' \end{pmatrix} = (x_1 \;\; x_2 \;\; z_1 \;\; z_2) \begin{pmatrix} z_1' \\ z_2' \\ x_1' \\ x_2' \end{pmatrix} \\ &= x_1 z_1' \oplus x_2 z_2' \oplus z_1 x_1' \oplus z_2 x_2'. \end{aligned} \tag{25}$$

For n qubits, the same calculation gives $p\Omega q^T = x \cdot z' \oplus z \cdot x'$, where $\Omega = \begin{pmatrix} 0 & I_n \\ I_n & 0 \end{pmatrix}$ and all arithmetic is modulo two. The inner product is zero when the Pauli strings commute and one when they anticommute. Equivalently, for Pauli strings P and Q with binary labels p and q,

$$PQ = (-1)^{p\Omega q^T} QP.$$

A *Clifford operator* is a unitary U that preserves the Pauli group under conjugation. In other words, conjugation maps each Pauli string to a distinct Pauli string, up to a sign. These operators form the *Clifford group*:

$$\mathcal{C}_n = \left\{U \in \mathrm{U}(2^n) \mid U\mathcal{P}_n U^\dagger = \mathcal{P}_n\right\}. \tag{26}$$

Since every Pauli string is a product of $X_1, ..., X_n, Z_1, ..., Z_n$, up to phase, a Clifford circuit's action is determined by the images of these $2n$ generators under conjugation [26]. Recording these images gives a compact representation called a *Clifford tableau*. Constructing a Clifford circuit from its tableau is called *Clifford synthesis*, typically with the aim of reducing the two-qubit-gate count or depth.

2.3.2.1 Clifford tableaus

Let U be the operator implemented by an n-qubit Clifford circuit. Its signed *Clifford tableau* consists of a binary matrix $M_U \in \mathbb{F}_2^{2n\times 2n}$, whose rows encode the conjugated Pauli generators, and a column r_U recording their phases. Since the generators are Hermitian and conjugation preserves Hermiticity, these phases can only be ± 1, rather than $\pm i$. Each phase is therefore a sign, recorded by a single bit r through the factor $(-1)^r$.

Throughout this thesis, we write Clifford tableaus in the following format. The upper and lower halves track the X and Z generators, respectively, while the left and right halves of M_U record their X and Z components, giving four $n \times n$ blocks. The final column records the signs:

$$\begin{array}{c|c|ccc|ccc|c}
 & & x_1 & \cdots & x_n & z_1 & \cdots & z_n & r \\
X_1 & \left[\right. & x_{1,1} & \cdots & x_{1,n} & z_{1,1} & \cdots & z_{1,n} & r_1 \\
\vdots & & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
X_n & & x_{n,1} & \cdots & x_{n,n} & z_{n,1} & \cdots & z_{n,n} & r_n \\
\hline
Z_1 & & x_{n+1,1} & \cdots & x_{n+1,n} & z_{n+1,1} & \cdots & z_{n+1,n} & r_{n+1} \\
\vdots & & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
Z_n & \left.\right] & x_{2n,1} & \cdots & x_{2n,n} & z_{2n,1} & \cdots & z_{2n,n} & r_{2n}
\end{array}. \tag{27}$$

Since conjugation preserves commutation and anticommutation, the generator images must retain the original relations: X_i anticommutes with Z_i, while all other pairs of distinct generators commute. These constraints are expressed by the *symplectic condition*

$$M_U \Omega M_U^T = \Omega. \tag{28}$$

Equivalently, $M_U^T \Omega M_U = \Omega$, so M_U belongs to the symplectic group $\mathrm{Sp}(2n, \mathbb{F}_2)$ [27].

Example 4 Under forward conjugation by the one-qubit Clifford generators,

$$H : (X, Y, Z) \mapsto (Z, -Y, X), \quad S : (X, Y, Z) \mapsto (Y, -X, Z). \tag{29}$$

In the row order (X, Z) and with columns labelled $(x \quad z \quad r)$, their signed Clifford tableaus are

$$H: \begin{array}{c} \\ X \\ Z \end{array} \left[\begin{array}{c|c|c} x & z & r \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 0 \end{array}\right] \qquad S: \begin{array}{c} \\ X \\ Z \end{array} \left[\begin{array}{c|c|c} x & z & r \\ \hline 1 & 1 & 0 \\ \hline 0 & 1 & 0 \end{array}\right]$$

Example 5 For CNOT with control qubit 1 and target qubit 2, and for CZ on the same qubits, forward conjugation gives

$$\mathrm{CNOT}_{1,2} : X_1 \mapsto X_1X_2, \quad X_2 \mapsto X_2, \quad Z_1 \mapsto Z_1, \quad Z_2 \mapsto Z_1Z_2. \tag{30}$$

$$\mathrm{CZ}_{1,2} : X_1 \mapsto X_1Z_2, \quad X_2 \mapsto Z_1X_2, \quad Z_1 \mapsto Z_1, \quad Z_2 \mapsto Z_2. \tag{31}$$

Encoding these images in the form of Equation 27 gives the signed tableaus

CNOT

$$\begin{array}{c} \\ X_1 \\ X_2 \\ Z_1 \\ Z_2 \end{array} \left[\begin{array}{cc|cc|c} x_1 & x_2 & z_1 & z_2 & r \\ \hline 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{array}\right]$$

CZ

$$\begin{array}{c} \\ X_1 \\ X_2 \\ Z_1 \\ Z_2 \end{array} \left[\begin{array}{cc|cc|c} x_1 & x_2 & z_1 & z_2 & r \\ \hline 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Having represented a Clifford circuit by its tableau, we now consider how the tableau changes when a gate is appended. Given a circuit that implements U, appending G gives GU, so each stored Pauli image $P' = UPU^\dagger$ is updated by

$$P' \mapsto GP'G^\dagger.$$

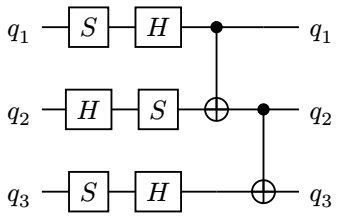
This updates each row of the tableau independently. As shown in Example 4, a Hadamard gate on qubit a swaps x_a and z_a, flipping the sign when both are one. An S gate adds x_a to z_a, with the same sign correction. The CNOT conjugation rules in Example 5 give $x_b \leftarrow x_b \oplus x_a$ and $z_a \leftarrow z_a \oplus z_b$, where a is the control and b the target. Table 1 collects the complete updates, including the CNOT sign correction.

Gate	X bits	Z bits	Sign bit
H_a	$x_{a'} = z_a$	$z_{a'} = x_a$	$r' = r \oplus x_a z_a$
S_a	$x_{a'} = x_a$	$z_{a'} = z_a \oplus x_a$	$r' = r \oplus x_a z_a$
$\mathrm{CNOT}_{a,b}$	$x_{b'} = x_b \oplus x_a$	$z_{a'} = z_a \oplus z_b$	$r' = r \oplus x_a z_b (x_b \oplus z_a \oplus 1)$

Table 1: Clifford tableau update rules [28]. For CNOT, a is the control and b the target. Right-hand sides use the original bit values; unlisted entries are unchanged.

Since H, S, and CNOT generate all Clifford circuits up to global phase, the tableau updates for any other Clifford gate can be derived from these rules. To construct a circuit's tableau, we start with $M = I_{2n}$ and $r = 0$ and apply the updates gate by gate.

Example 6 Consider the three-qubit Clifford circuit



Reading from left to right, the generator images after each layer are:

Layer	Gates	X-Generator Images			Z-Generator Images		
		X_1	X_2	X_3	Z_1	Z_2	Z_3
1	$S_1H_2S_3$	Y_1	Z_2	Y_3	Z_1	X_2	Z_3
2	$H_1S_2H_3$	$-Y_1$	Z_2	$-Y_3$	X_1	Y_2	X_3
3	$\mathrm{CNOT}_{1,2}$	$-Y_1X_2$	Z_1Z_2	$-Y_3$	X_1X_2	Z_1Y_2	X_3
4	$\mathrm{CNOT}_{2,3}$	$-Y_1X_2X_3$	Z_1Z_2	$-Z_2Y_3$	$X_1X_2X_3$	$Z_1Y_2X_3$	X_3

The resulting signed Clifford tableau is

$$\begin{array}{c} \\ X_1 \\ X_2 \\ X_3 \\ Z_1 \\ Z_2 \\ Z_3 \end{array} \begin{array}{c} \begin{array}{cccccc|c} x_1 & x_2 & x_3 & z_1 & z_2 & z_3 & r \end{array} \\ \left[\begin{array}{cccccc|c} 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right] \end{array} \begin{array}{r} \\ -Y_1X_2X_3 \\ Z_1Z_2 \\ -Z_2Y_3 \\ X_1X_2X_3 \\ Z_1Y_2X_3 \\ X_3 \end{array}$$

2.3.2.2 CNOT circuits as a symplectic subclass

The relationship to the preceding CNOT representation is especially simple. For column vectors $a, b \in \mathbb{F}_2^n$, write $X(a) = \prod_i X_i^{a_i}$ and $Z(b) = \prod_i Z_i^{b_i}$. If $U|x\rangle = |Ax\rangle$ as in Equation 20, then

$$UX(a)U^\dagger = X(Aa), \quad UZ(b)U^\dagger = Z(A^{-T}b), \tag{32}$$

where $A^{-T} = \left(A^{-1}\right)^T$. The tableau stores each generator image as a row, so

$$M_U = \begin{bmatrix} A^T & 0 \\ 0 & A^{-1} \end{bmatrix}. \tag{33}$$

Substituting the matrix from Example 1 gives the Clifford tableau of the same three-qubit CNOT circuit.

2.3.2.3 Aaronson–Gottesman

In their work on the efficient classical simulation of stabiliser circuits, Aaronson and Gottesman [28] also give a structured decomposition for Clifford synthesis.

The idea is to separate a Clifford into single-qubit stages and CNOT circuits, then synthesise each CNOT circuit using PMH.

Their decomposition has eleven stages, each containing only Hadamard, S, or CNOT gates. Read from left to right, these are:

$$H + \mathrm{CNOT} + S + \mathrm{CNOT} + S + \mathrm{CNOT} + H + S + \mathrm{CNOT} + S + \mathrm{CNOT}\,.$$

Applying PMH to each of the five CNOT stages gives $O(n^2/\log n)$ CNOTs in total. This scaling is asymptotically optimal under the following assumptions.

> Proposition 2 *Let $d_{\mathrm{Cliff}}(n)$ be the maximum, over n-qubit Clifford targets up to global phase, of the minimum CNOT count for ancilla-free synthesis with all-to-all connectivity, fixed wire labels, and free single-qubit Clifford gates. Then*
>
> $$d_{\mathrm{Cliff}}(n) = \Omega(n^2/\log n). \tag{34}$$

This lower bound follows from a counting argument similar to the CNOT case, but accounting also for free single-qubit Clifford gates. The full proof is given in Section A.3.

2.3.2.4 Circuit identities and completeness

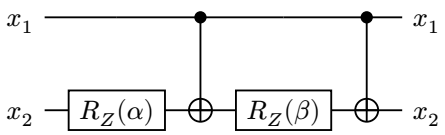
Alternatively, Clifford circuits can be rewritten directly using gate identities. Selinger [29] gives a complete set of circuit identities and a procedure for rewriting every circuit into a unique normal form. His presentation uses Hadamard, S, and CZ gates, together with an explicit global-phase factor. Since Hadamard conjugation converts CZ into CNOT, the presentation can also be expressed using the gate set adopted here. Chapter 3 introduces an alternative approach through the complete graphical rules of the ZX-calculus.

2.3.3 *CNOT+R_Z circuits and phase polynomials*

CNOTs apply a linear transformation to the bits, while R_Z gates change the relative phases between basis states. A circuit built from these gates therefore applies a linear transformation on the basis states together with an input-dependent phase.

One can think of such a circuit as a CNOT circuit with R_Z gates inserted along its wires, each acting on an intermediate parity. Each rotation contributes its angle when that parity is one, making the accumulated phase a weighted sum of parities — the circuit's *phase polynomial* [30]. Together with the final binary transformation, this polynomial describes the circuit's action.

Example 7 Consider the two-qubit CNOT+R_Z circuit



Reading from left to right, the rotations act on the parities x_2 and $x_1 \oplus x_2$, respectively. Disregarding the R_Z gates leaves two CNOTs that cancel, so the input bits are unchanged. Including the accumulated phase, the circuit acts as

$$|x_1, x_2\rangle \mapsto e^{i(\alpha x_2 + \beta(x_1 \oplus x_2))}|x_1, x_2\rangle.$$

Writing f for the phase polynomial and A for the final binary transformation, the circuit acts as

$$U|x\rangle = e^{if(x)}|Ax\rangle, \quad f(x) = \sum_{y \in \mathbb{F}_2^n, y \neq 0} \theta_y \chi_y(x), \tag{35}$$

where $A \in \mathrm{GL}(n, \mathbb{F}_2)$, $\theta_y \in \mathbb{R}$, and $\chi_y(x) = y_1 x_1 \oplus ... \oplus y_n x_n$. Each parity takes the value zero or one, contributing either zero or its rotation angle to f.

To synthesise a phase polynomial, each parity appearing in it must be computed on a wire so that the corresponding phase rotation can be applied. Amy, Azimzadeh and Mosca [31] call a CNOT circuit that computes all these parities a *parity network*. Their GraySynth heuristic shares intermediate parity computations across the terms of the phase polynomial, avoiding the cost of computing and uncomputing each term separately. After inserting the rotations, the authors use PMH to synthesise the remaining parity map needed to obtain the required outputs.

Meijer-van de Griend and Duncan [30] extend this approach to devices with restricted connectivity, using Steiner trees to synthesise the required parity computations along the available connections.

2.3.3.1 Circuit identities and completeness

The phase-polynomial representation gives a circuit normal form for CNOT+R_Z circuits. Circuit equality is determined by the final linear map and the phase function modulo 2π [31]. Choosing each phase value in $[0, 2\pi)$ gives a unique expansion in the non-zero parity functions. We can arrange the resulting non-zero phase gadgets in lexicographic order of their parities, followed by the final linear map, using fixed CNOT constructions for both. Equivalent circuits then have the same circuit normal form, giving a complete method for establishing circuit identities.

2.3.4 *CNOT+T and Clifford+T circuits*

Restricting the phase angles in CNOT+R_Z circuits to multiples of $\frac{\pi}{4}$ gives the CNOT+T family. Writing $\omega = e^{i\frac{\pi}{4}}$, its phase polynomial has coefficients $a_y \in \mathbb{Z}_8$:

$$U|x\rangle = \omega^{p(x)}|Ax\rangle, \quad p(x) = \sum_{y\in\mathbb{F}_2^n, y\neq 0} a_y \chi_y(x) \pmod 8. \tag{36}$$

In phase-polynomial synthesis, each odd coefficient requires one T or $T^\dagger$ gate in a parity-based implementation, while even coefficients require only Clifford phase gates. Different coefficient choices can describe the same unitary, providing opportunities to reduce the T count. Writing $b = \big(a_y \bmod 2\big)_y$, the number of non-zero entries counts the T gates in this implementation. Amy and Mosca [32] relate the search for an equivalent phase polynomial with fewer such entries to *Reed–Muller decoding*.

The same non-Clifford phase information can be represented by a symmetric *signature tensor* over $\mathbb{F}_2$:

$$\mathcal{S} = \sum_{y\neq 0} b_y y^{\otimes 3} = \sum_{r=1}^{t} v_r^{\otimes 3}.$$

Each non-zero binary vector v_r specifies a parity receiving one T gate. A decomposition with fewer terms reduces the T-count, with a diagonal Clifford correction restoring the desired phase operation. Heyfron and Campbell [33] use this representation in their TODD optimiser. AlphaTensor-Quantum uses reinforcement learning to search for these tensor decompositions [17].

These methods optimise T-count in ancilla-free CNOT+T circuits, treating Clifford gates as free. They exploit the fixed rotation angle $\frac{\pi}{4}$, whereas Chapter 4 studies reductions in the number of independent continuous parameters.

Clifford+T circuits are generated by $\{H, S, \mathrm{CNOT}, T\}$ and are approximately universal up to global phase. This gate set is important for implementing more general quantum circuits: arbitrary phase rotations can be approximated to a specified accuracy using Ross and Selinger's [34] algorithm.

The phase-polynomial optimisation methods can also be applied to Hadamard-free regions of a larger Clifford+T circuit. To represent the full circuit, the *sum-over-paths representation* generalises phase polynomials by tracking the phases along the computational paths introduced by Hadamard gates and summing their amplitudes [35].

2.3.4.1 Circuit identities and completeness

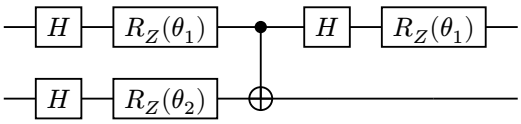
For CNOT+T circuits, Amy, Chen and Ross [36] give a complete set of circuit identities, obtained by restricting their presentation of CNOT-dihedral circuits, which also permits X gates. Their identities rewrite each circuit into a unique normal form, so equivalent circuits can be related by rewriting both into that form.

For the full Clifford+T family, complete circuit-level rulesets are known for one- and two-qubit Clifford+T circuits, but completeness for arbitrary numbers of qubits within the same ancilla-free circuit language remains open [37]. For arbitrary

numbers of qubits, Jeandel, Perdrix and Vilmart [12] establish completeness in the ZX-calculus. Equivalent circuits can therefore be related through graphical rewrites, although the intermediate diagrams need not have circuit form. This greater freedom is one motivation for the graphical approach introduced in Chapter 3.

2.3.5 *Parametrised circuits and Pauli rotations*

A *parametrised quantum circuit* (PQC) is a family $C\left(\vec{\theta}\right)$ specified by a fixed arrangement of gates whose angles depend on parameters $\vec{\theta} \in \mathbb{R}^k$. A parameter may occur on one gate or be shared across several gates. PQCs are used in *variational quantum algorithms*, where a classical optimiser adjusts the parameters using measurement results from the quantum circuit [38]. The following two-qubit PQC has three parametrised gates but only two parameters, since θ_1 is shared by two rotations:



We seek to reduce the two-qubit gate count or parameter count of these circuits while preserving the computation for every parameter assignment.

The phase-polynomial picture extends naturally to these circuits. Where CNOT gates track intermediate parities, Clifford gates track intermediate Pauli strings. Each R_Z gate becomes a rotation about the corresponding Pauli axis, using

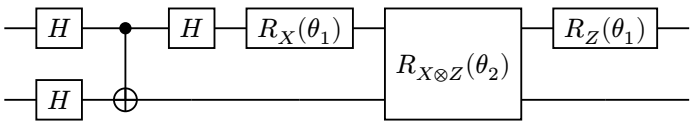
$$CR_Q(\theta) = R_{CQC^\dagger}(\theta)C. \tag{37}$$

Moving the Clifford gates through the rotations gives

$$U\left(\vec{\theta}\right) = R_{Q_m}\left(\varphi_m\left(\vec{\theta}\right)\right)...R_{Q_1}\left(\varphi_1\left(\vec{\theta}\right)\right)C, \tag{38}$$

where each Q_j is a signed Hermitian Pauli string. The Clifford circuit C acts first, followed by the ordered Pauli rotations. The m rotation angles depend on the k parameters and need not be independent.

For the two-qubit PQC introduced above, this gives the following equivalent circuit. Its Clifford remainder is $C = (H \otimes I)\,\mathrm{CNOT}(H \otimes H)$, and its rotation axes, in execution order, are X_1, X_1Z_2, and Z_1:



In a phase polynomial, all rotation axes contain only Z operators and therefore commute. Here, the axes may contain X, Y, and Z, so the order of the rotations matters. Rotations about commuting Pauli strings can be reordered:

$$[Q, R] = 0 \rightarrow R_Q(\alpha)R_R(\beta) = R_R(\beta)R_Q(\alpha). \tag{39}$$

In our two-qubit example, X_1 commutes with X_1Z_2, so the first two rotations can be reordered.

When commuting rotations can be reordered to bring two rotations about the same axis together, they can be fused:

$$R_Q(\alpha)R_Q(\beta) = R_Q(\alpha + \beta). \tag{40}$$

If the independent parameters α and β occur only in these two rotations, fusion replaces them with the single parameter $\alpha + \beta$. Rotations about anticommuting Pauli strings generally cannot be reordered, although particular angle assignments may permit additional simplifications.

A *Pauli Dependency DAG* (PDDAG) records the commutation constraints of this Pauli-rotation representation. It stores the Clifford tableau together with a directed acyclic graph of rotations, whose edges preserve the order of anticommuting axes [39]. Rotations about the same axis can be brought together and fused when no directed path separates them.

Zhang and Chen [40] use this Pauli-rotation representation in TOpt to reduce T-count by merging rotations. For more general rotation angles, Cowtan, Simmons and Duncan [41] group commuting rotations, jointly diagonalise them using Clifford circuits and synthesise the resulting phase polynomials. These approaches expose opportunities to share gates across rotations rather than synthesising each rotation independently.

Clifford operations can also be tracked classically by updating subsequent Pauli measurements, a technique used in Pauli-based computation [42] and lattice surgery [43].

For Trotter approximations to Hamiltonian evolution, choosing the order of Pauli rotations can reduce entangling-gate counts [44, 45]. In the noisy intermediate-scale quantum (NISQ) setting, Huang et al. [46] prioritise this reduction on the assumption that entangling-gate noise outweighs the additional Trotter error introduced by reordering.

2.3.5.1 Circuit identities and completeness

Beyond these commutation and fusion identities, Clément et al. [47] give a complete set of identities for circuits built from Clifford gates and arbitrary phase rotations, subsequently simplified by Clément, Delorme and Perdrix [48]. Equivalent circuits can therefore be related without leaving circuit form. These results concern arbitrary numerical angles; completeness for symbolic parameter families additionally requires a derivation valid for every parameter assignment.

For symbolic parametrised circuits, Jeandel, Perdrix and Vilmart [49] establish completeness in the ZX-calculus when gate angles are integer linear combinations

of parameters with constants in multiples of $\frac{\pi}{4}$. This includes repeated parameters, although intermediate diagrams need not have circuit form. In Chapter 4, we prove completeness for constants in multiples of $\frac{\pi}{2}$, under the restriction that each parameter appears at most once in each diagram.

2.3.6 *Architecture-aware compilation*

In this thesis, we focus on compilation without connectivity constraints, allowing two-qubit gates between any pair of qubits. Superconducting processors typically restrict interactions to a fixed coupling graph, so implementing gates between unconnected qubits can require additional operations, increasing gate count and depth [50, 51]. SABRE routes circuits by inserting SWAP gates [52]. Trapped-ion processors can offer more flexible connectivity by transporting ions between gate zones, although movement and scheduling still have physical costs [53].

Synthesising circuits directly using gates that respect the hardware connectivity can reduce routing overhead by exploiting circuit structure. Steiner-tree methods support CNOT [54, 55] and CNOT+R_Z synthesis [30], while architecture-aware tableau elimination produces Clifford circuits that respect the connectivity [56]. Allowing a final output permutation lets the compiler choose which physical qubit carries each output, providing further opportunities to reduce gate count [57].

2.3.7 *Synthesis with ancillas*

Ancillary qubits provide temporary workspace that can reduce circuit depth or gate count. Selinger [58] gives a Clifford+T implementation of the Toffoli gate using four ancillas and a single layer of T gates. Gidney's [59] temporary logical-AND construction uses an ancilla to store an intermediate result and measurement-based uncomputation to reduce the T-count of quantum addition. Such methods trade additional qubits, and sometimes measurement and classical control, for lower circuit costs. Comparisons must therefore specify the available ancillas and how they are initialised and restored.

The CNOT and Clifford synthesis experiments in Chapter 6 use all-to-all connectivity, fixed input and output labels, and no ancillas.

Chapter III

The ZX-Calculus for Circuit Optimisation

Chapter 2 introduced quantum circuit optimisation through the traditional language of the circuit model and reviewed algebraic representations and equational rewrite methods for several common circuit families. Circuit notation is useful for studying compilation, but other representations can reveal structure that helps us understand and optimise circuits.

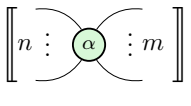
Where circuit diagrams express computations as sequences of gates acting on wires, the ZX-calculus breaks those gates down into smaller graphical building blocks called *spiders* [11]. Since the ZX-calculus is universal, one need not devise a new rewrite system for each gate set, nor complicated extra rules to describe how two gate sets interact.

This chapter introduces the ZX-calculus prerequisites necessary for the thesis. We first introduce the generators of ZX-diagrams and their semantics, and show how to express circuits using only these generators. Then we give the rewrite rules that form the stabiliser and universal fragments of the ZX-calculus, before using them to relate parity projectors to phase gadgets. These projectors and phase gadgets will be used to study the compilation of parametrised quantum circuits in Chapter 4. We next convert the usual red-green representation of ZX-diagrams to the green-Hadamard representation used by the software library PyZX, whose algorithms and underlying graph-like representation are used extensively in the thesis. We introduce stabiliser states through their Pauli generators, then develop their graph state with local Cliffords (GSLC) and affine-with-phases (AP) representations in Section 3.5. AP form occurs as an intermediate step in Clifford simplification, while GSLC form is used in the parameter-optimality proof in Chapter 4. Finally, we introduce Pauli webs and use them to read Clifford tableaus from GSLC diagrams.

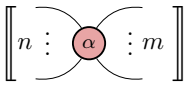
For a comprehensive introduction to the ZX-calculus, see van de Wetering's survey [60] or the book *Picturing Quantum Software* [61].

3.1 ZX-diagrams

ZX-diagrams are *string diagrams* consisting of phase-labelled Z and X spiders connected by open wires. These spiders are essentially non-unitary, multi-wire generalisations of the Z and X rotation gates. A spider may have any number of inputs and of outputs, and its phase is interpreted modulo 2π. We write $Z_{n,m}(\alpha)$ and $X_{n,m}(\alpha)$, where n and m are the numbers of input and output wires, respectively; the phase argument is omitted when it is zero. Their images under the interpretation functor $[\![-]\!]$ are defined graphically below.



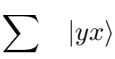
$= |0\rangle^{\otimes m}\langle 0|^{\otimes n} + e^{i\alpha}|1\rangle^{\otimes m}\langle 1|^{\otimes n},$



$= |+\rangle^{\otimes m}\langle +|^{\otimes n} + e^{i\alpha}|-\rangle^{\otimes m}\langle -|^{\otimes n}.$

A phaseless spider is shorthand for a spider with its phase set to zero.

ZX-diagrams are also equipped with swaps, cups, and caps, which allow the wires of the diagram to freely move around. Their graphical definitions are


$= \sum_{x,y\in\{0,1\}} |yx\rangle\langle xy|$

$= \langle 00| + \langle 11|$

$= |00\rangle + |11\rangle$

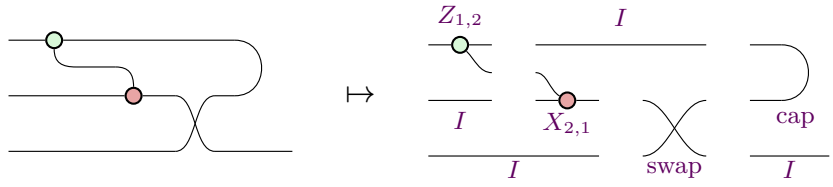
ZX-diagrams compose horizontally and vertically (see Example 8). Let $D_1 : n_1 \to m_1$ and $D_2 : n_2 \to m_2$ be two ZX-diagrams. As with circuit diagrams, inputs are on the left and outputs on the right, while $D_2 \circ D_1$ denotes D_1 followed by D_2. Horizontal composition connects the outputs of D_1 to the inputs of D_2 and corresponds to matrix multiplication. The output and input wire counts must match, so $m_1 = n_2$. Vertical composition stacks the diagrams and corresponds to the tensor product:

$$[\![D_2 \circ D_1]\!] = [\![D_2]\!][\![D_1]\!], \quad [\![D_1 \otimes D_2]\!] = [\![D_1]\!] \otimes [\![D_2]\!].$$

Together, the Z and X generators can represent any linear map between qubits. This property is called *universality*.

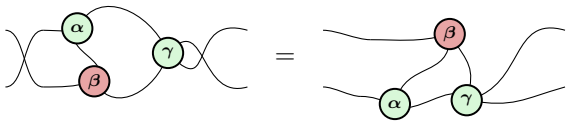
> Theorem 3 ***Universality.*** *For any non-negative integers n and m and any linear map $L : (\mathbb{C}^2)^{\otimes n} \to (\mathbb{C}^2)^{\otimes m}$, there is a ZX-diagram $D : n \to m$ with arbitrary real spider phases such that $[\![D]\!] = L$ exactly, including its scalar factor [61].*

Example 8 The following ZX-diagram decomposes into eight generators:



$$D = (\text{cap} \otimes I) \circ (I \otimes \text{swap}) \circ \big(I \otimes X_{2,1} \otimes I\big) \circ \big(Z_{1,2} \otimes I \otimes I\big).$$

ZX-diagrams are read topologically: *only connectivity matters* (OCM). This allows us to rearrange the positions of the spiders and wires of a ZX-diagram without changing the underlying linear map, provided we don't change the ordering of the wires at the boundary. For example:



(OCM)

Throughout this thesis, $=$ denotes exact equality. For ZX-diagrams, circuits, and linear maps, $A \propto B$ means that their interpretations differ by a non-zero scalar.

To translate a quantum circuit into a ZX-diagram, replace each gate with its graphical representation. Figure 5 collects the representations used throughout this thesis, omitting non-zero scalar factors.

Scalars in ZX are themselves diagrams with no inputs or outputs, and many, including zero, $e^{i\alpha}$, and powers of $\sqrt{2}$, have simple graphical representations:

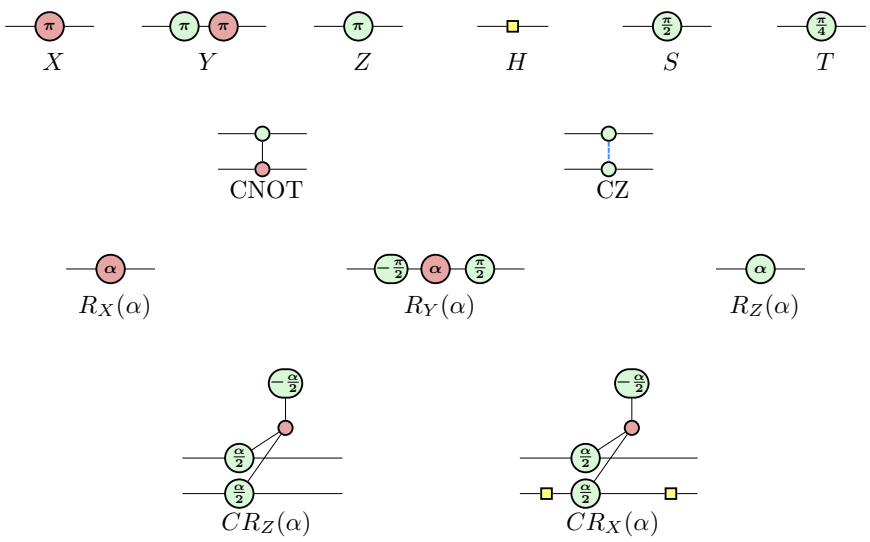
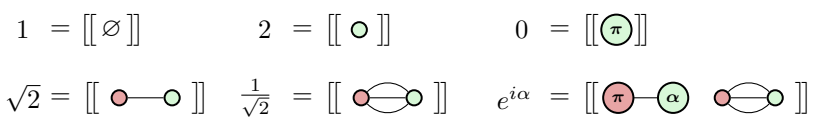


Figure 5: Common quantum gates represented as ZX-diagrams, up to non-zero scalar.



Two additional scalar rules are needed to reason precisely with ZX-diagrams. The inverse rule (IV) cancels the graphical factors $\sqrt{2}$ and $\frac{1}{\sqrt{2}}$, while the zero rule (ZO) allows us to disconnect diagrams whose interpretation is the zero map [62]:

In practice, we often write such a scalar directly next to a diagram for convenience.

Corresponding to the usual circuit fragments, we classify ZX-diagrams according to the phases of their spiders. Throughout, phases are taken modulo 2π. The successive phase angles π, $\frac{\pi}{2}$, and $\frac{\pi}{4}$ mirror the Pauli, Clifford, and T levels of the Clifford hierarchy for single-qubit phase gates [63].

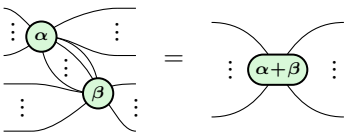
Definition 4 A *phase-free diagram* is a ZX-diagram in which every spider has phase zero. It is the graphical generalisation of a CNOT circuit.

Definition 5 A spider phase is *Pauli* if it is an integer multiple of π, *Clifford* if it is an integer multiple of $\frac{\pi}{2}$, and *proper Clifford* if it is an odd integer multiple of $\frac{\pi}{2}$. Any phase that is not Clifford is *non-Clifford*. A *Clifford diagram* is a ZX-diagram in which every spider has a Clifford phase. It is the graphical generalisation of a Clifford circuit.

Definition 6 A *Clifford+T diagram* is a ZX-diagram in which every spider phase is an integer multiple of $\frac{\pi}{4}$. It is the graphical generalisation of a Clifford+T circuit.

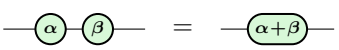
3.2 ZX rewrites

Connected spiders of the same colour can be combined using *spider fusion*. The phases of the two spiders are added up modulo 2π.

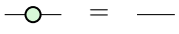


(f)

When both spiders have one input and one output, fusion corresponds to adding the angles of single-qubit phase gates (Equation 11):



Likewise, the following rule captures the idea that a zero phase rotation is equivalent to the identity.

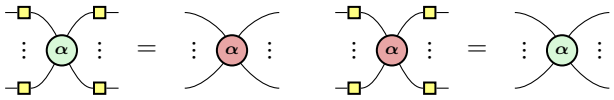


(i1)

The yellow H box abbreviates the Hadamard map shown below. The Hadamard is involutive, so two adjacent H boxes cancel out:

$$[\![\text{—}\square\text{—}]\!] = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \qquad \text{—}\square\square\text{—} = \text{——} \tag{i2}$$

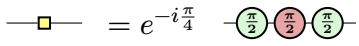
The Hadamard has two main roles. First, applying it to every leg of the spider changes its colour and switches between the Z and X bases:



(h)

Due to this rule, all of the ZX rules still hold with the spider colours interchanged, while leaving the Hadamard unchanged.

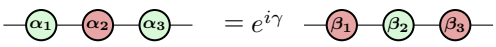
Second, the Hadamard can be expressed using Z and X spiders through its Euler decomposition:



(HD)

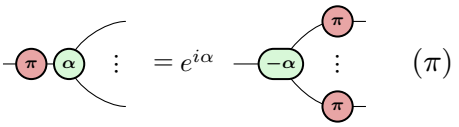
This rule was omitted from the original ZX-calculus ruleset, but turned out to be essential for deriving local-equivalence properties of graph states [64]. We return to graph states in Section 3.5.2.

More generally, the *generalised Euler rule* converts any alternating sequence of three arbitrary-angle rotations into one using the opposite colour order. See Vilmart's work [65] for the exact conversion between the Euler angles.

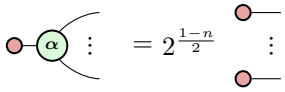


(EU)

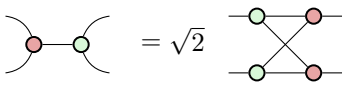
To complete the calculus, we also need the following rules that describe the interaction between Z and X spiders: rule (π) commutes a Pauli-X spider through a Z spider and negates its phase, while (c) copies the unnormalised computational-basis state represented by the degree-one phaseless X spider; in (c), n is the number of output legs:



(π)

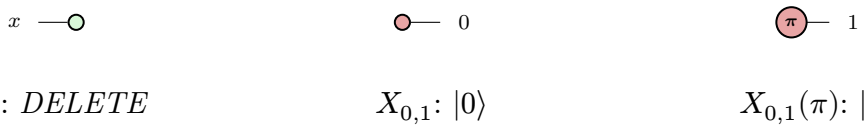


(c)



(b)

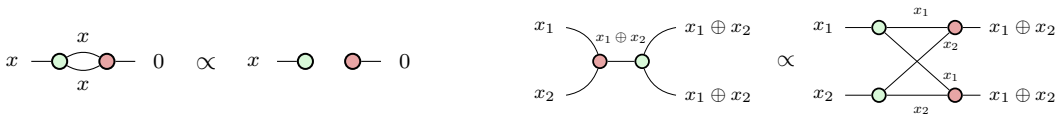
Together with fusion, these three rules enrich the vector space spanned by $|0\rangle$ and $|1\rangle$ with the group structure of $\mathbb{Z}_2$ on its basis elements, thereby yielding a *group algebra*. Concretely, the one-legged X spiders $X_{0,1}$ and $X_{0,1}(\pi)$ are proportional to $|0\rangle$ and $|1\rangle$, respectively, and hence represent its two basis elements. On this basis, $X_{0,1}$ acts as the identity element 0, while $X_{2,1}$ acts as the group multiplication, combining two values by addition modulo 2. In the complementary green structure, $Z_{1,2}$ copies each basis element, whereas $Z_{1,0}$ maps either one to a non-zero scalar and thereby deletes it. Thus, the red unit and multiplication form a monoid, while the green counit and comultiplication form a comonoid.



$Z_{1,0}$: *DELETE* $X_{0,1}$: $|0\rangle$ $X_{0,1}(\pi)$: $|1\rangle$

$Z_{1,2}$: *COPY* $X_{2,1}$: *XOR*

The resulting group algebra satisfies two characteristic interaction laws. The Hopf law expresses the fact that every element of $\mathbb{Z}_2$ is self-inverse, while the bialgebra law describes the interaction between XOR and basis copying: copying an XOR is equivalent to XORing corresponding copies. Both laws are illustrated below, up to a non-zero scalar, with the basis values propagated through each diagram.

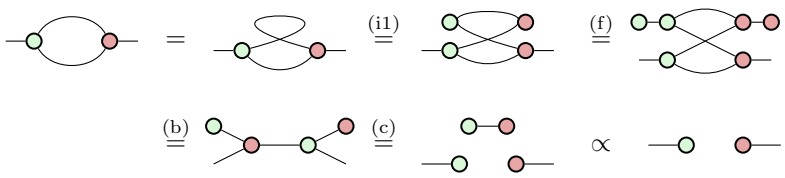


Hopf law *Bialgebra law*

The Hopf law, called the *antipode rule* by Duncan et al. [19], has the following standard ZX form:

(a)

It follows from identity, fusion, bialgebra, and copy, together with topological deformation of the wires:



The final proportionality discards a disconnected non-zero scalar.

Recall that a rewrite system is *complete* for a fragment when every semantically valid equality in that fragment can be derived from its rules. The rewrite rules introduced in this section give rise to two completeness results.

Theorem 7 ***Clifford completeness.*** *Define the Clifford rewrite system by*

$$\mathcal{R}_{\text{Cliff}} = \{(\text{OCM}), (\text{f}), (\text{i1}), (\text{i2}), (\text{h}), (\text{HD}), (\pi), (\text{c}), (\text{b}), (\text{IV}), (\text{ZO})\}.$$

If D_1 and D_2 are Clifford diagrams with $[\![D_1]\!] = [\![D_2]\!]$, then a sequence of rewrites from $\mathcal{R}_{\text{Cliff}}$ transforms D_1 into D_2 [62, 66].

Theorem 8 ***Universal completeness.*** *Define the universal rewrite system by*

$$\mathcal{R}_{\text{univ}} = \mathcal{R}_{\text{Cliff}} \cup \{(\text{EU})\}.$$

The rewrite system $\mathcal{R}_{\text{univ}}$ is complete for ZX-diagrams with arbitrary phases [65].

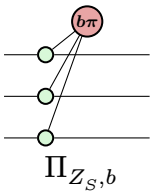
3.3 Projectors and phase gadgets

We now use ZX rewriting to study two further constructions: projectors and phase gadgets.

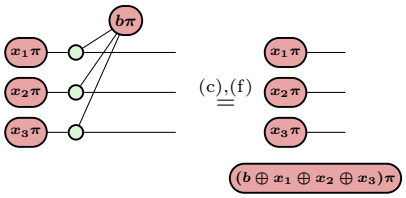
A projective measurement of a Hermitian Pauli operator Q has two possible outcomes, corresponding to the eigenvalues $+1$ and -1. Each outcome selects the associated eigenspace. Indexing the outcomes by $b \in \mathbb{F}_2$, with eigenvalue $(-1)^b$, the corresponding projectors are

$$\Pi_{Q,b} = \frac{I + (-1)^b Q}{2}, \quad b \in \mathbb{F}_2.$$

For $S \subseteq \{1, ..., n\}$, let Z_S denote the Pauli string that applies Z to each qubit in S and I elsewhere. The corresponding parity projector is represented projectively by

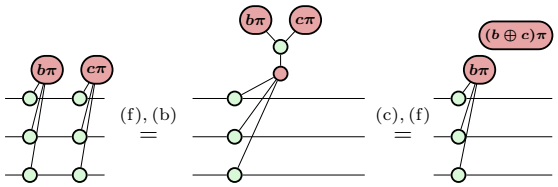


Example 9 (**Projector on basis states.**) For $S = \{1, 2, 3\}$, plugging the basis values $x_1, x_2, x_3 \in \mathbb{F}_2$ into the projector and applying copy (c) and spider fusion (f) gives



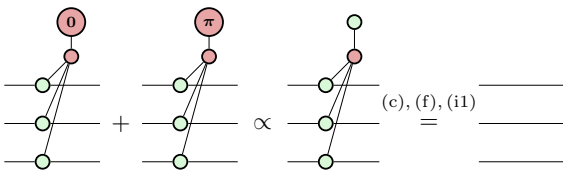
The disconnected red scalar has phase $(b \oplus x_1 \oplus x_2 \oplus x_3)\pi$. When $b \neq x_1 \oplus x_2 \oplus x_3$, this phase is π modulo 2π, so the scalar is zero and annihilates the diagram. Otherwise it is non-zero and the basis state is retained projectively.

Since each projector selects one of two orthogonal eigenspaces, applying the same projector twice has no further effect, so the projector is idempotent. Applying projectors for different outcomes instead gives the zero diagram. The disconnected scalar distinguishes these cases: its phase is $(b \oplus c)\pi$, so it is non-zero when $b = c$ and zero otherwise.



(54)

The two outcome projectors account for the whole state space. Their eigenspaces are complementary, so their sum is the identity. In the first graphical equality, the two red basis states combine into a phaseless green state because $|0\rangle + |1\rangle$ is proportional to $|+\rangle$.

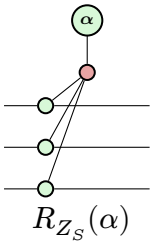


(55)

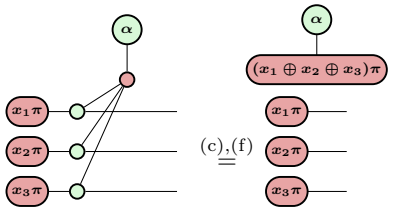
Phase gadgets act on the two eigenspaces of Q by leaving the $+1$ eigenspace unchanged and applying a relative phase $e^{i\alpha}$ to the -1 eigenspace:

$$R_Q(\alpha) = \Pi_{Q,0} + e^{i\alpha}\Pi_{Q,1}. \tag{56}$$

The $+1$ and -1 eigenspaces of Z_S are respectively the even- and odd-parity subspaces on S. Attaching a green phase α to a phaseless red parity spider gives a projective representation of the corresponding phase gadget.

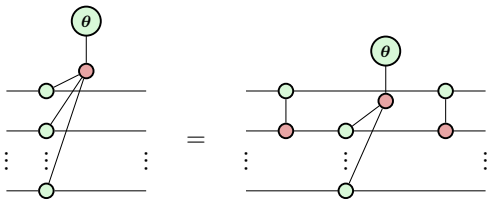


Example 10 (**Phase gadget on basis states.**) For $S = \{1, 2, 3\}$, plugging the basis values $x_1, x_2, x_3 \in \mathbb{F}_2$ into the phase gadget and applying copy (c) and spider fusion (f) gives



After discarding a common non-zero factor, the disconnected scalar evaluates to 1 when the parity $x_1 \oplus x_2 \oplus x_3$ is even and to $e^{i\alpha}$ when it is odd. Thus the phase gadget preserves the basis state while applying a phase precisely to the odd-parity subspace.

A phase gadget can be implemented recursively by removing one target with a CNOT pair:



(PG)

Iterating (PG) produces a CNOT ladder. Since the target wires of a phase gadget may be permuted, the same parity computation can instead be arranged as a balanced tree:

Phase gadgets are useful intermediate representations for circuit compilation. Kissinger and van de Wetering [20] use them to expose non-local cancellations of non-Clifford phases; Cowtan et al. [67] synthesise them into shallow circuits; and Meijer-van de Griend and Duncan [30] incorporate hardware connectivity into phase-polynomial synthesis.

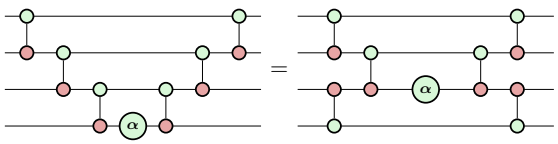


Figure 6: Four-qubit CNOT-ladder (left) and balanced-tree (right) phase-gadget implementations, adapted from Cowtan et al. [67]. Both use $2(n-1)$ CNOTs in general, with depths $2(n-1)$ and $2\lceil \log_2 n \rceil$.

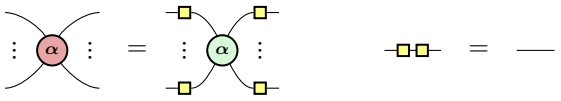
3.4 From circuits to graph-like diagrams

We have seen how to represent quantum circuits as ZX-diagrams and reason about them using a complete set of rewrite rules. While the ZX-calculus is complete, it is often not clear how to efficiently derive identities or simplify ZX-diagrams using these rules. The rules can be applied in either direction, and the proofs of Theorem 7 and Theorem 8 use this freedom. Unrestricted rewriting need not terminate, but fixing the direction of each rule can limit the available simplifications.

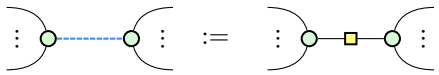
Motivated by the need for a terminating simplification procedure for ZX-diagrams, Duncan et al. introduced *graph-like* ZX-diagrams [19]. As a first stage, we use the following intermediate form.

> **Definition 9** A ZX-diagram is in *green-Hadamard (GH) form* when every spider is a Z spider and every connection between spiders is either an ordinary wire or contains exactly one Hadamard box.

An arbitrary ZX-diagram is converted into GH form by applying colour change (h) to every X spider, introducing a Hadamard box on each incident wire, and cancelling adjacent pairs using (i2):



After these cancellations, every connection between Z spiders contains at most one Hadamard box. We draw a connection with no Hadamard box as an ordinary black wire, and one with a Hadamard box as a blue dashed Hadamard edge:



Graph-like form further restricts GH form so that its spiders and Hadamard connections define a simple graph, with a standard form at the boundary.

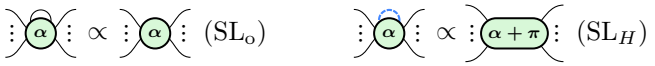
> **Definition 10** A ZX-diagram is *graph-like* when:
> - every spider is a Z spider;
> - spiders are connected only by Hadamard edges;
> - the underlying graph has no self-loops or parallel edges;
> - every input and output is connected to a spider; and
> - each spider is incident to at most one boundary wire.

A spider incident to an input or output is called a *boundary spider*; every other spider is *internal*.

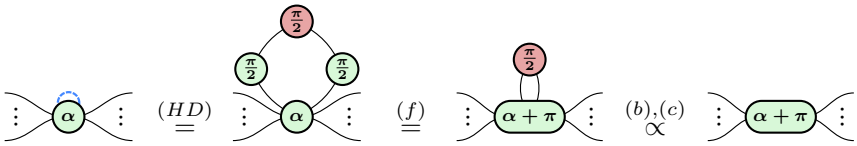
Throughout the remainder of the graph-like reduction, graphical rewrites are understood up to a non-zero scalar. We therefore write $\propto$ and omit scalar factors, which are immaterial for circuit optimisation.

Lemma 11 ***GH-to-graph-like conversion.*** *Every ZX-diagram in GH form can be transformed into an equivalent graph-like ZX-diagram.*

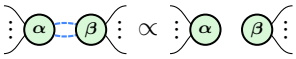
Proof. Let D be in GH form. Whenever two spiders are joined by an ordinary wire, fuse them along that wire. This removes one spider and turns every other edge between the pair into a self-loop. These self-loops can be removed using the following identities:



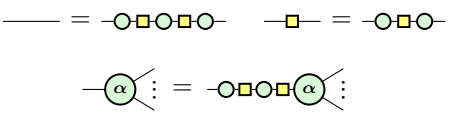
The first identity follows from spider fusion, and the second identity can be derived as follows:



Repeating this process removes every ordinary wire between spiders. Any remaining connections between two spiders are therefore Hadamard edges. Parallel Hadamard edges cancel in pairs using the Hopf law (a):



Finally, we enforce the boundary conditions. Two kinds of defect can remain. First, a boundary may not be connected to a Z spider. Such a component is either a bare wire or a wire containing a Hadamard box; in either case, applying identity (i1) and Hadamard cancellation (i2) from right to left inserts a phaseless Z spider. Second, several boundary wires may meet the same Z spider. Applying the same construction to all but one of these wires gives each boundary its own spider:



The resulting diagram is equivalent to the original diagram and satisfies all the requirements of a graph-like diagram [19]. □

Since every ZX-diagram can first be converted into GH form, Lemma 11 implies that every ZX-diagram admits an equivalent graph-like form (See Figure 7).

3.5 Graph states and AP form

By bending its input wires into outputs, every non-zero Clifford ZX-diagram can be viewed as a stabiliser state, up to a non-zero scalar. This correspondence, known as the *Choi–Jamiołkowski isomorphism*, lets us study Clifford ZX-diagrams using stabiliser theory [66]. We first describe stabiliser states through their Pauli generators, then introduce their graphical normal forms.

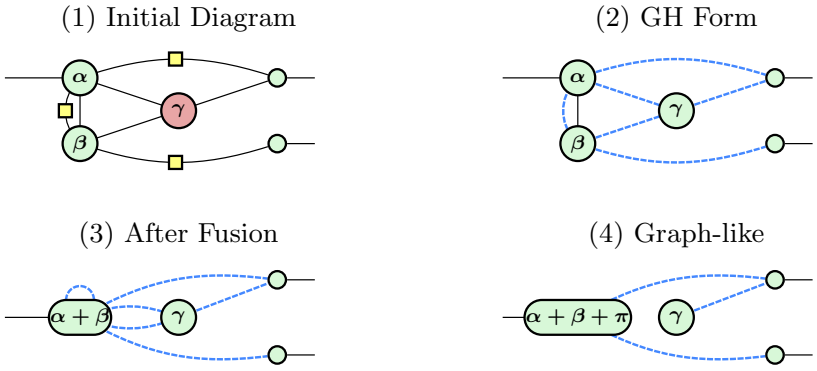


Figure 7: Conversion of a red-green ZX-diagram to GH form and then to graph-like form.

3.5.1 *Stabiliser states*

A Pauli operator P *stabilises* a state $|\psi\rangle$ when $P|\psi\rangle = |\psi\rangle$. Its Pauli stabiliser is

$$\mathrm{Stab}(|\psi\rangle) = \{P \in \mathcal{P}_n \mid P|\psi\rangle = |\psi\rangle\}. \tag{58}$$

A *stabiliser group* is a subgroup of the Pauli group that does not contain $-I$. Its elements are mutually commuting, signed Hermitian Pauli operators. The generators are *independent* if none can be written as a product of the others.

An n-qubit *stabiliser state* is the common $+1$ eigenstate of n independent commuting Pauli generators whose group excludes $-I$. These generators determine the state uniquely up to global phase [68]. Their products give all 2^n stabilising Paulis, so n signed Pauli strings provide an $O(n^2)$-bit description of the state.

> Example 11 For the Bell state in Figure 3,
>
> $$X_1X_2|\Phi^+\rangle = \frac{|11\rangle + |00\rangle}{\sqrt{2}} = |\Phi^+\rangle, \quad Z_1Z_2|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = |\Phi^+\rangle. \tag{59}$$
>
> Hence
>
> $$\mathrm{Stab}(|\Phi^+\rangle) = \langle X_1X_2, Z_1Z_2\rangle = \{I, X_1X_2, Z_1Z_2, -Y_1Y_2\}. \tag{60}$$
>
> The Z_1Z_2 condition restricts the state to the span of $|00\rangle$ and $|11\rangle$, while X_1X_2 fixes their relative sign. The product $-Y_1Y_2 = (X_1X_2)(Z_1Z_2)$ also illustrates that multiplying Pauli labels requires a sign update.

Stabiliser states are exactly those obtainable from $|0^n\rangle$ by Clifford circuits [28], [68]. Clifford conjugation transports their stabilisers with the state:

$$\mathrm{Stab}(U|\psi\rangle) = U\,\mathrm{Stab}(|\psi\rangle)U^\dagger. \tag{61}$$

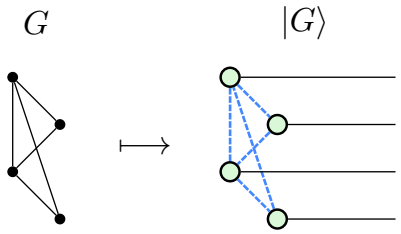
Indeed, $P|\psi\rangle = |\psi\rangle$ implies $\big(UPU^\dagger\big)U|\psi\rangle = U|\psi\rangle$.

3.5.2 *Graph states*

For a simple graph $G = (V, E)$ on n vertices, its *graph state* is prepared by placing each qubit in $|+\rangle$ and applying a CZ gate along each edge:

$$|G\rangle = \prod_{(u,v)\in E} \mathrm{CZ}_{u,v} \, |+\rangle^{\otimes n}. \tag{62}$$

In the ZX-calculus, this preparation is represented by one zero-phase Z spider and one output for every vertex, with a Hadamard edge for every edge of G [11]:

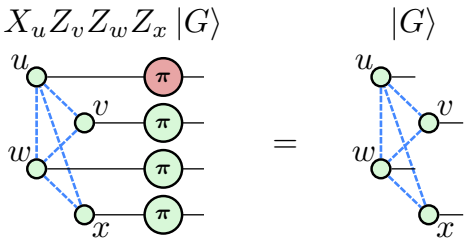


Graph states are stabiliser states. Applying X at a vertex and Z at each of its neighbours leaves the state unchanged, as illustrated in Equation 64. Each vertex therefore gives a stabiliser generator

$$P_v = X_v \prod_{u\in N(v)} Z_u, \quad v \in V, \tag{63}$$

so $\mathrm{Stab}(|G\rangle) = \langle P_v \mid v \in V \rangle$ [69].

For the graph G shown here, copying the red π spider through vertex u produces a green π spider at each neighbour, cancelling the three Z gates. The equality is exact:



(64)

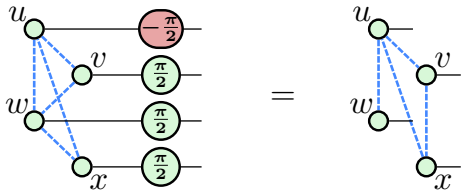
Graph states are useful because every stabiliser state can be obtained from a graph state by applying single-qubit Clifford operators. An n-qubit *local Clifford* has the tensor-product form $C_1 \otimes ... \otimes C_n$, with each C_j a one-qubit Clifford.

> Theorem 12 *(Van den Nest, Dehaene, and De Moor [70]): Every stabiliser state admits a* graph state with local Cliffords *(GSLC) representation*
>
> $$|\psi\rangle = (C_1 \otimes ... \otimes C_n)|G\rangle \tag{65}$$
>
> *up to global phase.*

The GSLC representation is not unique: local complementations can change the graph while local Clifford corrections preserve the represented state up to global phase. A *local complementation* at a vertex u complements the subgraph induced by $N_G(u)$, leaving every vertex and all other adjacencies unchanged. The corresponding

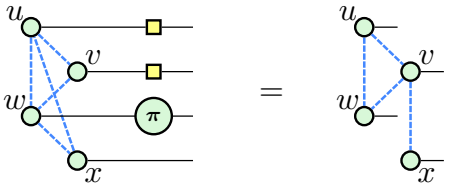
local Clifford applies $X_{-\frac{\pi}{2}}$ to u and $Z_{\frac{\pi}{2}}$ to each of its neighbours [19]. The example in Equation 66 keeps every graph-state vertex.

$$R_{X_u}(-\tfrac{\pi}{2})\, S_v\, S_w\, S_x\, |G\rangle \qquad |G \star u\rangle \tag{66}$$



A *pivot* along an edge uv is the composition $G \wedge uv = G \star u \star v \star u$. It preserves u and v while rearranging the edges among their neighbours. On graph states, pivoting is implemented by Hadamards on u and v and a Z_π gate on every common neighbour [19]. In Equation 67, the common neighbourhood is $\{w\}$, so the operation is $H_u H_v Z_w$.

$$H_u H_v Z_w\, |G\rangle \qquad |G \wedge uv\rangle \tag{67}$$



It turns out that local complementations completely characterise equivalence between graph states up to local Clifford operations.

> Theorem 13 *(Van den Nest, Dehaene, and De Moor [70]): Two graph states are related by local Clifford operators, up to global phase, if and only if their graphs are related by a sequence of local complementations.*

3.5.3 *Affine-with-phases form*

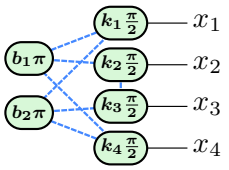
Affine-with-phases (AP) form provides another representation of stabiliser states, using affine binary constraints and a phase polynomial. It arises during Clifford simplification, after local complementation and internal pivoting, before boundary pivots convert the diagram to GSLC form [1].

> Definition 14 A graph-like Clifford state is in *affine-with-phases* (AP) form when every internal spider has phase $b_a\pi$, with $b_a \in \mathbb{F}_2$, and is adjacent only to boundary spiders [71].

Each internal spider, together with its Hadamard edges to the boundary, can be interpreted as a projector (defined in Section 3.3) applied to $|+\rangle^{\otimes n}$. Together, these commuting projectors select an affine subspace of $\mathbb{F}_2^n$. The phases and Hadamard edges on the boundary can be interpreted as a diagonal Clifford unitary built from S and CZ gates [61].

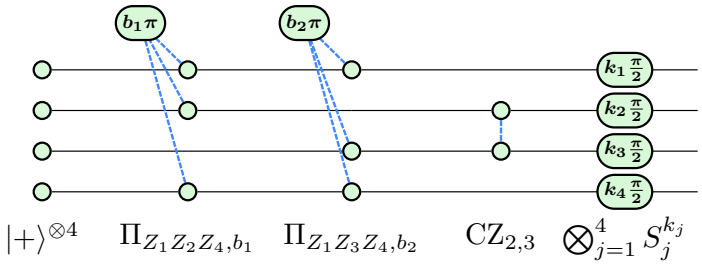
Example 12 This AP diagram has affine support $\mathcal{A}$ and phase polynomial q_D. Its decomposition prepares $|+\rangle^{\otimes 4}$, applies parity projectors, and then applies $\mathrm{CZ}_{2,3}$ and local S gates, up to a non-zero scalar:

$$[\![D]\!] \propto \sum_{\mathbf{x}\in\mathcal{A}} i^{q_D(\mathbf{x})} |x_1, \ldots, x_4\rangle$$



$$\mathcal{A} = \left\{ \mathbf{x} \in \mathbb{F}_2^4 \;\middle|\; \begin{array}{l} x_1 \oplus x_2 \oplus x_4 = b_1, \\ x_1 \oplus x_3 \oplus x_4 = b_2 \end{array} \right\}$$

$$q_D(\mathbf{x}) = \underbrace{\sum_{j=1}^{4} k_j x_j}_{\substack{\text{boundary phases} \\ \text{linear}}} + \underbrace{2x_2x_3}_{\substack{\text{boundary H edge} \\ \text{quadratic}}} \pmod 4$$



A benefit of AP form over GSLC form is that it admits a unique normal form. Writing the affine equations defining $\mathcal{A}$ in reduced row-echelon form (RREF), eliminating bound variables from q_D, and fixing coefficient representatives gives the *reduced AP form*. For every non-zero state with a fixed boundary ordering, this form is unique up to a non-zero scalar. See Kissinger and van de Wetering [61], Section 5.5, for details.

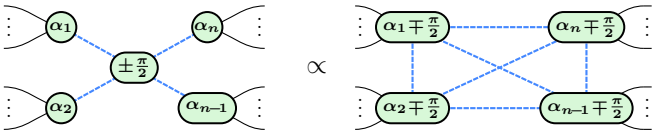
3.6 Graph-like simplification

The `full_reduce` procedure, described by Duncan et al. [19] and later extended by Kissinger and van de Wetering [20], simplifies graph-like ZX-diagrams by removing spiders and combining phases. We now describe these rewrites and examine the resulting forms for CNOT, Clifford, and Clifford+T circuits.

3.6.1 *Clifford reduction to GSLC form*

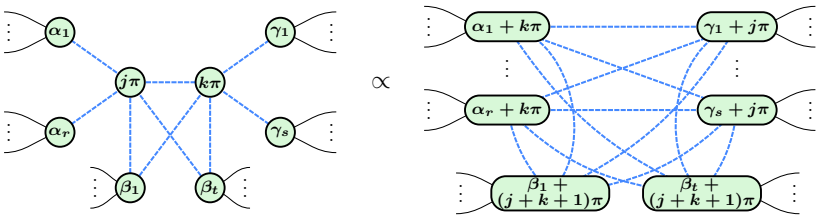
We start with the three rewrite rules used by Duncan et al. [19], which we call the *Clifford simplification rules*.

The local complementation rule (LC) follows from the graph-state identity of the same name in Equation 66. Given an internal proper-Clifford spider u, (LC) applies the local-complementation graph transformation, changes the phases of its neighbours $N(u)$, and then removes the spider.



(LC)

Exhaustive application of (LC) removes every internal spider whose phase is an odd multiple of $\frac{\pi}{2}$. The remaining internal Clifford spiders are Pauli, and the pivot rule (P1) removes any adjacent pair u and v. In the diagram, the sets of spiders $\{\alpha_i\}$, $\{\gamma_i\}$, and $\{\beta_i\}$ denote u's exclusive neighbourhood, v's exclusive neighbourhood, and their shared neighbourhood $N(u) \cap N(v)$, respectively. The rule applies the displayed Pauli phase updates and toggles the connections between these three neighbourhoods. When the shared neighbourhood $\{\beta_i\}$ is empty, this is analogous to the bialgebra rule.



(P1)

After exhaustive application of (LC) and (P1), every remaining internal spider has phase 0 or π and is connected only to boundary spiders. Bending the inputs into outputs gives a state in AP form (Definition 14) [1]. Figure 8 illustrates this for a three-qubit circuit.

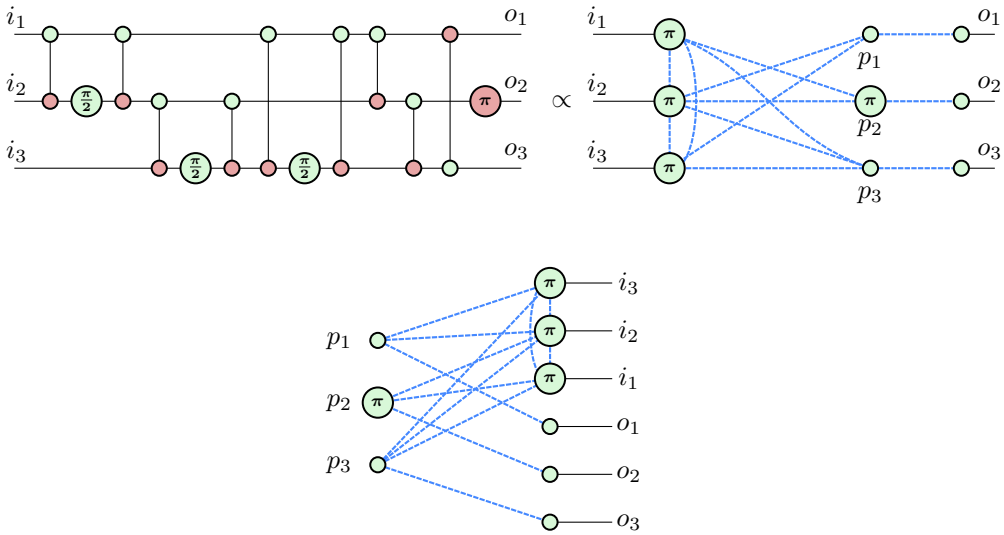
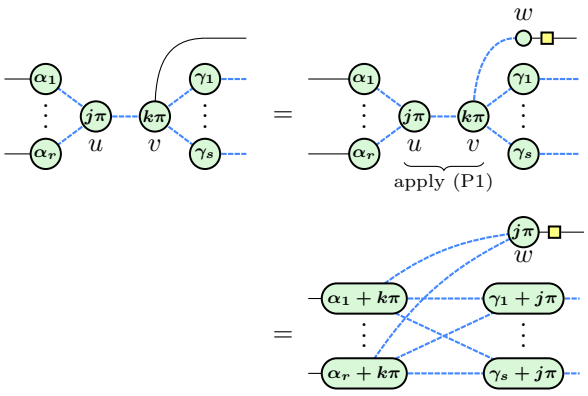


Figure 8: Simplification of a Clifford circuit (top left) to a graph-like diagram (top right), up to a non-zero scalar. Bending the inputs into outputs via the Choi–Jamiołkowski correspondence gives the AP form below, where the internal spiders p_1, p_2, p_3 represent parity projectors.

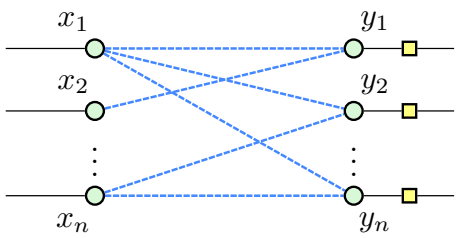
The *boundary-pivot rule* removes the remaining internal Pauli spiders, converting AP form to GSLC form. First, we unfuse a neighbouring boundary wire so that its boundary spider becomes internal, then apply (P1). The remaining spider becomes the new boundary:



(BP)

Together, exhaustive application of the Clifford simplification rules eliminates all internal spiders in a Clifford ZX-diagram. This yields the following two results.

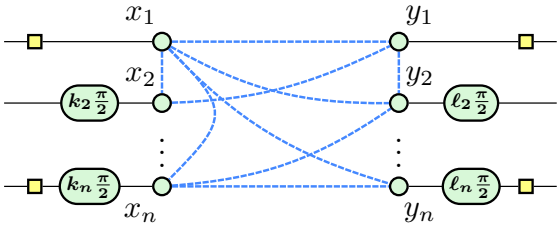
Theorem 15 *(Normal form for CNOT circuits) Let C be an n-qubit CNOT circuit, and let D_C and $A_C \in \mathrm{GL}(n, \mathbb{F}_2)$ be its associated graph-like ZX-diagram and parity matrix. The procedure using* (LC)*,* (P1)*, and* (BP) *through output-boundary spiders rewrites D_C to the unique bipartite GSLC form below. Its Hadamard-edge connections have A_C as their biadjacency matrix.*



Proof. This is the graph-like version of the parity normal form described by Kissinger and van de Wetering [61], using a slightly different ruleset.

Proper Clifford spiders do not feature in the initial diagram and cannot be introduced by (P1) or (BP), so (LC) is never applied. Applying (P1) until no match remains eliminates every adjacent pair of internal Pauli spiders. Each remaining internal Pauli spider is then eliminated by a boundary pivot through an output-boundary spider; this introduces the displayed Hadamard on that output. Repeating these steps leaves the bipartite GSLC graph shown above. It is straightforward to check that this form is unique over the basis states. □

Theorem 16 *(Pseudo-normal form for Clifford circuits) Let D be the ZX-diagram corresponding to a Clifford circuit. Repeated application of* (LC), (P1), *and* (BP) *transforms D into a graph-like ZX-diagram D' with no internal spiders, up to single-qubit unitaries on its inputs and outputs.*

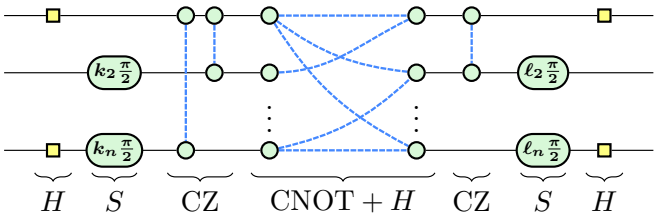


Bending the input wires into outputs, as in Figure 8, gives a GSLC representation of the corresponding Choi state, with one graph vertex for each input and output wire. Each displayed phase is an independent multiple of $\frac{\pi}{2}$: the coefficients $k_i, \ell_i \in \mathbb{Z}_4$ are chosen independently for each boundary vertex. The Hadamard edges on the boundary correspond to CZ *gates. This representation gives the following eight-layer decomposition of Clifford circuits:*

$$H + S + \mathrm{CZ} + \mathrm{CNOT} + H + \mathrm{CZ} + S + H,$$

Here, each H layer contains at most one Hadamard per wire, each S layer contains between zero and three S gates per wire, and the CZ *and* CNOT *layers may contain arbitrarily many gates of the indicated type.*

Proof. The GSLC form is obtained as in Theorem 15; see Duncan et al. [19], Theorem 5.4, for the proof. The eight-layer circuit form follows by expressing edges between input-boundary spiders, and those between output-boundary spiders, as CZ gates. The remaining graph is then the CNOT normal form described in Theorem 15, but without the final Hadamard layer. The resulting sequence of layers is displayed explicitly below.



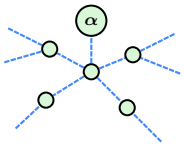
□

3.6.2 *Phase-gadget rewrites*

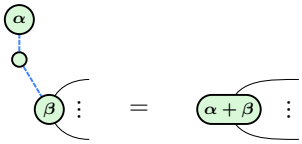
When the Clifford simplification rules are applied to non-Clifford ZX-diagrams, internal non-Clifford spiders can remain. Kissinger and van de Wetering [20] extend the Clifford simplification strategy with four rules that turn such phases into gadgets and then absorb or combine those gadgets. We call these rules the *phase-gadget*

extension, and their union with the Clifford simplification rules the *full-reduction rules*.

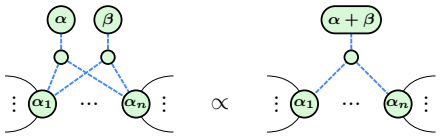
Recall from Section 3.3 that, in graph-like form, a phase gadget consists of an arity-one phase spider joined by a Hadamard edge to a zero-phase body spider. Its body can occur anywhere in the graph-like skeleton, with its Hadamard-edge neighbours as targets.



Two direct simplifications act on gadgets. A gadget with one target is absorbed into that target by (ID), adding their phases. Two gadgets with the same complete target set fuse by (GF), again adding their phases modulo 2π.

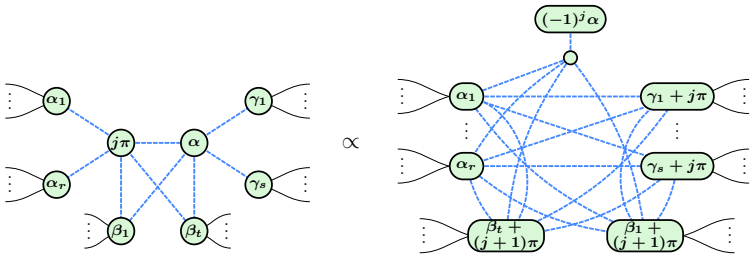


(ID)



(GF)

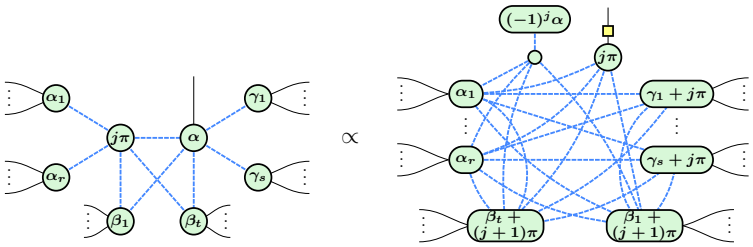
The remaining two rules handle an internal Pauli spider $j\pi$, where $j \in \{0, 1\}$, adjacent to a spider of arbitrary phase α. As in (P1), the spiders labelled α_i, γ_i, and β_i are adjacent exclusively to $j\pi$, exclusively to α, and to both, respectively. If α is internal, (P2) removes $j\pi$ and turns α into a phase gadget. If α is a boundary spider, (P3) moves its boundary and creates the gadget. Taken together, (P2) and (P3) subsume the boundary-pivot step (BP), extending it to arbitrary-phase neighbours in the internal and boundary cases. The displayed signs and boundary Hadamard are part of these projective rules [20].



(P2)

```
procedure full_reduce(D)
repeat
    apply (LC) exhaustively
    apply (P1), (P2), and (P3) exhaustively
    remove every Clifford phase gadget using (LC) and (P1)
    apply (ID) and (GF) exhaustively
until neither (ID) nor (GF) was applied in the pass
return D
```

Algorithm 1: Full reduction of a graph-like ZX-diagram.



(P3)

3.6.3 *Full reduction and phase teleportation*

The Clifford and phase-gadget rules studied above can be combined into the *full-reduction* algorithm of Kissinger and van de Wetering [20], summarised in Algorithm 1.

Every step removes a spider or phase gadget, so the strategy terminates. For a graph-like input with n spiders, it performs at most n simplifying steps. A step may toggle edges among neighbourhoods of size at most n, giving an $O(n^3)$ upper bound in elementary graph operations [20].

The procedure in Algorithm 1 returns a diagram in reduced gadget form, defined by Kissinger and van de Wetering [20] as follows:

> Definition 17 A graph-like ZX-diagram is in *reduced gadget form* when:
> - every internal spider is non-Clifford or belongs to a non-Clifford phase gadget;
> - every phase gadget has more than one target; and
> - no two phase gadgets have the same target set.

Phase teleportation can be used to combine non-Clifford phases without extracting a new circuit or changing the original circuit structure. PyZX's `teleport_reduce` applies `full_reduce` to a copy of D while maintaining an auxiliary map Φ from the resulting phase combinations to their original circuit locations. If gadgets fuse or their phases change, Φ is updated accordingly. The final phase values are then transferred back to the original circuit, leaving its connectivity and non-phase gates unchanged [20]. We study phase teleportation and its optimality for parametrised ZX-diagrams in Chapter 4.

3.7 Pauli webs

Bombín et al. [72] introduced Pauli webs as a graphical method for identifying the checks and stabilisers of ZX network diagrams. Developed for stabiliser fault tolerance, Pauli webs track Pauli information through Clifford ZX diagrams. The Pauli labels on the inputs and outputs of a diagram record the action of a Clifford map on Paulis and collectively determine its tableau. We later use this correspondence as an interface between ZX diagrams, tableau synthesis, and circuit extraction.

Definition 18 Let D be a Clifford ZX-diagram. A *Pauli web* on D assigns a label from $\{I, X, Y, Z\}$ to every half-edge. We draw X support in red, Z support in green, and Y support in both colours. The labelling is *valid* when:

- the labels agree across an ordinary edge, while across a Hadamard edge they are related by Hadamard conjugation, ignoring phase, so X and Z are exchanged while I and Y are unchanged;
- at a spider with phase 0 or π, an even number of incident half-edges have support in the spider's own colour, while support in the opposite colour occurs on either all or none of them; and
- at a spider with phase $\pm\frac{\pi}{2}$, either an even number of incident half-edges have support in its own colour and none in the opposite colour, or an odd number have support in its own colour and all have support in the opposite colour.

Here the own-colour support of a Z-spider is Z, and that of an X-spider is X [73].

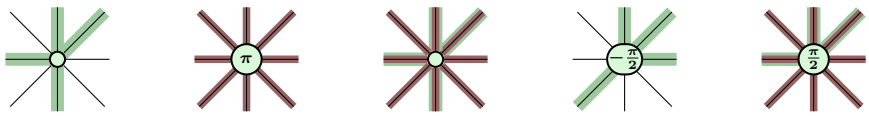


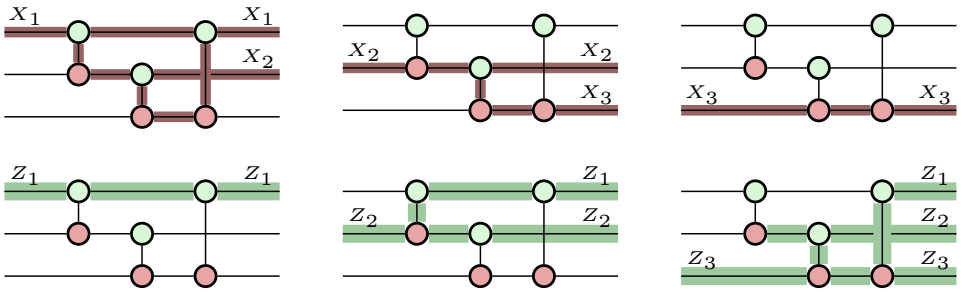
Figure 9: Example Pauli webs on single spiders. Green and red highlighting denote Z- and X-type support, respectively; overlapping highlighting denotes Y support. Adapted from Rodatz, Poór, and Kissinger [73].

For a Clifford unitary U, a web with input Pauli P_{in} and output Pauli P_{out} records

$$UP_{\text{in}}U^{\dagger} = P_{\text{out}}. \tag{75}$$

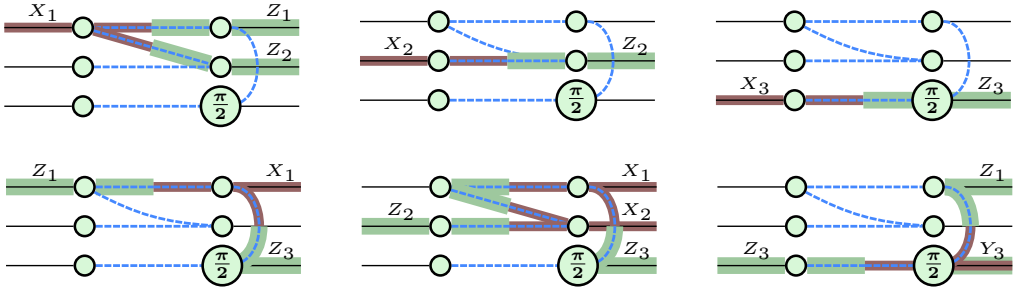
Webs on the CNOT GSLC normal form of Theorem 15 never mix X- and Z-type support. If A is its biadjacency matrix, the generator webs assemble the phase-free tableau $\operatorname{diag}(A^T, A^{-1})$ from Equation 33. Thus the CNOT parity map already contains the complete phase-free Clifford action.

EXAMPLE 13 The following are the six generator webs of the three-qubit CNOT circuit from Example 1, ordered as X_1, X_2, X_3 (top) and Z_1, Z_2, Z_3 (bottom). Their output labels give the phase-free Clifford action of this circuit.



For Clifford circuits, a Pauli generator can be mapped to a string containing both X and Z, as Example 14 shows.

EXAMPLE 14 The following six webs are obtained by propagating each input Pauli generator through a three-qubit Clifford GSLC diagram, ordered as X_1, X_2, X_3 (top) and Z_1, Z_2, Z_3 (bottom). Each output Pauli supplies the corresponding row of its operation tableau.



Reading the output labels in the tableau row order $(X_1, X_2, X_3, Z_1, Z_2, Z_3)$ gives

$$\begin{array}{c} \\ X_1 \\ X_2 \\ X_3 \\ Z_1 \\ Z_2 \\ Z_3 \end{array} \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & z_1 & z_2 & z_3 \end{array} \\ \left[\begin{array}{cccccc|c} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{array}\right. \end{array} \begin{array}{c} r \\ \left.\begin{array}{c} \\ \\ \\ \\ \\ \end{array}\right] \end{array} \begin{array}{r} \\ Z_1Z_2 \\ Z_2 \\ Z_3 \\ X_1Z_3 \\ X_1X_2Z_3 \\ Z_1Y_3 \end{array}$$

Pauli webs thus let us read the Clifford tableau directly from a ZX-diagram. For non-Clifford diagrams, the ZX-flow criterion uses a generalisation called Pauli *semiwebs*. Diagrams satisfying this criterion can be efficiently extracted into quantum circuits [74].

Chapter IV

Optimal Compilation of Parametrised Quantum Circuits

Parametrised quantum circuits (PQCs), defined in Section 2.3.5, are families $C(\vec{\theta})$ obtained by varying gate angles $\vec{\theta}$ within a fixed circuit structure, known as an *ansatz*. These circuits are used in *variational quantum algorithms* (VQAs), where a classical optimiser varies the gate angles $\vec{\theta}$ to optimise an objective evaluated on a quantum processor [38]. Prominent examples include the variational quantum eigensolver (VQE), which minimises the expected energy of a Hamiltonian over an ansatz family [75], and the quantum approximate optimisation algorithm (QAOA), whose alternating operator layers define an ansatz for combinatorial optimisation problems such as MAX-CUT [76]. Trainable PQCs are also used as models in quantum machine learning [77-78].

A useful ansatz must be expressive enough to represent good solutions while remaining sufficiently shallow and trainable. The expressiveness of an ansatz can be characterised through its dynamical Lie algebra [79], or through an effective quantum dimension derived from the quantum Fisher information [80]. However, expressiveness and trainability can be in tension: as an ansatz approaches a unitary 2-design, its gradient variance can decrease and its cost landscape flatten, making optimisation more difficult [81]. An ansatz can contain redundant parameters: removing them preserves the circuit family and hence its expressiveness.

This chapter studies parameter redundancy through compilation. Given a PQC family $C(\vec{\theta})$, we ask whether it can be compiled into a circuit $C'(\vec{\varphi})$ with fewer inde-

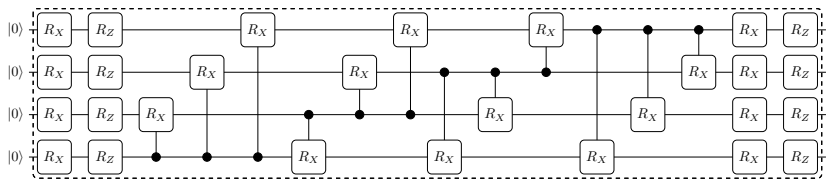
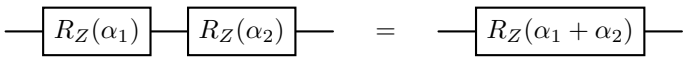


Figure 10: An example of a parametrised quantum circuit ansatz. Circuit 6 of Figure 2 from Sim et al. [82]

pendent symbolic angles, such that every original assignment $\vec{\theta}$ has a corresponding reduced assignment $\vec{\varphi}$ implementing the same operation up to global phase.

The simplest example of such a reduction occurs when two adjacent parametrised rotations are about the same axis:



The same principle merges multi-qubit phase rotations that act on the same parity function, as in sum-over-paths methods. In a Pauli-exponential representation, it likewise merges rotations about the same Pauli axis once commutation relations bring them together. We refer to these techniques collectively as *phase folding* [1].

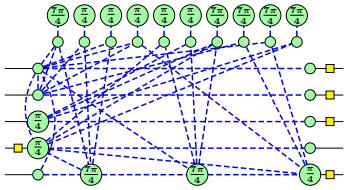
Eliminating redundant parameters can reduce both circuit implementation costs and training time. Fusing rotations can expose further cancellations and reduce the number of error-prone two-qubit gates [9]. During training, fewer independent parameters generally require fewer circuit evaluations for each full-gradient estimate [83], reducing the time per optimisation step. The classical optimiser also searches a lower-dimensional space after removing parameter directions that leave the implemented operation unchanged.

The same compilation techniques can also be used to reduce T-count in Clifford+T circuits. Since $T = R_Z(\frac{\pi}{4})$, each T gate can be treated temporarily as an independent symbolic phase, provided that every compilation step remains valid for all assignments of those phases. PyZX's phase teleportation [20] and TOpt [40] follow this principle, reducing T-count through phase folding without using identities specific to the angle $\frac{\pi}{4}$.

Although both algorithms reduce T-count through phase folding, they use very different representations. PyZX applies MBQC-inspired graph rewrites to ZX-diagrams, whereas TOpt represents T gates as Pauli exponentials and merges them according to Clifford commutation relations. Nevertheless, they produce circuits with matching T-counts on the benchmark circuits in Figure 11. Simmons explains this coincidence by relating graph-like ZX-diagrams to Pauli Dependency DAGs through a bisimulation based on Pauli flow. Another possible explanation is that both methods attain the same optimum within the class of algorithms restricted to phase folding. This had not been proved before this work.

For parametrised Clifford circuits in which every parameter occurs on a unique rotation gate, we make five contributions.

Chapter Contributions



(a) PyZX

Ins	Outs	Sign
X	Y	$+$
X	X	$+$
Z	Z	$+$
Z	Z	$+$

$$(Z_1I_2,-\alpha_0) \longrightarrow (X_1I_2,\alpha_2) \longrightarrow (Y_1X_2,-\alpha_3) \longrightarrow (Z_1I_2,-\alpha_6)$$
$$(Y_1X_2,-\alpha_5)$$
$$(I_1Z_2,-\alpha_1) \longrightarrow (I_1X_2,\alpha_4) \longrightarrow (I_1Y_2,-\alpha_7)$$

(b) TOpt

	Adder8	VBE3	QCLA7	GF24	GF25	GF26	GF27	GF28
PyZX	173	24	237	68	115	150	217	264
TOpt	173	24	237	68	115	150	217	264

Figure 11: Illustration of the intermediate representations used in [20] and [40], and their identical final T-counts for various benchmark circuits. Diagram (b) is taken from Simmons. [39]

1. **Reduction model.** We introduce *affine reductions* (Definition 31) and *affine parsimonious reductions* (Definition 32) between parametrised quantum circuits. In an affine reduction, each output angle is an integer-affine combination of the input angles. An affine reduction is parsimonious if no input parameter is copied to more than one output location. Together, these notions formalise the black-box transformations generated by phase folding and Clifford rewrites.
2. **Parametrised Clifford completeness.** We extend the completeness of the Clifford ZX-calculus (Theorem 7) to parametrised Clifford diagrams with uniquely occurring, non-trivial parameters. If two such diagrams are equal up to a non-zero scalar for every parameter assignment, then there exists a single *uniform* sequence of Clifford rewrites that proves their equality for all assignments (Theorem 28).
3. **Optimality of phase teleportation.** We prove that the phase-teleportation algorithm of [20], reviewed in Section 3.6.3, efficiently produces a circuit with the minimum parameter count obtainable by an affine parsimonious reduction. Each output angle in the reduced circuit is a signed sum of input angles plus a constant Clifford offset (Theorem 39).
4. **Explaining the matching T-counts of PyZX and TOpt.** We show that PyZX's phase teleportation [20] and TOpt [40] attain the same optimum when T gates are treated as black-box phase gates. This explains their matching benchmark T-counts: both are optimal under phase folding, without using T-specific identities (Figure 11; Section 4.4.1).
5. **MBQC optimality and hardness under relaxed assumptions.** We extend the optimality result to unitary measurement-based computations with gflow in which each arbitrary-angle measurement has a distinct parameter (Theorem 42). Parameter optimisation is NP-hard if the discrete gate set includes T gates (Proposition 46) or if parameters may repeat in post-selected Clifford circuits (Proposition 49). The complexity of repeated parameters in ordinary unitary circuits remains open.

The chapter proceeds as follows. Section 4.1 introduces parametrised circuits in the ZX-calculus and establishes completeness of the Clifford+parameters fragment. Section 4.2 defines parametrised Clifford circuits and the affine parsimonious comparison class. Section 4.3 applies phase teleportation to symbolic parameters, establishes the required properties of parsimonious affine reductions, and proves that the resulting parameter count is optimal. Section 4.4.1 compares the PyZX and TOpt realisations of phase folding. Section 4.4.2 extends the optimality result to measurement-based quantum computations with gflow, while Section 4.5 studies what happens when the assumptions on the gate set and parameter occurrence are relaxed. Finally, Section 4.6 summarises the limitations and open problems. Supporting results about parameter-dependent scalars are collected in Appendix B.

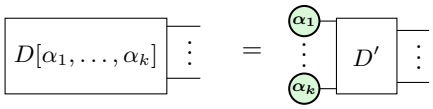
4.1 Parametrised Clifford completeness

We allow independent parameters as spider phases alongside Clifford phases, giving *Clifford+phase diagrams*. The fusion rule holds for arbitrary phase labels, so any adjacent spiders of the same colour in a Clifford+phase diagram may be fused, whether their phases are Clifford or parametrised. The completeness argument treats each parametrised spider as an input to an otherwise Clifford diagram and shows that the same rewrite system remains complete for this fragment.

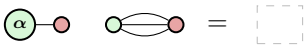
Previous work has used the ZX-calculus to analyse PQC expectation values, gradients, and ansatz structure [84], and to check equivalence between parametrised circuits symbolically [85]. We ask whether the Clifford rewrite rules can prove every valid equality between Clifford+phase diagrams using a single sequence of rewrites that works for all parameter values.

Throughout this chapter, phase parameters take values in $S^1 := \frac{\mathbb{R}}{2\pi\mathbb{Z}}$. Parameter assignments are nevertheless written as $\vec{\alpha} \in \mathbb{R}^k$, with every component understood modulo 2π.

Definition 19 A *parametrised diagram with k parameters* D is a ZX-diagram that is entirely Clifford, apart from the phases $\vec{\alpha} = (\alpha_1, ..., \alpha_k)$ that are each present *at most once* in the diagram. For each choice of $\alpha_j \in S^1$, the following holds:



where the equality is exact, including scalar factors, and D' is a Clifford diagram. The diagram is drawn as a state, with no input wires, without loss of generality. If some phase α_j is not present in $D[\alpha_1, ..., \alpha_k]$, a Clifford diagram D' can still be found because



The notation D denotes the ZX-diagram with abstract parameters and $D[\vec{\alpha}]$ the diagram instantiated at particular values $\alpha_1, ..., \alpha_k$.

4.1.1 *Exact equality*

Consider two Clifford+phase diagrams D_1 and D_2 with the same parameters, and let D_1' and D_2' be their associated Clifford diagrams defined above. Suppose that $D_1[\vec{\alpha}] = D_2[\vec{\alpha}]$ for every parameter assignment $\vec{\alpha}$. We show that $D_1' = D_2'$, apply Clifford completeness, and reattach the parametrised phases to obtain a rewrite valid for every $\vec{\alpha}$.

Definition 20 Write $|+_\alpha\rangle = |0\rangle + e^{i\alpha}|1\rangle$, graphically:

Note that these are unnormalised states, so $|+_0\rangle = \sqrt{2}|+\rangle$.

Lemma 21 *Let $\alpha, \beta \in \mathbb{R}$ be two different phases modulo 2π. Then $|+_\alpha\rangle$ and $|+_\beta\rangle$ form a basis. Writing $|+_\gamma\rangle = a|+_\alpha\rangle + b|+_\beta\rangle$, the coefficients satisfy $a + b = 1$ and*

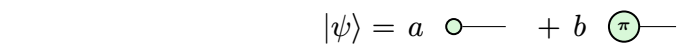
$$a = \frac{e^{i\gamma} - e^{i\beta}}{e^{i\alpha} - e^{i\beta}}, \quad b = \frac{e^{i\alpha} - e^{i\gamma}}{e^{i\alpha} - e^{i\beta}}. \tag{76}$$

In the special case $\alpha = 0$ and $\beta = \pi$, $a - b = e^{i\gamma}$.

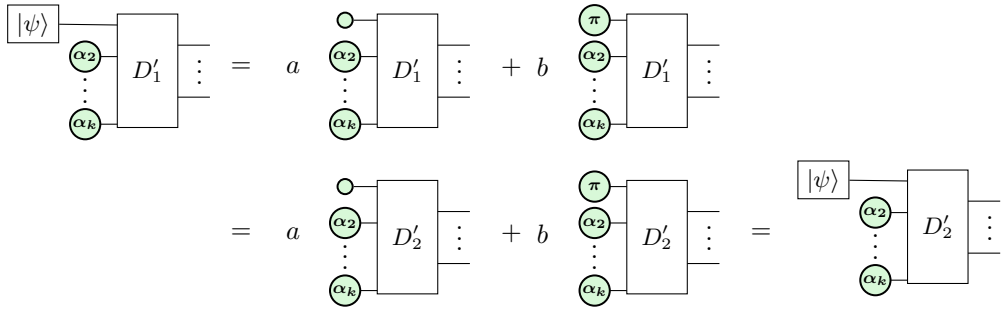
Because each parameter occurs only once, the states $|+_{\alpha_j}\rangle$ can be varied independently. By Lemma 21, choosing two distinct values for each phase gives a tensor-product basis for the full input space of D'. The corresponding parameter assignments therefore determine the action of D' on that space. If a parameter occurred more than once, several inputs would be constrained to receive states with the same phase, and the resulting assignments might span only their symmetric subspace. Allowing repeated parameters also makes parameter optimisation NP-hard for post-selected circuits (Section 4.5).

Proposition 22 *The Clifford rules are complete for parametrised Clifford diagrams. Let D_1 and D_2 be parametrised diagrams with the same number of parameters and suppose $D_1[\vec{\alpha}] = D_2[\vec{\alpha}]$ for every choice of $\alpha_1, ..., \alpha_k$. Then $D_1' = D_2'$, and $D_1[\vec{\alpha}]$ and $D_2[\vec{\alpha}]$ can be uniformly rewritten into one another using the Clifford rewrite rules of the ZX-calculus.*

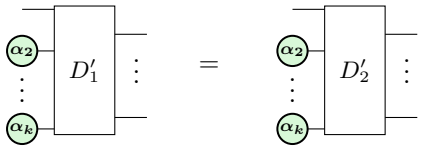
Proof. Let $|\psi\rangle = a|+_0\rangle + b|+_\pi\rangle$ be an arbitrary, not necessarily normalised, qubit state. Graphically,



Hence:



Since the two linear maps agree on every state plugged into this input, they are equal:



Repeating this argument for each remaining parameter gives $D_1' = D_2'$. These are Clifford diagrams representing the same linear map, so Clifford completeness provides a sequence of Clifford rewrites from D_1' to D_2'. Restoring the parametrised phases does not affect the applicability of these rewrites, giving a uniform rewrite from $D_1[\vec{\alpha}]$ to $D_2[\vec{\alpha}]$. □

Remark 23 The proof does not require the diagrams to be equal for every choice of α. It suffices for the diagrams to agree on two different values of α_j for each j. This is because $\{|+_\alpha\rangle, |+_\beta\rangle\}$ forms a basis of the qubit state space if $\alpha \neq \beta$. This is a special case of verifying parametrised equations using a finite number of cases [86]. In particular, if a parameter occurs k times, equality must be verified at $k+1$ different values of that parameter. This is because the symmetric subspace of k qubits is $(k+1)$-dimensional.

4.1.2 *Equality up to a scalar*

For circuit equivalence, we need equality up to global phase. More generally, we allow ZX-diagrams to differ by a scalar:

$$D_1[\vec{\alpha}] = \lambda(\vec{\alpha}) D_2[\vec{\alpha}] \tag{77}$$

for an arbitrary non-zero scalar $\lambda(\vec{\alpha})$. The dependence of this scalar on the parameters introduces an additional complication: a parameter may be redundant up to scalar. Under exact equality, two parameter values determine the action of the underlying Clifford diagram on a basis. Up to scalar, the proportionality factor may depend on the parameter value. If varying a parameter changes only this factor, its different values provide no information about the relative scaling of the corresponding basis states. Such a parameter is called *trivial*. We will show that $\lambda(\vec{\alpha})$ is constant when all parameters are non-trivial. We can then apply exact parametrised completeness up to a fixed Clifford scalar.

Definition 24 Let $D\left[\alpha, \vec{\delta}\right]$ be a parametrised diagram. The parameter α is *trivial* if there is a Clifford assignment $\vec{\delta} \in (\frac{\pi}{2}\mathbb{Z})^m$ for the other parameters and two values $\theta \neq \varphi \bmod 2\pi$ such that $D\left[\theta, \vec{\delta}\right] \propto D\left[\varphi, \vec{\delta}\right]$. Otherwise α is *non-trivial*.

Lemma 25 *If α is trivial in $D\left[\alpha, \vec{\delta}\right]$, then for the same fixed Clifford assignment $\vec{\delta}$ the diagrams $D\left[\theta, \vec{\delta}\right]$ are mutually proportional for all values of θ.*

Proof. Since α is trivial, choose distinct β, γ such that $D\left[\beta, \vec{\delta}\right] = \mu D\left[\gamma, \vec{\delta}\right]$. For an arbitrary θ, Lemma 21 supplies a, b with $|+_\theta\rangle = a|+_\beta\rangle + b|+_\gamma\rangle$. Linearity in the open parameter input gives

$$D\left[\theta, \vec{\delta}\right] = aD\left[\beta, \vec{\delta}\right] + bD\left[\gamma, \vec{\delta}\right] = (a\mu + b)D\left[\gamma, \vec{\delta}\right]. \tag{78}$$

Thus every value is proportional to the fixed diagram $D\left[\gamma, \vec{\delta}\right]$. □

Proposition 26 *Every uniquely occurring parameter of a unitary circuit built from phase gates $R_Z(\alpha_j)$ is non-trivial in its ZX translation.*

Proof. Fix the other parameters and isolate one occurrence as $C(\alpha) = U_2 R_Z(\alpha) U_1$, where U_1 and U_2 do not depend on α. If $C(\alpha') \propto C(\alpha)$, then $C(\alpha')^\dagger C(\alpha)$ is proportional to the identity. However,

$$C(\alpha')^\dagger C(\alpha) = U_1^\dagger R_Z(\alpha - \alpha') U_1. \tag{79}$$

Conjugation preserves whether an operator is scalar, and $R_Z(\alpha - \alpha')$ is scalar only when $\alpha = \alpha' \bmod 2\pi$. Thus two distinct parameter values modulo 2π cannot give proportional circuit operators. □

Proposition 27 *Let D_1 and D_2 be parametrised diagrams with the same parameters, all non-trivial in D_1. If*

$$D_1[\vec{\alpha}] = \lambda(\vec{\alpha}) D_2[\vec{\alpha}] \tag{80}$$

for every $\vec{\alpha}$ and some function $\lambda : \mathbb{R}^k \to \mathbb{C} \setminus \{0\}$, then there is a constant $\lambda' \in \mathbb{C} \setminus \{0\}$ such that $D_1' = \lambda' D_2'$.

The proof is given in Section B.1.1.

Theorem 28 *Let D_1 and D_2 be parametrised Clifford diagrams with the same parameters, all non-trivial in D_1, and suppose $D_1[\vec{\alpha}] = \lambda(\vec{\alpha})D_2[\vec{\alpha}]$ for every $\vec{\alpha}$, where $\lambda(\vec{\alpha}) \neq 0$. Then there is a Clifford scalar diagram S such that $D_1' = S \otimes D_2'$, and the Clifford ZX rules uniformly rewrite $D_1[\vec{\alpha}]$ into $S \otimes D_2[\vec{\alpha}]$.*

Proof. By Proposition 27, there is a constant non-zero scalar μ such that $D_1' = \mu D_2'$. If both open diagrams are zero, take $S = 1$. Otherwise, the tensor-product basis from Proposition 22 gives a Clifford assignment $\vec{\delta} \in \{0, \pi\}^k$ for which $D_2\left[\vec{\delta}\right] \neq 0$. At this assignment,

$$D_1\left[\vec{\delta}\right] = \mu D_2\left[\vec{\delta}\right]. \tag{81}$$

Both evaluations are Clifford diagrams, so μ is represented by a Clifford scalar diagram S [62]. Hence $D_1[\vec{\alpha}] = S \otimes D_2[\vec{\alpha}]$ for every parameter assignment. Exact parametrised completeness (Proposition 22) supplies a single sequence of Clifford rewrites between these diagrams, uniformly in the parameters. □

4.2 Parametrised Clifford circuits and reductions

We restrict the PQCs of Section 2.3.5 to Clifford gates and parametrised Z-phase gates, and compare circuits up to a parameter-dependent global phase.

Definition 29 A *parametrised Clifford circuit* (PCC) with k parameters is a circuit consisting of Clifford gates and Z-phase gates $R_Z(\alpha_j)$, where each parameter α_j occurs on exactly one phase gate.

Definition 30 A parametrised diagram D_1 with parameter space $(S^1)^k$ *reduces* to a parametrised diagram D_2 with parameter space $(S^1)^l$ when there exists a function $f : (S^1)^k \to (S^1)^l$ such that

$$D_1[\vec{\alpha}] = \lambda(\vec{\alpha})D_2[f(\vec{\alpha})] \tag{82}$$

for all $\vec{\alpha}$, where $\lambda : (S^1)^k \to \mathbb{C} \setminus \{0\}$ is a global scalar function. When both diagrams represent unitary circuits, $|\lambda(\vec{\alpha})| = 1$ automatically.

Phase folding and Clifford rewrites transform phase angles by adding input angles, changing their signs, and introducing constant Clifford phases. We therefore use integer-affine reductions to describe these transformations.

Definition 31 A parametrised diagram D_1 *affinely reduces* to D_2 when, for all $\vec{\alpha}$,

$$D_1[\vec{\alpha}] = \lambda(\vec{\alpha})D_2[M\vec{\alpha} + \vec{c}] \tag{83}$$

modulo 2π, for an integer matrix M, a constant vector $\vec{c}$, and a global scalar function λ.

An affine map can copy one input parameter into several output angles: α_j contributes to every row for which the jth column of M is non-zero. Phase folding combines parameter occurrences rather than duplicating them, so such copying is excluded.

Definition 32 An affine reduction is *parsimonious* if M has at most one non-zero entry in each column. Thus no input parameter is copied into more than one output location.

Definition 33 The **affine parsimonious parameter optimisation problem for Clifford circuits** is to find, for a given PCC C_1, a PCC C_2 with the fewest parameters obtainable from C_1 by an affine parsimonious reduction.

4.3 Optimality of phase teleportation

We prove that phase teleportation [20], reviewed in Section 3.6.3, minimises the parameter count under affine parsimonious reductions. Recall that phase teleportation applies full reduction to a symbolic ZX representation of the circuit and transfers the resulting phase combinations back to the original phase locations. Redundant locations are set to zero, while the non-phase gates and circuit connectivity remain unchanged.

4.3.1 *Phase teleportation and reduced gadget form*

Proposition 34 *Let $C[\vec{\alpha}]$ be a PCC with k uniquely occurring parameters. Applying full reduction to its symbolic ZX-diagram produces a diagram $D\left[\vec{\beta}\right]$ with l parameters such that:*

1. *D is in reduced gadget form; and*
2. *for some $P \in \{-1, 0, 1\}^{l\times k}$, $\vec{c} \in \left(\frac{\pi}{2}\mathbb{Z}\right)^l$, and global scalar function λ,*

$$C[\vec{\alpha}] = \lambda(\vec{\alpha})D[P\vec{\alpha} + \vec{c}], \tag{84}$$

where P has exactly one non-zero entry in each column.

Consequently, the corresponding phase-teleportation update is an affine parsimonious reduction from C to a circuit with l parameters.

Proof. For the first claim, the reduced-gadget-form criteria are exactly the exit conditions of full reduction. Since every successful step removes a spider or phase gadget, the procedure terminates and its output D is in reduced gadget form [20].

For the second claim, track the invariant that every input parameter occurs with coefficient $+1$ or -1 in at most one symbolic phase label, while every constant term is a Clifford multiple of $\frac{\pi}{2}$. This holds initially because the phase labels contain distinct input parameters. Local complementation and pivoting only change signs and add Clifford constants, while gadget absorption and fusion combine labels with disjoint parameter supports, since parameters are never copied.

The surviving labels therefore have the form

$$\vec{\beta} = P\vec{\alpha} + \vec{c}. \tag{85}$$

Here $P \in \{-1, 0, 1\}^{l\times k}$ has at most one non-zero entry in each column and $\vec{c} \in \left(\frac{\pi}{2}\mathbb{Z}\right)^l$. A zero column would make the right-hand side independent of the corresponding input parameter up to a scalar, contradicting Proposition 26. Thus every column has exactly one non-zero entry. Taking $\vec{\beta}$ to consist of the surviving symbolic labels also ensures that P has no zero rows.

Every graphical rewrite is sound, giving the stated equality. Phase teleportation transfers each surviving label back to one of its original phase locations and sets the other locations contributing to that label to zero. The resulting circuit therefore has l parameters, given by the rows of $P\vec{\alpha} + \vec{c}$, and is equivalent to $C[\vec{\alpha}]$ up to global phase. Since P has at most one non-zero entry in each column, this is an affine parsimonious reduction. □

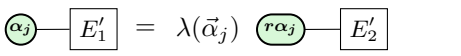
4.3.2 *Properties of parsimonious affine reductions*

A parsimonious affine reduction can neither discard a non-trivial input parameter nor multiply it by an integer other than $+1$ or -1.

> Lemma 35 *Let D_1 and D_2 have k and l parameters, suppose every parameter of D_1 is non-trivial, and let*
>
> $$D_1[\vec{\alpha}] = \lambda(\vec{\alpha})D_2[P\vec{\alpha} + \vec{c}] \tag{86}$$
>
> *be a parsimonious affine reduction. Then every column of P has a unique non-zero entry, and that entry is $+1$ or -1.*

Proof. Fix a column j of P. By parsimony, it is either zero or contains a unique non-zero entry. If column j were zero, then fixing the other input parameters to zero would leave the right-hand side independent of α_j. This is impossible because every parameter of D_1 is non-trivial. Therefore column j contains a unique non-zero entry $r = P_{ij}$.

Define $E_1[\theta] = D_1\left[\theta\vec{e}_j\right]$ and $E_2[\theta] = D_2[\vec{c} + \theta\vec{e}_i]$. Restricting the reduction to the jth input parameter gives $E_1[\theta] \propto E_2[r\theta]$, as represented by:



Since r is a non-zero integer, either $r = \pm 1$ or $|r| \geq 2$. Suppose $|r| \geq 2$. Then $\frac{2\pi}{r}$ and 0 are distinct modulo 2π, but periodicity gives

$$E_1\left[\frac{2\pi}{r}\right] \propto E_2[2\pi] = E_2[0] \propto E_1[0]. \tag{87}$$

With all other input parameters fixed to zero, this says that two distinct values of α_j give proportional diagrams. Thus α_j is trivial, contradicting the hypothesis. Therefore $r = \pm 1$. □

When every target parameter is used, Lemma 35 also supplies a parsimonious right inverse for the parameter map. Thus, to prove optimality, it suffices to show that the reduced diagram admits no further reduction in parameter count.

Corollary 36 *Let D_1 have non-trivial parameters and suppose it parsimoniously and affinely reduces to D_2 and D_3. Suppose that every parameter of D_2 is used by the reduction from D_1, and that D_2 admits no parsimonious affine reduction to fewer parameters. Then D_3 has at least as many parameters as D_2.*

Proof. Let the parameter counts be k_1, k_2, k_3, and write the two reductions as

$$D_1[\vec{\alpha}] \propto D_2[P_2\vec{\alpha} + \vec{c}_2], \quad D_1[\vec{\alpha}] \propto D_3[P_3\vec{\alpha} + \vec{c}_3]. \tag{88}$$

By Lemma 35, every column of P_2 has a unique entry ± 1, and the assumption that every target parameter is used says that every row is non-zero. For each standard basis vector $\vec{e}_i$ of the target parameter space, choose a signed standard basis vector $\vec{q}_i$ satisfying $P_2\vec{q}_i = \vec{e}_i$. Taking the $\vec{q}_i$ as columns defines an integer right inverse Q with $P_2Q = I$ and exactly one non-zero entry in each column.

Substitute $\vec{\alpha} = Q\left(\vec{\beta} - \vec{c}_2\right)$ into both reductions. Their right-hand sides are then proportional to the same instance of D_1, and $P_2Q = I$ gives

$$D_2\left[\vec{\beta}\right] \propto D_3\left[P_3Q\vec{\beta} + \vec{c}_3 - P_3Q\vec{c}_2\right]. \tag{89}$$

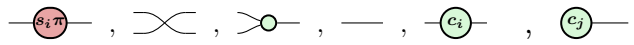
Every column of P_3Q is a signed column of P_3, so it contains at most one non-zero entry. Thus this is a parsimonious affine reduction from D_2 to D_3. Minimality of D_2 therefore implies $k_2 \leq k_3$. □

For the graphical optimality proof, we represent the parameter map as a linear map between the open parameter inputs of D_1' and D_2'. By Lemma 35, we can build this map from sign changes, phase fusions, and constant phase shifts.

Lemma 37 *Let D_1 and D_2 have k and l parameters, suppose all parameters of D_1 are non-trivial, and suppose there is a parsimonious affine reduction*

$$D_1[\vec{\alpha}] = \lambda(\vec{\alpha})D_2[P\vec{\alpha} + \vec{c}]. \tag{90}$$

There is a linear map $\hat{P} \in \mathbb{C}^{2^l \times 2^k}$ built from the graphical generators below, with $s_i \in \{0, 1\}$ and $c_j \in \mathbb{R}$,



such that:

1. for $\vec{\beta} = P\vec{\alpha} + \vec{c}$,

$$\hat{P} \otimes_{i=1}^{k} |+_{\alpha_i}\rangle \propto \otimes_{j=1}^{l} |+_{\beta_j}\rangle; \tag{91}$$

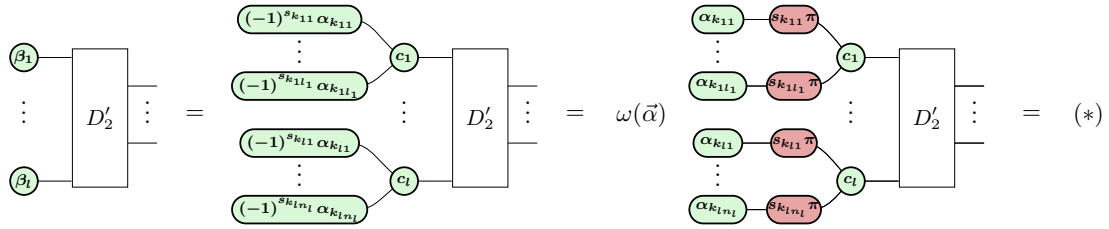
2. there is a constant non-zero scalar μ for which

$$D_1' = \mu D_2' \circ \hat{P}. \tag{92}$$

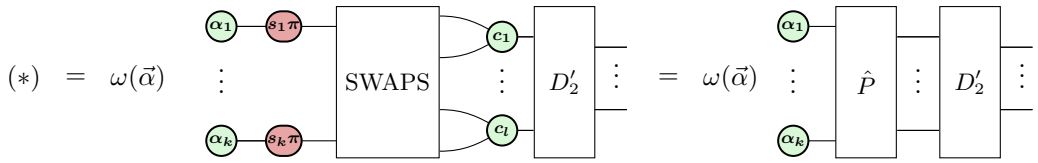
Proof. By Lemma 35, each input column of P contains exactly one entry ± 1. Let $s_i = 1$ when the entry in input column i is -1, and zero otherwise. For each output row j,

$$\beta_j = (-1)^{s_{k_{j1}}} \alpha_{k_{j1}} + \ldots + (-1)^{s_{k_{jn_j}}} \alpha_{k_{jn_j}} + c_j. \tag{93}$$

Here n_j may be zero, in which case the final generator displayed above prepares the constant output $|+_{c_j}\rangle$. The relation is implemented graphically by:



The sign-correction scalar $\omega(\vec{\alpha}) = \prod_{i=1}^{k} e^{is_i\alpha_i}$ arises from $X|+_\alpha\rangle = e^{i\alpha}|+_{-\alpha}\rangle$. Every input parameter occurs once, so swaps put the map into the form:



This defines $\hat{P}$ as a composite of the stated generators. Hence

$$D_1[\vec{\alpha}] = \lambda(\vec{\alpha})\omega(\vec{\alpha})\big(D_2 \circ \hat{P}\big)[\vec{\alpha}]. \tag{94}$$

Applying Proposition 27 to the non-trivial parameters of D_1 makes the product $\lambda(\vec{\alpha})\omega(\vec{\alpha})$ a constant μ, giving $D_1' = \mu D_2' \circ \hat{P}$. □

4.3.3 *Minimality of reduced gadget form*

Reducing the diagram produced by phase teleportation to fewer parameters would require the graphical parameter map to fuse at least two parameter inputs. Such a fusion induces a signed weight-two Z_iZ_j stabiliser of the underlying Clifford diagram. The following lemma, due to Aleks Kissinger, classifies these stabilisers in GSLC form.

> Lemma 38 *Let $|\psi\rangle = (C_1 \otimes \ldots \otimes C_n)|G\rangle$ be a graph state followed by local Cliffords. For distinct qubits u, v with $C_u, C_v \in \{I, H\}$, suppose either Z_uZ_v or $-Z_uZ_v$ stabilises $|\psi\rangle$. Only the positive sign is possible, and either:*
>
> 1. *u and v are adjacent in G, and*
>   - *$C_u \otimes C_v = H \otimes I$ with $\mathrm{nhd}(u) = \{v\}$; or*
>   - *$C_u \otimes C_v = I \otimes H$ with $\mathrm{nhd}(v) = \{u\}$; or*
> 2. *u and v are not adjacent, have identical neighbourhoods, and $C_u \otimes C_v = H \otimes H$.*

Proof. Conjugating by the local Clifford layer gives a signed weight-two stabiliser of $|G\rangle$, supported on u and v. By Equation 63, graph-state stabilisers are generated by

$$P_w = X_w \prod_{w' \in \mathrm{nhd}(w)} Z_{w'}.$$

A product of k distinct generators has weight at least k, because every selected vertex remains in its support. A weight-two stabiliser is therefore a product of one or two generators.

If it is a single generator, it is P_u or P_v. In the first case, support only on u, v forces $\mathrm{nhd}(u) = \{v\}$ and $P_u = X_u Z_v$; conjugation gives $Z_u Z_v$ precisely when $C_u \otimes C_v = H \otimes I$. The case $P_v = Z_u X_v$ is symmetric, with $\mathrm{nhd}(v) = \{u\}$ and $C_u \otimes C_v = I \otimes H$. This gives case 1.

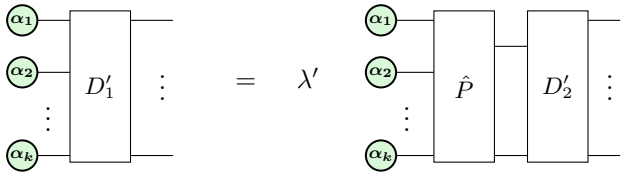
If it is a product of two generators, those generators must be P_u and P_v. Absence of support away from u, v requires $\mathrm{nhd}(u) \setminus \{v\} = \mathrm{nhd}(v) \setminus \{u\}$. When u and v are adjacent, $P_u P_v = Y_u Y_v$, which no local Clifford in $\{I, H\}$ maps to $Z_u Z_v$. Hence they are non-adjacent, have identical neighbourhoods, and $P_u P_v = X_u X_v$, giving case 2 under $H \otimes H$.

The stabilisers $X_u Z_v$, $Z_u X_v$, and $X_u X_v$ obtained above all have positive sign, and conjugation by I and H introduces no sign in these cases. Thus $-Z_u Z_v$ cannot stabilise $|\psi\rangle$. □

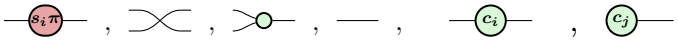
Theorem 39 *Let C_1 be a PCC, and let C_2 be the circuit obtained by phase teleportation. If C_3 is any parametrised diagram to which C_1 affinely and parsimoniously reduces, then C_3 has at least as many parameters as C_2. Thus the algorithm of [20] is optimal within affine parsimonious reductions.*

Proof. Start with a PCC $C\left[\vec{\theta}\right]$. Its input parameters are non-trivial by Proposition 26. Let $D_1[\vec{\alpha}]$ be the reduced diagram supplied by Proposition 34. Every output parameter α_j is also non-trivial. Each output angle contains an input parameter with coefficient $+1$ or -1. Since inputs are not shared between output angles and the offsets are Clifford, any Clifford assignment of the other output parameters can be realised by a Clifford assignment of the circuit inputs. If α_j were trivial for such an assignment, varying an input parameter contributing to it would contradict Proposition 26. Extraction preserves the parameter count, so by Corollary 36 it suffices to prove that D_1 is minimal.

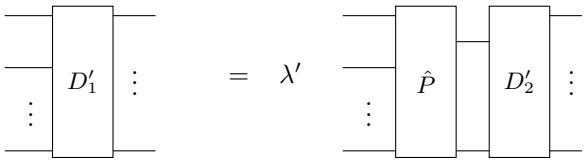
Suppose instead that $D_1[\vec{\alpha}] = \lambda(\vec{\alpha}) D_2[P\vec{\alpha} + \vec{c}]$ and D_2 has fewer parameters. By Lemma 37:



Here the displayed scalar is constant and $\hat{P}$ is built from:



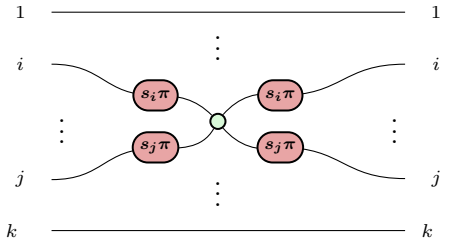
Since the equality holds for every $\vec{\alpha}$, it gives the state equality:



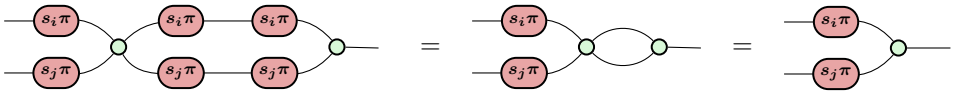
(95)

The construction in Lemma 37 attaches each of the parameter inputs of D_1 to exactly one output row. Since D_2 has fewer parameters, two inputs must be attached to the same row, so $\hat{P}$ contains a Z-fusion:

Let α_i and α_j be two fused parameters, and define the projector Π acting on their corresponding parameter inputs:



Spider fusion shows $\hat{P} \circ \Pi = \hat{P}$:



By Equation 95, $D_1' \circ \Pi = D_1'$. Thus Π stabilises D_1', and so does the signed Pauli $(-1)^{s_i+s_j} Z_i Z_j$.

The diagram D_1' is in GSLC form, and the two parameter sites have local Clifford I or H. If $s_i + s_j$ is odd, Lemma 38 rules out the resulting negative signed stabiliser immediately. If it is even, that lemma leaves two possibilities. In its first case, the parameter vertex carrying the Hadamard has exactly one target, contrary to reduced gadget form (Definition 17). In its second case, the two parameter vertices are phase gadgets with identical target sets, which reduced gadget form also excludes. Both cases are contradictions.

Hence no smaller D_2 exists. The reduced diagram is minimal and therefore optimal by Corollary 36; circuit extraction preserves that count. □

The gadgets in Figure 12 illustrate how different target sets can represent the same parity. Here $\oplus$ denotes XOR: it is 1 when its two bits differ and 0 when they agree. In particular, $x \oplus x = 0$, so two occurrences of the same bit cancel.

Propagating the Hadamards through the initial circuit exposes a shared CNOT network in which the parities change between rotations. The red–green intermediate diagram separates the phase gadgets from the remaining Clifford computation. Its leftmost red spider on the second row combines x_1 and x_2, so the green vertex to its right carries $x_1 \oplus x_2$. The next red spider combines this bit with x_1 again,

giving $(x_1 \oplus x_2) \oplus x_1 = x_2$. All three output bits equal their input values. After this propagation, the two fixed $\frac{\pi}{2}$ phases act on $x_1 \oplus x_3$ and its complement on different wires, so together they contribute only a global phase i.

With a zero-phase body, a phase gadget acts on the XOR of the bits at its targets. The α_1 gadget targets the computed bit $x_1 \oplus x_2$ and the third input bit x_3, whereas α_2 targets the three output bits individually. Both therefore act on $x_1 \oplus x_2 \oplus x_3$. The β_1 gadget targets x_1, $x_1 \oplus x_2$, and x_3. Cancelling the two occurrences of x_1 leaves $x_2 \oplus x_3$, but the π phase on its body complements this bit, giving $1 \oplus x_2 \oplus x_3$. Removing this Pauli phase negates β_1, up to a global phase $e^{i\beta_1}$. The β_2 gadget acts directly on $x_2 \oplus x_3$, so the pair fuses to $\beta_2 - \beta_1$.

Colour change turns the red spiders green and their incident edges into Hadamard edges. The resulting graph is annotated with the computed bit and gadget parities. Removing the internal computation and accounting for the Pauli phase exposes the matching targets for gadget fusion.

The circuit-side description in Section 2.3.5 reaches the same result by transporting rotations through Clifford gates. Here the fixed Clifford part is the identity up to global phase, and the four rotations can be written about two distinct Pauli axes:

$$Q_1 = Z_1 Z_2 Z_3, \quad Q_2 = Z_2 Z_3.$$

Since these axes commute, both routes give $\alpha_1 + \alpha_2$ and $\beta_2 - \beta_1$. The final gadgets have distinct target sets, and neither is single-target, so Theorem 39 certifies the minimum of two parameters within affine parsimonious reductions.

> Remark 40 We only needed the circuit assumption to establish that the input parameters are non-trivial. The argument also works for parametrised Clifford diagrams whose parameters each occur once and are non-trivial. Since the target diagram need not be unitary, post-selection cannot reduce the parameter count further under affine parsimonious reductions.

4.4 Consequences and related algorithms

4.4.1 *PyZX and TOpt*

Recall that PyZX's phase teleportation [20] rewrites ZX-diagrams, whereas TOpt [40] commutes and merges Pauli rotations. Both were introduced for T-count reduction, but also apply to arbitrary or symbolic phase angles.

Simmons [39] relates the two approaches by using Pauli flow to extract a Pauli Dependency DAG from a measurement pattern or graph-like ZX-diagram, using the representation introduced in Section 2.3.5. In this representation, PyZX's graph rewrites can be simulated by commuting Clifford-valued Pauli rotations through the PDDAG and merging compatible gates.

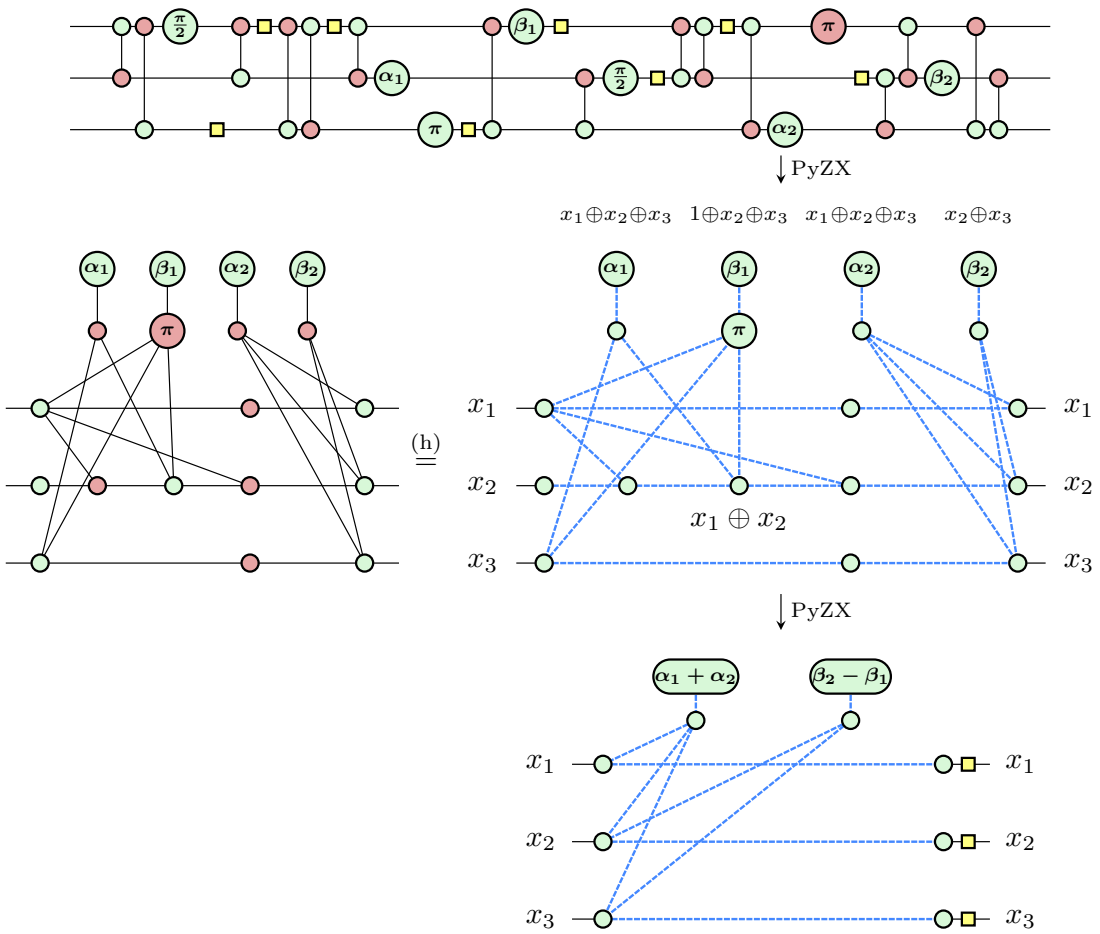


Figure 12: Simplification of a parametrised Clifford circuit. The red–green diagram reveals the COPY/XOR structure of the intermediate diagram, as seen in Section 3.2. Removing the internal spiders exposes equal parities, allowing the corresponding phase gadgets to fuse. The π body phase acts as a NOT, swapping the even- and odd-parity cases and thus negating β_1 up to global phase. This example illustrates why `full_reduce` is optimal under affine parsimonious reductions (Theorem 39): no internal spiders remain for the gadgets to attach to, and all phase gadgets with the same target set have fused.

> Corollary 41 *Let C be a parametrised Clifford circuit with uniquely occurring parameters. Both phase teleportation and the symbolic extension of TOpt obtain the minimum parameter count within affine parsimonious reductions.*

Proof. Phase teleportation is optimal by Theorem 39. Simmons shows that its graph-theoretic rewrites can be simulated in the PDDAG representation by commuting and merging Pauli rotations [39]. Vandaele, Perdrix, and Vuillot subsequently prove directly that this circuit-side phase-merging procedure is optimal for parametrised Clifford circuits with non-repeated parameters [87]. Hence the two algorithms attain the same optimum in the black-box parameter model. □

For a Clifford+T circuit, this correspondence explains the identical counts in Figure 11. When each T gate is treated only as a black-box phase gate, neither algorithm can improve upon the other by further phase folding. Reducing the T-count

further requires using the fixed angle $\frac{\pi}{4}$, for example through fixed-angle identities or the phase-polynomial and Reed–Muller methods reviewed in Section 2.3.4.

4.4.2 *Measurement-based quantum computation*

In the one-way model of Raussendorf and Briegel [88], a highly entangled cluster state is prepared first, and computation proceeds through adaptive single-qubit measurements. For the standard cluster-state family, the resource preparation is fixed apart from the required size, while the measurement bases encode the program and may depend on earlier outcomes [60], [61]. Measuring the qubits consumes the resource, which gives the model its name [89].

Following the terminology of Backens et al. [90], we represent measurement patterns in the one-way model by *labelled open graphs*. A labelled open graph is a tuple $\Gamma = (G, I, O, \lambda)$: the graph $G = (V, E)$ generalises the cluster lattice, I and O designate its inputs and outputs, and $\lambda_v \in \{XY, XZ, YZ\}$ specifies the measurement plane of each non-output vertex. These three measurement planes are those used in the generalised-flow formulation of Browne et al. [91]. In the ZX-diagram, the resource state has the graph-state representation described in Section 3.5.2. Figure 13 shows this representation together with the three measurement effects.

Once the resource state is prepared, computation proceeds by measuring the non-output qubits. Each measurement returns an outcome bit. If the outcome is 0, no correction is needed; if it is 1, the measurement introduces a Pauli error. A gflow specifies how this error is propagated to later vertices: it is absorbed by adjusting later measurement bases through classical feed-forward, while any correction reaching an output can be applied or recorded classically [91]. In the ZX-calculus, this propagation appears as Pauli phases moving through the diagram, as illustrated in Figure 14 [90].

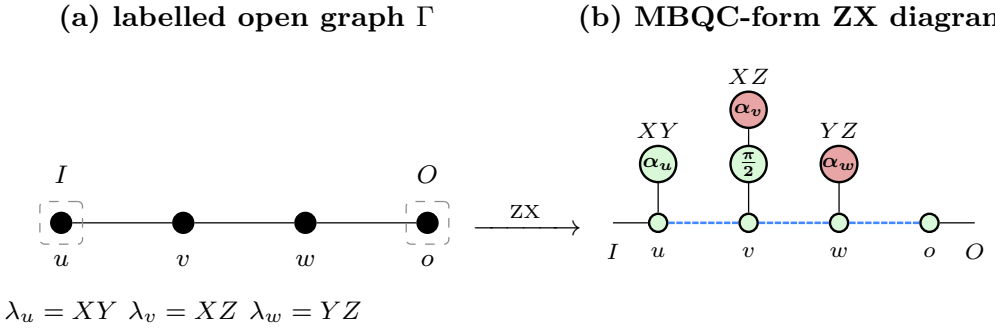


Figure 13: A labelled open graph and its MBQC-form ZX translation. Vertices and edges become zero-phase Z spiders and Hadamard edges, while I and O become boundary wires. The effects show XY, XZ, and YZ measurements. Adapted from [61], Eqs. (9.5)–(9.6), and [90], Table 1.

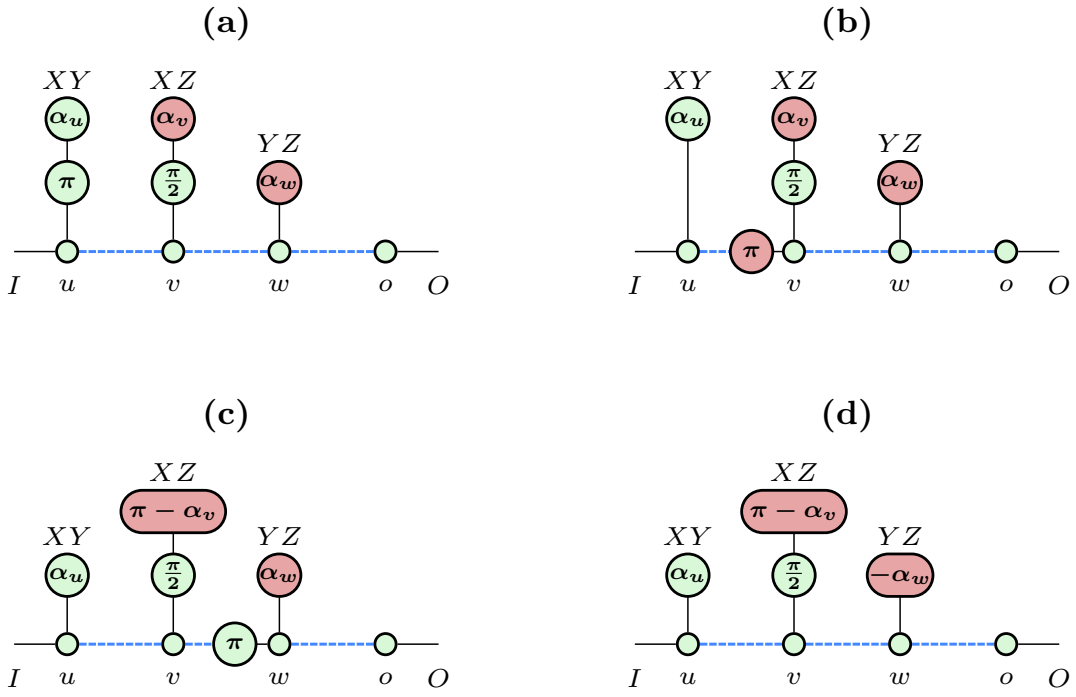


Figure 14: Propagation of an outcome-dependent correction in the same MBQC-form ZX-diagram, adapted from Backens et al. [90]. (a) A π error occurs at the measured XY vertex u. (b) It propagates as an X correction to v. (c) Absorbing this correction changes the XZ angle to $\pi - \alpha_v$ and sends a Z correction to w. (d) Absorbing the Z correction changes the YZ angle to $-\alpha_w$.

In MBQC, each parametrised spider represents a measurement angle at a non-output vertex. Reducing the number of these spiders therefore reduces the number of parametrised measurements. We obtain the following optimality result:

> Theorem 42 *Let $\mathcal{D}$ be a measurement pattern with gflow, equally many inputs and outputs, and a distinct parameter on each parametrised measurement. The optimised pattern $\mathcal{D}'$ produced by the algorithm of [90] has the minimum number of parametrised measurements among patterns with gflow to which $\mathcal{D}$ affinely and parsimoniously reduces.*

Proof. Translate $\mathcal{D}$ to a parametrised Clifford ZX-diagram. Since $\mathcal{D}$ has gflow and equally many inputs and outputs, the extraction procedure of [90] turns it into a unitary circuit with one uniquely occurring $R_Z(\alpha)$ gate for each parametrised measurement. These parameters are non-trivial by Proposition 26. The same argument applies to every competing pattern with gflow, so Theorem 39 gives the optimal parameter count for the simplified ZX-diagram. The measurement-pattern algorithm of [90] uses the corresponding rewrites, and therefore produces a pattern with the same optimal count. □

4.5 Hardness under relaxed assumptions

Tuomas Laakkonen showed that parameter optimisation is NP-hard for circuits containing Clifford and Toffoli gates, using a multi-parameter reduction from SAT. For the two hardness results below, we replace each Toffoli gate with either an exact Clifford+T implementation or a post-selected implementation using repeated parametrised phase gates.

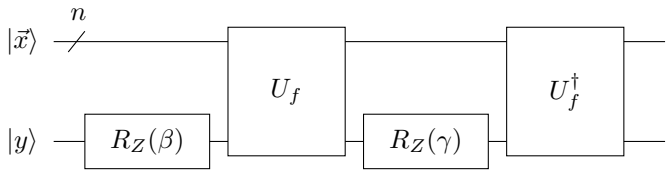
4.5.1 *Toffoli-based hardness*

Proposition 43 *Computing the optimal parameter count for circuits containing Clifford and Toffoli gates is NP-hard.*

Proof. The proof reduces SAT to parameter optimisation. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean formula, and construct a polynomial-size Clifford+Toffoli circuit U_f satisfying

$$U_f|\vec{x}, y\rangle = |\vec{x}, y \oplus f(\vec{x})\rangle. \tag{96}$$

Introduce two parameters β and γ and construct the circuit $C_f[\beta, \gamma]$:



After discarding a global phase, its action on computational-basis states is

$$|\vec{x}, y\rangle \mapsto e^{i\beta y + i\gamma(f(\vec{x}) \oplus y)}|\vec{x}, y\rangle. \tag{97}$$

If f is unsatisfiable, then $f(\vec{x}) = 0$ for every input, and the two parametrised phase gates fuse. If f is always true, then $f(\vec{x}) = 1$ for every input, and the phase gates again reduce to a single parameter after commuting through the X action on the final qubit. In both cases the optimal parameter count is one.

Otherwise, there are inputs on which f takes both values. The parameters β and γ then control two independent relative phases, so neither can be eliminated or fused with the other. The optimal parameter count is therefore two.

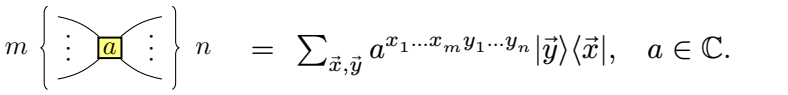
An algorithm for optimal parameter count can now decide SAT. A count of two shows that f is satisfiable. If the count is one, then f is either unsatisfiable or always true, and evaluating $f(0...0)$ distinguishes these cases. This gives a polynomial-time reduction from SAT. The construction is adapted from [10]. □

Remark 44 The construction also shows that deciding whether a Clifford+Toffoli circuit reduces to one parameter is coNP-hard. Given a Boolean formula f, define $g(\vec{x}, z) = f(\vec{x})z$. If f is unsatisfiable, then g is constantly zero and the optimal parameter count of C_g is one. If f is satisfiable, choose $\vec{x}$ with $f(\vec{x}) = 1$. Then $g(\vec{x}, 0) = 0$ and $g(\vec{x}, 1) = 1$, so g is nonconstant and the optimal count is two. This gives a polynomial-time reduction from UNSAT.

4.5.2 *ZH notation for multi-controlled gates*

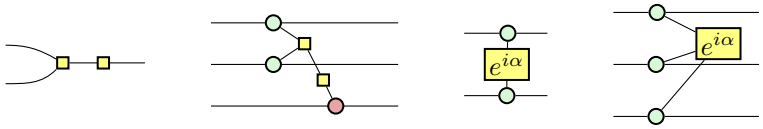
To represent the reversible circuits used in the SAT reductions, we use the ZH-calculus. Its H-boxes give compact diagrams for AND, Toffoli, and multi-controlled phase gates. We introduce the notation and rules used in the proofs below. See [92] and [93] for fuller treatments.

An *H-box* labelled a, with m inputs and n outputs, denotes:



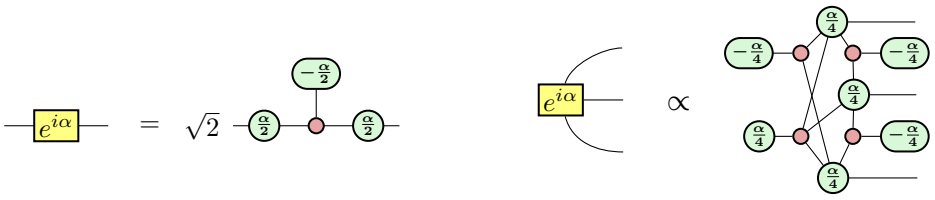
Every matrix entry is 1 except the bottom-right entry, which is a and corresponds to all incident bits being 1. A one-input, one-output H-box with $a = -1$ is a rescaled Hadamard, so the label is omitted in that case.

Working up to a non-zero scalar, H-boxes give compact representations of the AND operation $\mathrm{AND}|x, y\rangle = |xy\rangle$ and Toffoli gates. An H-box labelled $e^{i\alpha}$ also represents controlled and doubly controlled phase gates, using the convention in Equation 11.



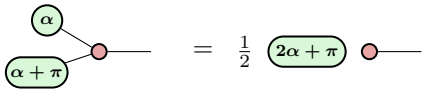
Two standard ZH rewrites are used: *H-box fusion* (HF) and the *graphical Fourier transform* (FT) [94]. The latter expresses an n-ary H-box labelled $e^{i\alpha}$ using $2^n - 1$ phase gadgets with phases $\pm\frac{\alpha}{2^{n-1}}$.

(HF)



(FT)

The ZX rule *supplementarity* (SUP) is also used. It merges two spiders with the same neighbourhood whose phases differ by π. For arbitrary phases, supplementarity is not derivable from the Clifford ZX rules alone [95], so it is stated explicitly.

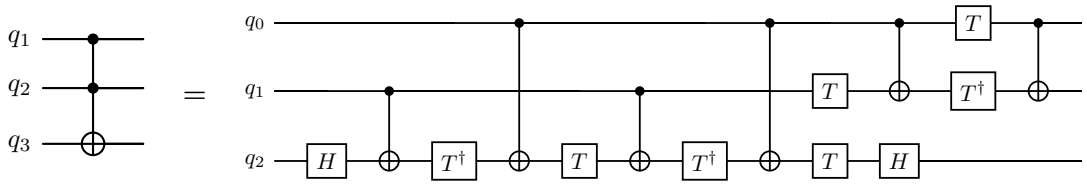


(SUP)

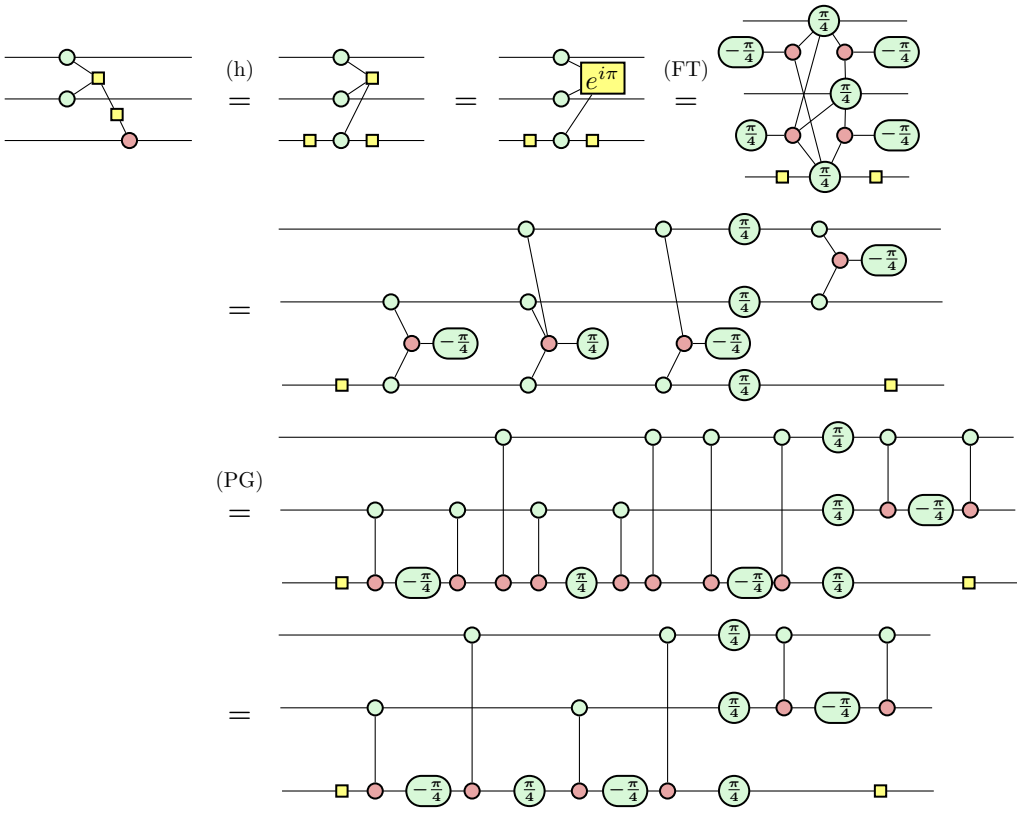
4.5.3 *Non-Clifford gates*

Allowing fixed T gates between the parametrised rotations makes parameter optimisation NP-hard. We prove this by replacing the Toffoli gates in Proposition 43 with Clifford+T circuits.

Lemma 45 *The following Clifford+T circuit implements a Toffoli gate:*



Proof.



The final ZX diagram translates directly to the displayed Clifford+T circuit. □

Proposition 46 *Parameter optimisation for circuits containing Clifford+T gates is NP-hard.*

Proof. Replace every Toffoli gate in the reduction of Proposition 43 by the Clifford+T circuit of Lemma 45. This incurs only constant overhead per Toffoli, so the reduction remains polynomial. □

4.5.4 *Repeated parameters*

A PCC allows each parameter to occur on at most one gate, but standard gate decompositions can introduce repeated parameters. For example, decomposing a controlled phase gate $CR_Z(\alpha)$ produces several phase gates whose angles depend on α. Allowing such repetitions makes parameter optimisation NP-hard for post-selected circuits.

Allowing arbitrary functions of a repeated parameter would make hardness immediate: $R_Z(\alpha)$ and $R_Z(-\alpha + \frac{\pi}{4})$ fuse to $R_Z(\frac{\pi}{4}) = T$. The stricter condition imposed here is that every gate angle is an integer multiple of the same repeated parameter. Such a multiple can be expanded into repeated copies of that parameter using phase fusion in reverse. Even under this restriction, parameter optimisation is NP-hard for post-selected circuits.

The selected outcome can have zero amplitude for some parameter values, so we relax Definition 30 to allow the scalar factor to vanish.

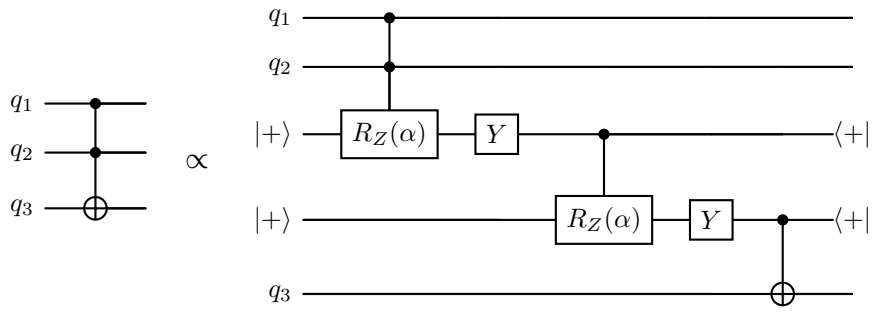
Definition 47 A parametrised post-selected circuit C_1 with parameter space $(S^1)^k$ *reduces* to a post-selected circuit C_2 with parameter space $(S^1)^l$ when there is a function $f : (S^1)^k \to (S^1)^l$ such that

$$C_1[\vec{\alpha}] = \lambda(\vec{\alpha}) C_2[f(\vec{\alpha})] \tag{101}$$

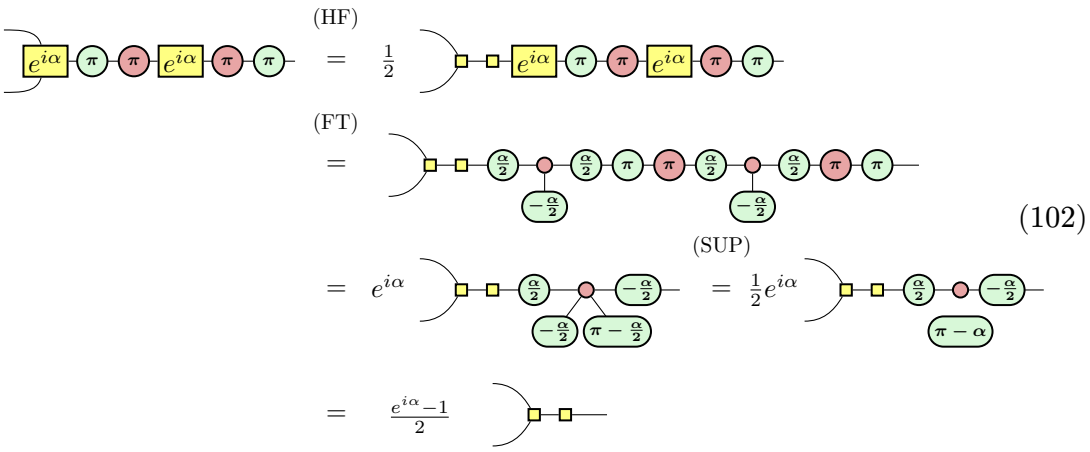
for every $\vec{\alpha}$, where $\lambda : (S^1)^k \to \mathbb{C}$ may take the value zero. The reduction is *affine* when f is affine in the sense of Definition 31.

We use the following post-selected circuit to replace each Toffoli gate in the SAT reduction.

Lemma 48 *Whenever $\alpha \neq 0 \bmod 2\pi$, the following post-selected circuit implements a Toffoli gate up to a non-zero scalar:*

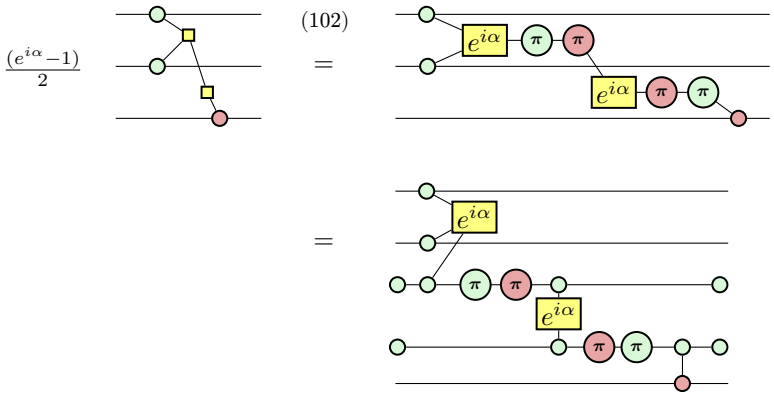


Proof.



(102)

The resulting scalar is $\frac{e^{i\alpha}-1}{2}$, which is non-zero precisely when $\alpha \neq 0 \bmod 2\pi$. Substituting this identity into the Toffoli diagram gives the following post-selected construction:



(103)

The two H-boxes attached to Z-spiders represent $CR_Z(\alpha)$ and $CCR_Z(\alpha)$. Decomposing them gives the circuit displayed above, in which several phase-gate angles depend on the same parameter α. □

Proposition 49 *Parameter optimisation for post-selected Clifford circuits with repeated parametrised phase gates is NP-hard.*

Proof. First specialise the construction of Lemma 48 to the repeated-parameter gate set. The H-boxes representing $CR_Z(\alpha)$ and $CCR_Z(\alpha)$ have Fourier decompositions containing $2^2 - 1 = 3$ and $2^3 - 1 = 7$ parameter-dependent phase gates, respectively.

The two Fourier decompositions introduce phases $\pm\frac{\alpha}{2}$ and $\pm\frac{\alpha}{4}$. To make every gate angle an integer multiple of a single repeated parameter θ, set $\alpha = 4\theta$. This clears the denominators, leaving only phases $\pm 2\theta$ and $\pm\theta$. A phase $\pm 2\theta$ can be split into two phases $\pm\theta$, and Clifford conjugation changes the sign of a Z phase up to a global phase. Hence every parametrised gate can be taken to be a repeated copy of $R_Z(\theta)$, with only Clifford gates between the copies.

Under this substitution, the post-selected scalar is $\frac{e^{4i\theta}-1}{2}$. It vanishes exactly when $\theta \in (\frac{\pi}{2})\mathbb{Z}$. At these Clifford values the selected branch represents the zero map, as

permitted by Definition 47. For every other value, the circuit implements a Toffoli up to a non-zero scalar.

Apply the reduction of Proposition 43, replacing every Toffoli gate by the circuit of Lemma 48 and using the same parameter θ throughout. If the reversible oracle contains m Toffoli gates, the replacement contributes only a scalar proportional to $\left(e^{4i\theta} - 1\right)^m$. This factor may vanish under Definition 47, and an affine post-selected reduction can discard θ.

The remaining parameters β and γ behave exactly as in Proposition 43. They fuse when f is constant and control independent relative phases when f is non-constant, giving optimal parameter counts of one and at least two, respectively. Evaluating $f(0...0)$ distinguishes the always-true case from the unsatisfiable case, so SAT reduces in polynomial time. □

With uniquely occurring parameters, post-selection still allows efficient parameter optimisation under affine parsimonious reductions (Remark 40). Allowing repeated parameters makes the post-selected problem NP-hard (Proposition 49). Whether the same holds for ordinary unitary circuits remains open.

4.6 Discussion

This chapter shows that phase teleportation efficiently minimises the parameter count of PCCs with uniquely occurring parameters, within the class of affine parsimonious reductions. PyZX and TOpt reach the same phase-folding optimum [39, 87], explaining their matching benchmark T-counts. These counts need not be globally optimal, since phase folding does not use identities specific to the T angle. Theorem 42 extends the parameter result to unitary measurement patterns with gflow.

Allowing fixed T gates among the Clifford gates gives NP-hardness by Proposition 46, while repeated parameters give NP-hardness for post-selected circuits by Proposition 49. The complexity of repeated parameters in ordinary unitary circuits remains open.

The proof assumes parsimony. A reduction that copies each input parameter to at most three output locations can be replaced by a parsimonious one [1], but the general non-parsimonious case remains open.

The next chapter asks whether a Transformer can learn to predict the output of PyZX's `full_reduce` from examples alone.

Chapter V

Learning ZX Rewriting with Transformers

The development of PyZX, together with advances in ZX algorithms, has enabled automated rewriting of diagrams containing hundreds of thousands of vertices, with built-in strategies that reduce Clifford diagrams to a pseudo-normal form and perform effective T-count optimisation [18], [19], [20]. As established in Chapter 4, PyZX's phase-teleportation procedure minimises parameter count for parametrised Clifford circuits under affine parsimonious reductions, provided that each parameter occurs once (Theorem 39).

Despite this scalability, such procedures provide only specialised forms of automated reasoning. They apply a predetermined strategy towards a particular reduced form, whereas practitioners may wish to obtain an equivalent diagram that exposes a useful property or minimises a chosen quantity, such as the number of vertices. In the longer term, our aim is to develop a general-purpose rewriting system, possibly with the aid of machine learning, that can adapt its strategy to each of these diagrammatic goals.

When this project began, there were no published applications of machine learning to ZX-diagram rewriting. This work is therefore among the earliest attempts to use neural networks for ZX rewriting.[1] As an initial proof of concept, we ask whether a Transformer can predict the result of PyZX's `full_reduce` procedure directly from an initial unsimplified ZX diagram.

We begin with a deliberately simple formulation of graph simplification as *sequence-to-sequence translation*. We use PyZX to generate pairs consisting of an initial graph and its `full_reduce` output, serialise both graphs as token sequences, and train an encoder–decoder Transformer to translate the first sequence into the second. The learner receives neither the ZX rules nor an implementation of the reduction strategy, only examples of initial graphs and their corresponding reduced outputs. At evaluation time, we use PyZX to check whether each generated graph

[1] At the time, neural networks had already been applied to quantum-circuit optimisation and compilation [96, 97], and ZX diagrams had been used to analyse quantum neural-network ansatz families [98, 99], but neither line of work trained a model to rewrite ZX diagrams.

is well formed, equivalent to the input, and no larger than the corresponding PyZX output.

Chapter Contributions

- **Sequence-to-sequence ZX rewriting.** We formulate learning PyZX's `full_reduce` procedure as a sequence-to-sequence task and symbolically verify each generated graph. On held-out ten-qubit Clifford circuits of depth 15, 96.4% of predictions are valid, equivalent to the input, and no larger than the corresponding `full_reduce` output. We obtain similar results for CNOT and Clifford+T circuits, but performance declines on circuits deeper than those used for training [2].

5.1 Sequence-to-sequence translation

5.1.1 *Conditional sequence generation*

Sequence-to-sequence learning models a mapping between variable-length input and output sequences. A recurrent neural network reads a sequence in order, updating an internal vector after each item. Sutskever et al. [100] introduced an encoder–decoder architecture in which one such network encodes the source sequence and another generates the target autoregressively, one token at a time, conditioned on the tokens already produced. Bahdanau et al. [101] subsequently augmented neural machine translation with attention, allowing each decoding step to consult the relevant source representations rather than relying on a single encoded vector. Vaswani et al. [102] introduced the Transformer, an attention-based encoder–decoder that dispenses with recurrence. During training, sequence positions can be processed in parallel; during generation, outputs are produced one token at a time.

The standard application of sequence-to-sequence learning is machine translation. Each sentence is first divided into discrete units called tokens, such as words or parts of words. The model encodes the input sequence of tokens and generates the target sequence one token at a time. Figure 15 illustrates this process for English-to-French translation.

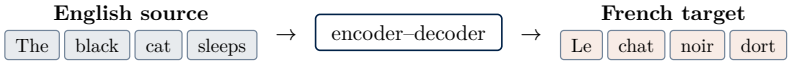


Figure 15: Sequence-to-sequence translation. Each target token is generated from the encoded source and the target tokens already produced.

5.1.2 *Mathematical transformations as sequences*

Sequence-to-sequence learning is not restricted to natural language. In principle, it can be used to learn any transformation whose inputs and outputs can be represented as sequences. A particularly relevant application is learning transformations between mathematical expressions, such as symbolic integration. Lample and Char-

ton [103] represented each expression as a syntax tree, with operators and functions as internal nodes and variables, constants, or numbers as leaves. They linearised the tree in prefix notation by placing each node before its children. Figure 16 illustrates this conversion for $\sin(x) + x\cos(x)$. In prefix notation, each operator precedes its operands, thereby capturing the expression's bracketing without parentheses.

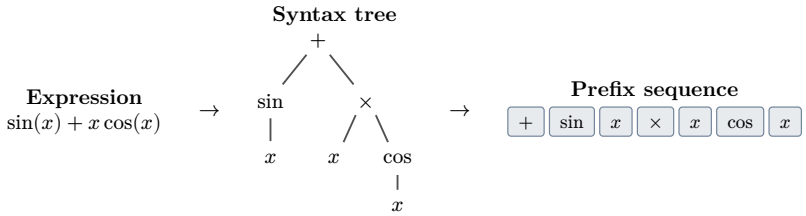


Figure 16: Linearising a mathematical expression. A preorder traversal visits each operator before its children, converting the expression tree for $\sin(x) + x\cos(x)$ into the prefix token sequence $[+, \sin, x, \times, x, \cos, x]$.

In general, differentiation is straightforward, whereas finding an antiderivative is difficult. Lample and Charton exploited this asymmetry through *backward generation*: they sampled an expression F, computed $f = F'$, and trained a Transformer on pairs (f, F) to recover the antiderivative. This produces training examples without first having to solve the integration problems. Figure 17 illustrates the resulting prediction task and the reverse direction used to construct its training pairs.

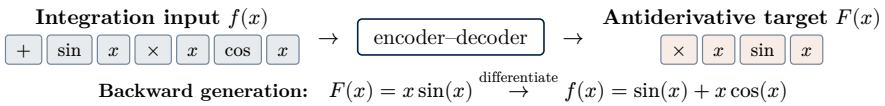


Figure 17: Sequence-to-sequence symbolic integration. Backward generation samples an antiderivative F and differentiates it to obtain the input f; the model learns the reverse mapping from f to F.

5.2 ZX rewriting as sequence-to-sequence translation

We now turn to applying sequence-to-sequence learning to ZX diagrams. Unlike symbolic integration, where each expression is represented by a tree, the underlying object here is a graph with no preferred vertex ordering. Nevertheless, representing diagrams as sequences provides a simple way to test whether a model can learn PyZX's reductions without explicitly incorporating graph structure or ZX semantics into its architecture.

5.2.1 *Learning goal and examples*

Given an initial ZX diagram D, the learning goal is to predict the diagram $F(D)$ produced by PyZX's `full_reduce` routine. Its reduction strategy and reduced gadget form are described in Section 3.6.3.

To construct an example, we convert a circuit to a ZX diagram in green-Hadamard form and apply `full_reduce` to a copy. The model receives only the pair $(D, F(D))$, represented as token sequences, with no ZX rules or intermediate rewrites. Figure 18 illustrates one such pair.

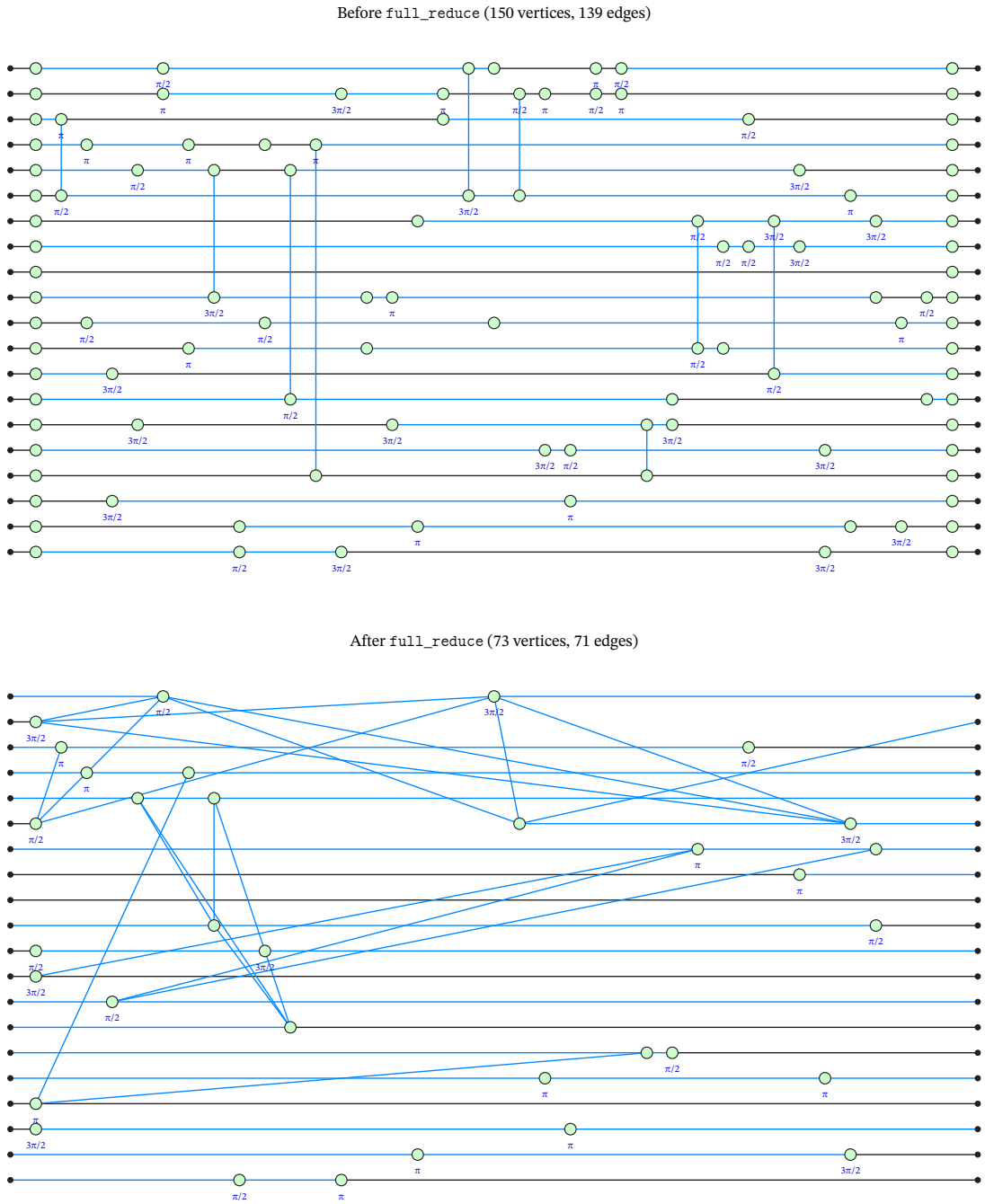


Figure 18: A generated twenty-qubit, depth-35 Clifford example before and after `full_reduce`: 150 vertices and 139 edges are reduced to 73 vertices and 71 edges. Only the initial and final graphs are shown, not the intermediate rewrites. Reproduced from [2].

5.2.2 *Serialising ZX graphs*

To represent ZX graphs as inputs and outputs of a sequence-to-sequence model, we define a serialisation S that converts each graph into a flat token sequence. The first two tokens give the numbers of vertices, $|V|$, and edges, $|E|$. We then order the vertices, assign them the identifiers `N0`, `N1`, and so on, and record the type and phase of each. Finally, each edge contributes its type and two endpoint identifiers. The resulting sequence has length $2 + 2|V| + 3|E|$.

All graphs are first converted to green-Hadamard form. Following PyZX's `VertexType` enumeration, 0 denotes a boundary and 1 a Z-spider; X-spiders therefore do not occur. Every phase in the Clifford+T fragment is an integer multiple of $\frac{\pi}{4}$, so the phase $k\frac{\pi}{4}$ is represented by the integer token k. Likewise, PyZX's `EdgeType` enumeration assigns 1 and 2 to ordinary and Hadamard edges, respectively. The rows in the figure are only a visual aid: the model receives one flat sequence.

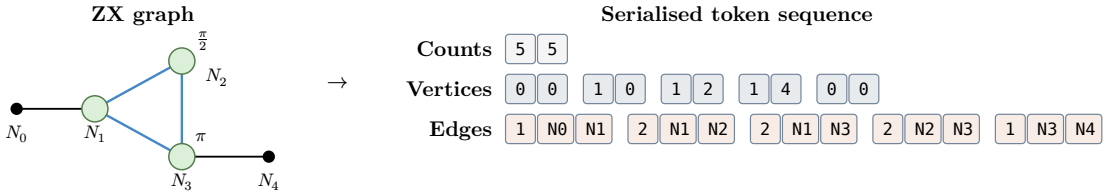


Figure 19: Serialising a ZX graph. Reading the records row by row gives the vertex and edge counts, the type and phase of each ordered vertex, and the type and endpoint identifiers of each edge.

Figure 19 illustrates the three parts of this representation. The implementation is given in Section 5.5.2.

With this serialisation, the induced transformation T on sequences is

$$T(S(D)) = S(F(D)). \tag{104}$$

Writing $x = S(D)$ and $y = S(F(D))$, each supervised example is the pair

$$(x, y) = (S(D), S(F(D))). \tag{105}$$

This serialisation is not unique: vertex names and edge ordering affect the token sequence, and edge records may occur far from the records of their endpoints. The model must learn to accommodate these arbitrary choices from data; it is not permutation invariant by construction.

5.3 Symbolically verified evaluation

Because an input ZX diagram can have several equivalent reduced forms, there may be several correct output diagrams. Moreover, the same output diagram can give several token sequences through different vertex names and edge orderings. The evaluation must therefore verify the diagram represented by the generated sequence, rather than rely on exact sequence matching.

Given an unseen input diagram D, the model generates a sequence that we attempt to parse into a candidate diagram $\hat{D}$. We accept the prediction if parsing succeeds, the diagram is well formed, it denotes the same linear map as D, and it has no more vertices than the PyZX output $F(D)$.

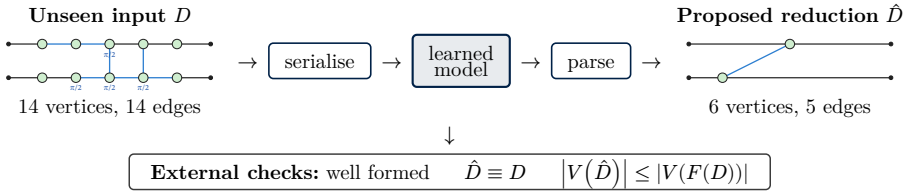


Figure 20: The ZX-reduction task. The learner receives a serialised graph but no ZX rules or semantic test. Parsing, equivalence and size relative to the PyZX output are checked externally after generation. Blue edges are Hadamard edges.

To check equivalence, we compose the source diagram D with the adjoint of the proposed diagram $\hat{D}$ and check whether the result reduces to the identity:

$$F\left(D \circ \hat{D}^{\dagger}\right) = I. \tag{106}$$

For Clifford diagrams, `full_reduce` is guaranteed to reduce a diagram representing the identity to bare wires. In the Clifford+T fragment, however, spider-nest identities can prevent this simplification from succeeding [104]. We count a prediction as a failure whenever the composed diagram does not reduce to bare wires, even though the source and predicted diagrams may still be semantically equivalent.

We report two evaluation metrics. The primary metric, *accepted-output accuracy*, is the fraction of predictions that parse, pass the equivalence check, and satisfy

$$\left|V\left(\hat{D}\right)\right| \leq |V(F(D))|. \tag{107}$$

Exact-match accuracy is the fraction that match the target sequence token for token. Listing 1 implements the three acceptance conditions. Here `Graph` is PyZX's `BaseGraph` interface, and `parse_graph` returns `Graph | None`.

```python
import pyzx

def accepts_prediction(source: Graph, target: Graph,
                       predicted_tokens: list[str]) -> bool:
    hypothesis: Graph | None = parse_graph(predicted_tokens)
    if hypothesis is None:
        return False

    difference = source + hypothesis.adjoint()
    pyzx.full_reduce(difference)
    if not difference.is_id():
        return False

    return hypothesis.num_vertices() <= target.num_vertices()
```

Listing 1: Prediction validation adapted from the original code. The parser reconstructs a graph, `full_reduce` tests equivalence, and the final comparison enforces the vertex limit.

5.4 Encoder–decoder Transformer

5.4.1 *Input representation and encoder*

We predict the output using an encoder–decoder Transformer with learned parameters θ [102]. Given the input sequence $x = (x_1, ..., x_m)$ as a list of tokens, the model first embeds each token x_i by assigning it a learned vector representation. The encoder then applies successive layers of self-attention and feed-forward updates. In self-attention, each token compares its current representation with those of every other input token and forms a weighted combination, assigning larger weights to the tokens most relevant to its update. Self-attention alone does not encode sequence order, so we use learned positional embeddings p_i and present $z_i = E[x_i] + p_i$ to the first encoder layer, where E is the token embedding table.

For one attention head, the layer maps the representation at each position to three vectors. The query describes what information that position is looking for, the key describes what information it offers, and the value carries the information that may be combined. These vectors are

$$q_i = W_Q z_i, \quad k_i = W_K z_i, \quad v_i = W_V z_i. \tag{108}$$

The matrices W_Q, W_K, and W_V are learned during training. The query and key vectors each have d_k components.

The attention score e_{ij} compares the query at position i with the key at position j, with a larger score indicating a stronger match. If their components are independent, with mean zero and unit variance, then the dot product has variance d_k. For large d_k, the resulting scores can push the softmax into a saturated region with very small gradients. Dividing by $\sqrt{d_k}$ keeps the scale of the scores approximately constant as the key dimension grows [102]. A softmax then converts the scores into non-negative attention weights that sum to one across j:

$$e_{ij} = \frac{q_i^T k_j}{\sqrt{d_k}}, \quad a_{ij} = \frac{\exp(e_{ij})}{\sum_{r=1}^{m} \exp(e_{ir})}. \tag{109}$$

Taking a weighted sum of the value vectors gives a *context vector*:

$$c_i = \sum_{j=1}^{m} a_{ij} v_j. \tag{110}$$

Thus a_{ij} measures how strongly position i attends to position j. Multiple heads repeat this calculation with different projection matrices, allowing the layer to capture different relationships between tokens. The layer combines their context vectors with the incoming representation and applies the same feed-forward neural network at every position. We denote the resulting updated embedding by h_i. This

becomes the input to the next layer. After L encoder layers, h_i summarises token x_i in the context of the complete input. We collect these final vectors as $H = \mathrm{Enc}_\theta(x) = (h_1, ..., h_m)$.

5.4.2 *Decoder and autoregressive generation*

The decoder generates the target from left to right. For this discussion, we write $y = (y_1, ..., y_n)$ with the final end token included. If $y_{<t} = (y_1, ..., y_{t-1})$ denotes the tokens preceding position t, then the probability assigned to the complete target factorises as

$$p_\theta(y \mid x) = \prod_{t=1}^{n} p_\theta(y_t \mid y_{<t}, x). \tag{111}$$

In other words, each factor gives the probability of the next token, conditioned on the source and the target tokens already generated. Generation begins with a special start token. At step t, the decoder embeds $y_{<t}$ and applies *masked self-attention*, meaning that position t may use earlier target positions but not later ones. It then uses cross-attention to select information from the encoded source vectors H. This is the same attention calculation as above, but its queries come from the decoder while its keys and values come from the encoder.

Let u_t be the decoder vector at position t, and let $\mathcal{V}$ be the token vocabulary. A final linear map assigns every possible next token $v \in \mathcal{V}$ an unnormalised score, called a *logit*. The softmax converts these logits into probabilities:

$$\ell_{t,v} = (W_{\mathrm{vocab}} u_t + b)_v, \quad p_\theta(v \mid y_{<t}, x) = \frac{\exp(\ell_{t,v})}{\sum_{w \in \mathcal{V}} \exp(\ell_{t,w})}. \tag{112}$$

The probabilities sum to one over the vocabulary. Once a token is chosen, it is appended to the prefix and the calculation is repeated until the decoder produces a special end token.

5.5 Training details

5.5.1 *Training data*

We compare models trained on random CNOT, Clifford, and Clifford+T circuit distributions. We generate training pairs on the fly, rather than storing a fixed data set. The main model is trained on ten-qubit Clifford circuits of depth 15, with $300,000$ new pairs sampled in each epoch [2].

Tests on unseen circuits of the same depth measure performance within the training distribution. Testing the same model on deeper circuits assesses generalisation to larger graphs and longer sequences.

5.5.2 *Sequence preparation*

The serialiser in Listing 2 implements the representation introduced in Figure 19. It writes the graph size, vertex records and edge records in order.

```
def serialise(graph: Graph) -> list[str]:
    vertices = sorted(graph.types())
    labels = {vertex: f"N{i}" for i, vertex in enumerate(vertices)}

    tokens = [str(graph.num_vertices()), str(graph.num_edges())]

    for vertex in vertices:
        tokens.extend([
            str(graph.type(vertex)),
            str(int(4 * graph.phase(vertex))),
        ])

    for edge in graph.edges():
        source, target = edge
        tokens.extend([
            str(graph.edge_type(edge)),
            labels[source],
            labels[target],
        ])

    return tokens
```

Listing 2: Graph serialiser adapted from the original code. The output contains the graph size, ordered vertex records, and typed edge records with endpoint identifiers.

The vocabulary contains fewer than one hundred distinct tokens. We prepend the special token `<sos>` to each target to supply the decoder's initial input and place `<eos>` after the final graph token. During inference, the model outputs `<eos>` to signal the end of generation. Shorter sequences are extended with `<pad>` tokens so that sequences of different lengths can be processed together in the same mini-batch. At ten qubits and depth 15, the paper reports 99th-percentile source and target lengths of approximately 360 and 210 tokens; at depth 40, these rise to approximately 650 and 400 [2].

5.5.3 *Training objective and decoding*

We train the decoder using *teacher forcing*: it receives the correct preceding tokens $y_{<t}$ when learning to predict y_t, even if it would itself have chosen a different token at an earlier step. For one training pair, we minimise the negative log-likelihood

$$\mathcal{L}(\theta) = -\sum_{t=1}^{n} \log p_\theta(y_t \mid y_{<t}, x). \tag{113}$$

Minimising this loss increases the probability assigned to the correct token at each position, with a larger penalty when that probability is very small. During optimisation, these contributions are averaged over the non-padding target tokens in each mini-batch.

At inference time, we instead feed the decoder its own previously generated tokens and choose the highest-probability token at each step. This procedure is called *greedy decoding*.

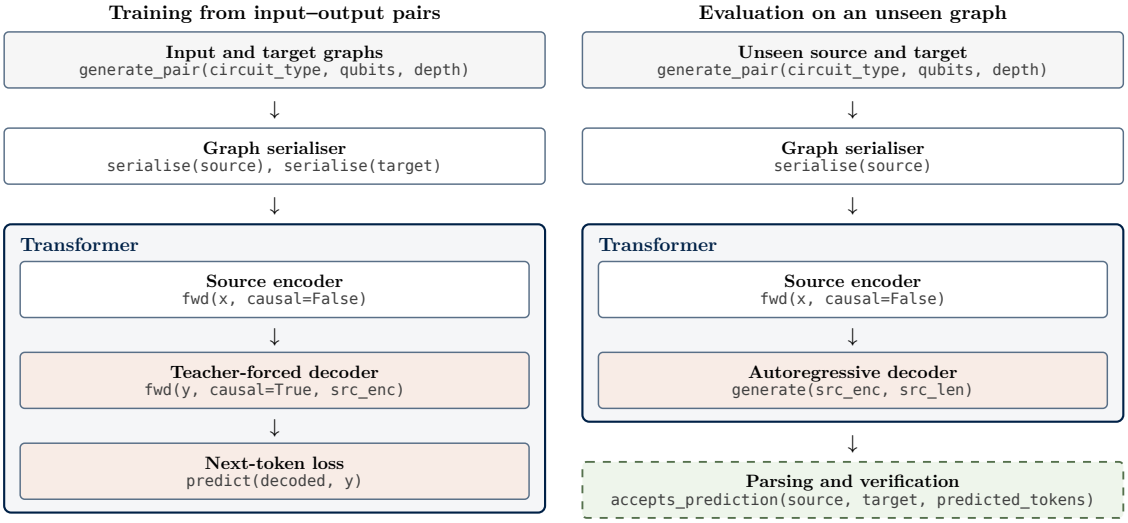


Figure 21: Training and evaluation in the implementation. During training, the decoder receives the correct preceding target tokens and is trained to predict the next token. During evaluation it generates a variable-length proposal one token at a time, using its own previous predictions. The `accepts_prediction` function performs graph parsing and mathematical verification outside the learned model.

Because inference uses the model's own predictions rather than the correct target prefix, errors can compound during generation. For example, one incorrect vertex identifier can produce a prefix that never occurred during training. This becomes increasingly important as graphs and their serialisations grow.

5.5.4 *Model configuration and checkpoint selection*

We use six layers in each of the encoder and decoder, 512-component token embeddings, and eight attention heads for the main Transformer. In the capacity comparisons, we vary the number of layers and, in some cases, the embedding dimension and number of heads. These experiments therefore compare complete model configurations rather than isolating the effect of layer count alone [2].

We set the base learning rate to 10^{-4} and save the model parameters periodically during training. We select a checkpoint based on its accepted-output accuracy on held-out examples, as defined in Section 5.3, rather than on training loss alone [2].

Across 770 epochs, the main model sees 231 million generated examples, only a fraction of the possible ten-qubit, depth-15 Clifford circuits [2].

5.6 Results

5.6.1 *Clifford reductions*

The main model achieves 96.4% accepted-output accuracy on held-out examples: 88.4% of predictions reproduce the PyZX target exactly, while a further 8.0% produce a different equivalent graph with the same number of vertices [2].

The gap between exact-match and accepted-output accuracy is informative. Some predictions differ from the `full_reduce` target but still represent the same linear map with no more vertices. This suggests that the model has learned features of valid reductions beyond memorising the outputs of `full_reduce`.

5.6.2 *Generalisation to greater circuit depth*

When the same model is applied without retraining to ten-qubit circuits deeper than the depth-15 training examples, accepted-output accuracy falls to 89.1% at depth 20, 63.2% at depth 25, and 29.5% at depth 30. Larger models generally improve accuracy, but none of the tested configurations overcomes the degradation as sequence length increases. The models in Figure 22 differ in their embedding dimensions and numbers of attention heads as well as their layer counts. The figure therefore compares complete model configurations and does not isolate the effect of adding layers.

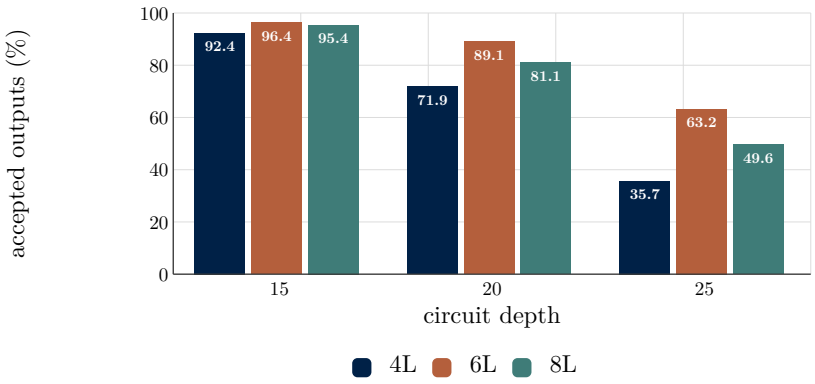


Figure 22: Accepted-output accuracy at increasing circuit depth. `4L`, `6L`, and `8L` denote the numbers of layers. The six-layer series is the main model with 512-component embeddings and eight heads; the eight-layer series uses 640-component embeddings and ten heads. Data from [2].

5.6.3 *Circuit families*

Separate eight-layer models were also trained on ten-qubit CNOT, Clifford, and Clifford+T circuit distributions. The CNOT model is more accurate than the Clifford model at all three tested depths, consistent with the restricted CNOT gate set

making the compilation problem easier. The Clifford+T models also exceed 90% throughout the tested range.

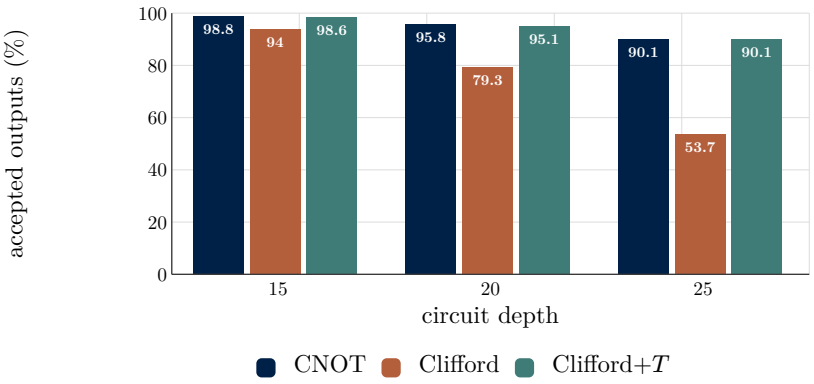


Figure 23: Accepted-output accuracy for separately trained eight-layer models on ten-qubit CNOT, Clifford, and Clifford+T distributions. Data from [2].

The Clifford+T results show that the model can learn to simplify diagrams containing non-Clifford phases. We assess these reductions by comparing their vertex counts with those obtained by the `full_reduce` heuristic on the sampled circuit families. Whether this also leads to fewer gates in the extracted circuits remains a question for further work.

5.7 Discussion and conclusion

The results show that a Transformer can learn a complex ZX-graph transformation from input–output examples alone. Given no rewrite rules or intermediate steps, it produces valid, semantics-preserving transformations on unseen examples, occasionally finding alternative reductions to those produced by `full_reduce`.

However, the accuracy of the model falls as circuits and their serialisations grow. This is consistent with the difficulty Transformers have in generalising beyond training sequence lengths [105]. Longer outputs also create more opportunities for one incorrect token to corrupt the remainder of the sequence.

A natural next step is to restrict the model to semantics-preserving actions, allowing it to focus on learning which choices lead to good solutions. Reinforcement learning lets an agent learn to choose among these actions using rewards for the quality of the result, while remaining within a sound rewriting framework. The agent need not reproduce a fixed target sequence. We adopt this approach in the next chapter to minimise the two-qubit gate count of Clifford and CNOT circuits.

For future work, we could explore alternative policy-optimisation methods and human feedback for ZX rewriting. Shao et al. [106] introduced group relative policy optimisation (GRPO), a PPO variant that compares sampled outputs without a separate learned value model. Ouyang et al. [107] used human rankings to train a reward model for reinforcement learning from human feedback (RLHF). Similar

feedback could tune a rewriting engine towards preferences that are difficult to formalise, while symbolic verification enforces correctness.

Chapter VI

Equivariant Reinforcement Learning for Matrix-Group Circuit Synthesis

Clifford circuits are quantum circuits generated by the gate set $\{H, S, \text{CNOT}\}$. Clifford circuits form an important class because they are efficiently simulable, yet expressive enough to exhibit entanglement and non-local correlations characteristic of quantum mechanics. Because of these properties, Clifford circuits feature prominently in quantum error correction [68], quantum key distribution, and quantum teleportation [21].

Parity circuits, which are generated solely by CNOT gates, form a subclass of Clifford circuits. CNOT is the only entangling gate in the generating Clifford gate set and is powerful enough to express the linear reversible fragment of classical logic. When supplemented with phase gates, these parity computations give rise to phase polynomials, which are used to represent and optimise CNOT+phase circuits. More generally, supplementing Clifford circuits with non-Clifford phase gates gives the Clifford+phase circuits studied in Chapter 4 and, in the special case of T gates, the approximately universal Clifford+T gate set.

Clifford circuits admit compact symplectic tableaus, while parity circuits admit compact invertible binary matrices (Figure 24). Unlike exponentially large unitary matrices, these representations use polynomial space and encode gate actions exactly. We call synthesis using these representations *matrix-group synthesis*.

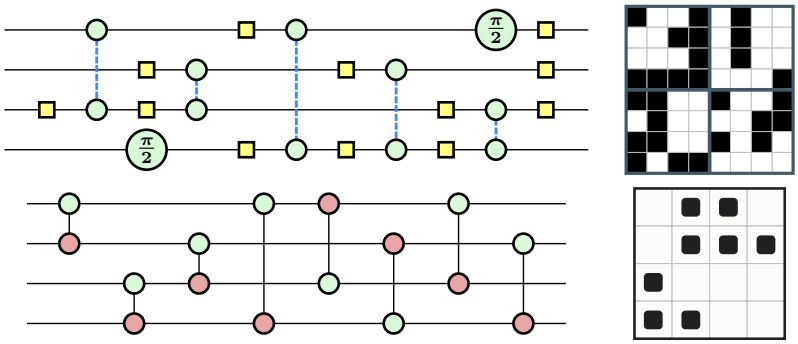


Figure 24: A Clifford circuit and its symplectic tableau (top), and a parity circuit and its parity matrix (bottom), adapted from [4].

The main optimisation objective for Clifford and CNOT circuits on near-term quantum devices is to reduce the number of two-qubit entangling gates, since they are typically more expensive and error-prone than single-qubit gates. For instance, Quantinuum reports typical one- and two-qubit gate infidelities of 3×10^{-5} and 1×10^{-3}, respectively, for its H2 system [53]. Reducing the number of entangling gates in Clifford circuits is also relevant in the fault-tolerant setting. In joint work, we developed CSSCat, which constructs fault-tolerant preparation circuits for CSS stabiliser states [108].

Although the CNOT and Clifford synthesis algorithms in Chapter 2 achieve asymptotically optimal worst-case entangling-gate counts, they need not give the shortest circuit for a particular target. We therefore ask whether learning can find shorter circuits at practical sizes, and how closely these circuits approach the optimum.

Finding optimal circuits quickly becomes intractable: the six-qubit Clifford group contains about 2.1×10^{23} elements, and constructing the corresponding optimal-circuit database required 2.1 TB of storage and more than $300,000$ CPU-hours [109]. To tackle this large search space, we instead combine reinforcement learning and curriculum learning with permutation-equivariant neural architectures tailored to the matrix representation and gate actions of each problem. This produces one reusable policy for each synthesis task, trained without optimal circuit labels and applicable to unseen targets and qubit counts. The contributions of this chapter are as follows.

Chapter Contributions

- **Matrix-group synthesis with reinforcement learning.** We formulate Clifford and parity synthesis as weighted word problems over $\mathrm{Sp}(2n, \mathbb{F}_2)$ and $\mathrm{GL}(n, \mathbb{F}_2)$, respectively, and solve them as sequential matrix-reduction tasks using reinforcement learning.
- **Reusable equivariant neural policies.** We develop problem-specific architectures whose shared layers respect qubit relabelling and whose action heads reflect the symmetry properties of the gate set. This shared parametrisation allows a single policy to be trained across both circuit difficulty and qubit count using the two-dimensional random-walk curriculum introduced here.
- **Near-optimal CNOT synthesis and transfer.** The CNOT policy remains within 0.73 gates of the certified mean optimum through seven qubits. After training up to 20 qubits, the same model transfers to 30 qubits and outperforms existing baselines for both finite-length random-walk and uniformly sampled targets.

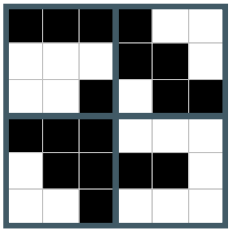
- **Near-optimal Clifford synthesis and transfer.** We recover 995 of the 1003 certified six-qubit optima of Bravyi et al. using policy-guided search, with each of the remaining eight circuits one CZ gate above optimum. A Clifford policy trained up to ten qubits transfers to 30-qubit random-walk targets and outperforms existing baselines. On uniformly sampled targets, it also produces substantially shorter circuits on the instances it solves, although its solve rate falls to 59% at 30 qubits.

We first recall the matrix representations of Clifford and CNOT circuits, formulate matrix-group synthesis, and introduce the comparison methods (Sections 6.1–6.3). We then develop and evaluate our reinforcement-learning approach: we define the synthesis environments, introduce the equivariant policy architectures, describe training and decoding, and report the CNOT and Clifford results (Sections 6.4–6.10). Finally, we connect matrix-group synthesis to circuit extraction from reduced ZX-diagrams (Section 6.11) and discuss the scope and limitations of our approach (Section 6.12).

6.1 Matrix representations

Chapter 2 introduced Clifford circuits and their tableau representation. Recall that a signed Clifford-operation tableau $(M|r)$ has two components: a symplectic matrix $M \in \mathrm{Sp}(2n, \mathbb{F}_2)$ and a vector of signs r. To simplify notation, throughout this chapter we omit r and depict the phase-free matrix M using the following grid representation.

	x_1	x_2	x_3	z_1	z_2	z_3	r	
X_1	1	1	1	1	0	0	1	$-Y_1X_2X_3$
X_2	0	0	0	1	1	0	0	Z_1Z_2
X_3	0	0	1	0	1	1	1	$-Z_2Y_3$
Z_1	1	1	1	0	0	0	0	$X_1X_2X_3$
Z_2	0	1	1	1	1	0	0	$Z_1Y_2X_3$
Z_3	0	0	1	0	0	0	0	X_3



Each row of M records the phase-free image of one of the Pauli generators X_i or Z_i under conjugation by the Clifford unitary, and the matrix as a whole determines that unitary up to a Pauli correction and global phase. The omitted signs can be restored using only single-qubit gates, without increasing the two-qubit gate count, as shown in Section 6.2.3.

CNOT circuits form a subclass of Clifford circuits and are represented by invertible parity maps $A \in \mathrm{GL}(n, \mathbb{F}_2)$. Under the tableau convention of Section 2.3.2, Equation 33 embeds a parity map into the Clifford symplectic representation as

$$A \mapsto \mathrm{diag}(A^T, A^{-1}). \tag{114}$$

Thus A alone determines the phase-free CNOT tableau. To simplify notation, we depict the parity map using the following grid representation.

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \qquad \begin{array}{|cccc|} \hline & \blacksquare & \blacksquare & \\ & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & & & \\ \blacksquare & \blacksquare & & \\ \hline \end{array}$$

We distinguish the parity grid from the Clifford grid by using different markers and omitting the quadrant dividers.

6.2 Matrix-group synthesis

Circuit synthesis is the task of constructing a circuit that implements a specified target unitary. Because many circuits can realise the same unitary, synthesis is often posed as an optimisation problem, such as minimising the number of two-qubit gates. Throughout this chapter, synthesis is exact: the output circuit must realise the target unitary rather than merely approximate it.

The matrix representations of Clifford and CNOT circuits allow us to formulate both compilation tasks as instances of a common matrix-group synthesis problem.

Definition 50 Let $\mathcal{G}_n$ be a finite matrix group, let $\mathcal{A}_n \subseteq \mathcal{G}_n$ be a generating set, and let $c_n : \mathcal{A}_n \to \mathbb{R}_{\geq 0}$ assign a cost to each generator. Given a target $M_{\text{target}} \in \mathcal{G}_n$, the *matrix-group synthesis problem* is to find a length $L \in \mathbb{N}_0$ and generators $G_1, ..., G_L \in \mathcal{A}_n$ such that

$$M_{\text{target}} = G_1 ... G_L, \tag{115}$$

while minimising $\sum_{t=1}^{L} c_n(G_t)$.

In this chapter, it is more convenient to use the reverse formulation of the synthesis problem. The CNOT and Clifford generating sets considered below consist entirely of involutions, so the same generators can be used to reduce a target back to the identity.

Definition 51 Given an instance of the matrix-group synthesis problem for which every generator in $\mathcal{A}_n$ is an involution, its *reverse formulation* is to find a length $L \in \mathbb{N}_0$ and generators $G_1, ..., G_L \in \mathcal{A}_n$ such that

$$M_{\text{target}} G_1 ... G_L = I, \tag{116}$$

while minimising $\sum_{t=1}^{L} c_n(G_t)$. Reversing this sequence gives a factorisation of the target, and hence a solution to the original synthesis problem:

$$M_{\text{target}} = G_L ... G_1. \tag{117}$$

6.2.1 *CNOT synthesis over* $\mathrm{GL}(n, \mathbb{F}_2)$

To formulate CNOT compilation as a matrix-group synthesis problem, we use the reverse formulation from Definition 51 and reduce the target parity map to the identity by right multiplication. For parity maps, $A \to AG$ represents the gate G followed by the circuit for A. Thus, right multiplication **prepends** the corresponding CNOT, and the synthesised circuit follows the reduction order. The group and directed generating set are

$$\mathcal{G}_n^{\mathrm{CNOT}} = \mathrm{GL}(n, \mathbb{F}_2), \quad \mathcal{A}_n^{\mathrm{CNOT}} = \left\{ \mathsf{CNOT}_{i,j} : i, j \in [n], i \neq j \right\}. \tag{118}$$

We use a sans-serif typeface for the matrix generator $\mathsf{CNOT}_{i,j}$ to distinguish it from the corresponding unitary gate $\mathrm{CNOT}_{i,j}$. Writing e_i for the ith standard basis vector, its parity-map matrix is

$$\mathsf{CNOT}_{i,j} := I_n + e_j e_i^T, \quad \mathsf{CNOT}_{i,j}^2 = I_n. \tag{119}$$

Every generator has unit cost:

$$c_n^{\mathrm{CNOT}} : \mathcal{A}_n^{\mathrm{CNOT}} \to \mathbb{R}_{\geq 0}, \quad c_n^{\mathrm{CNOT}}\left(\mathsf{CNOT}_{i,j}\right) = 1. \tag{120}$$

Given a target $M_{\mathrm{target}} = A \in \mathrm{GL}(n, \mathbb{F}_2)$, the total cost $\sum_{t=1}^{L} c_n^{\mathrm{CNOT}}(G_t)$ is exactly the number of CNOT gates. Right multiplication by $\mathsf{CNOT}_{i,j}$ adds column j to column i, as shown in Figure 25. This is the transpose counterpart of the row-operation formulation in Section 2.3.1.1: adding row j to row i in A^T is equivalent to adding column j to column i in A. The same elimination principles therefore apply, but the control–target labels and circuit order follow the right-multiplication convention used here. The worked example in Figure 26 gives a complete four-qubit reduction. We perform an exhaustive breadth-first search over all 20,160 elements of $\mathrm{GL}(4, \mathbb{F}_2)$ and first reach the target at depth seven, certifying that the circuit has minimum CNOT count.

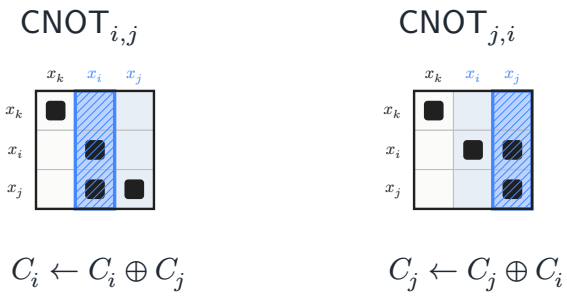


Figure 25: Column operations induced by the parity-map generators $\mathsf{CNOT}_{i,j}$ and $\mathsf{CNOT}_{j,i}$.

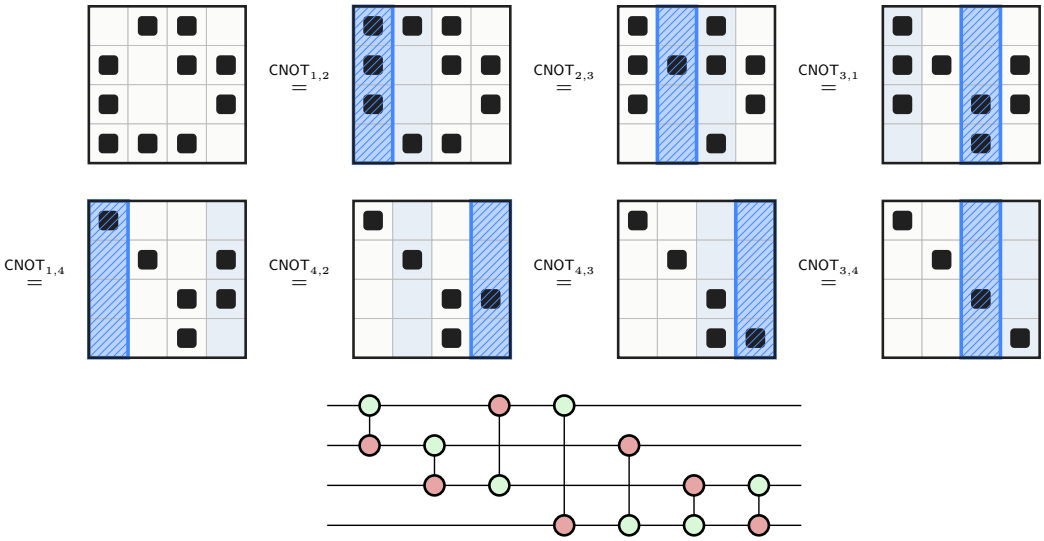


Figure 26: A seven-CNOT reduction of a four-qubit parity map to the identity, with the corresponding ZX circuit. Note that the circuit follows the reduction order.

6.2.2 *Clifford synthesis over* $\mathrm{Sp}(2n, \mathbb{F}_2)$

Clifford compilation can likewise be formulated as a matrix-group synthesis problem. Using the reverse formulation from Definition 51, we reduce the target tableau to the identity by right multiplication. Under our tableau convention, right multiplication by the gate tableau M_G gives $M_U M_G = M_{GU}$, which represents applying G after U. The synthesised circuit therefore follows the reverse reduction order. For Clifford tableaus, the group and generating set are

$$\mathcal{G}_n^{\mathrm{Cl}} = \mathrm{Sp}(2n, \mathbb{F}_2), \quad \mathcal{A}_n^{\mathrm{Cl}} := \{\mathsf{H}_i, \mathsf{S}_i : i \in [n]\} \cup \{\mathsf{CZ}_{i,j} : i, j \in [n], i < j\}. \quad (121)$$

Each generator is identified with the symplectic matrix of its corresponding physical gate. In the phase-free representation, all three generators are involutions:

$$\mathsf{H}_i^2 = \mathsf{S}_i^2 = \mathsf{CZ}_{i,j}^2 = I_{2n}. \quad (122)$$

Single-qubit generators have zero cost, while each CZ gate has unit cost:

$$c_n^{\mathrm{Cl}} : \mathcal{A}_n^{\mathrm{Cl}} \to \mathbb{R}_{\geq 0}, \quad c_n^{\mathrm{Cl}}(\mathsf{H}_i) = c_n^{\mathrm{Cl}}(\mathsf{S}_i) = 0, \quad c_n^{\mathrm{Cl}}(\mathsf{CZ}_{i,j}) = 1. \quad (123)$$

Given a target $M_{\mathrm{target}} \in \mathrm{Sp}(2n, \mathbb{F}_2)$, the total cost $\sum_{t=1}^{L} c_n^{\mathrm{Cl}}(G_t)$ is exactly the number of CZ gates. Since CNOT and CZ differ only by local Hadamard conjugation, this is equivalently the CNOT count when single-qubit gates are free. There are $2n + \binom{n}{2}$ available actions, of which only the $\binom{n}{2}$ CZ actions carry non-zero cost. Right multiplication by a generator performs the column operations shown in Figure 27.

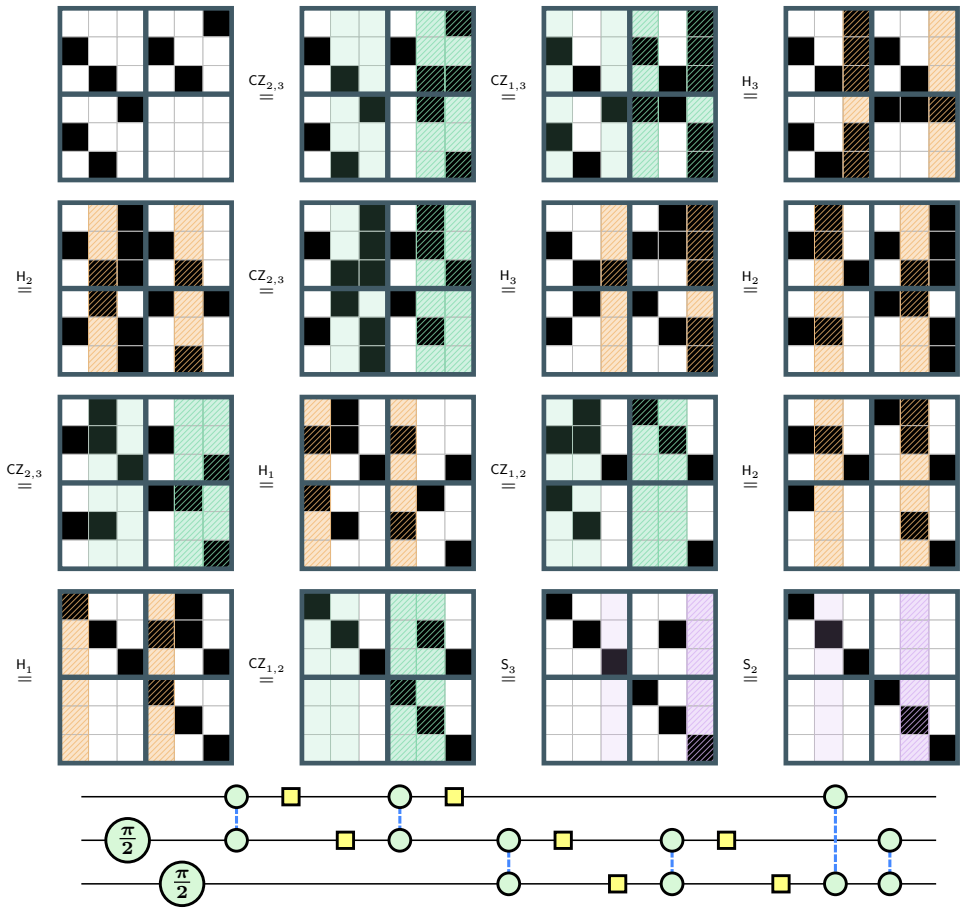


Figure 28: Reduction of a three-qubit Clifford tableau to the identity at optimal CZ cost six, with the corresponding ZX circuit. Note that the circuit follows the reverse reduction order.

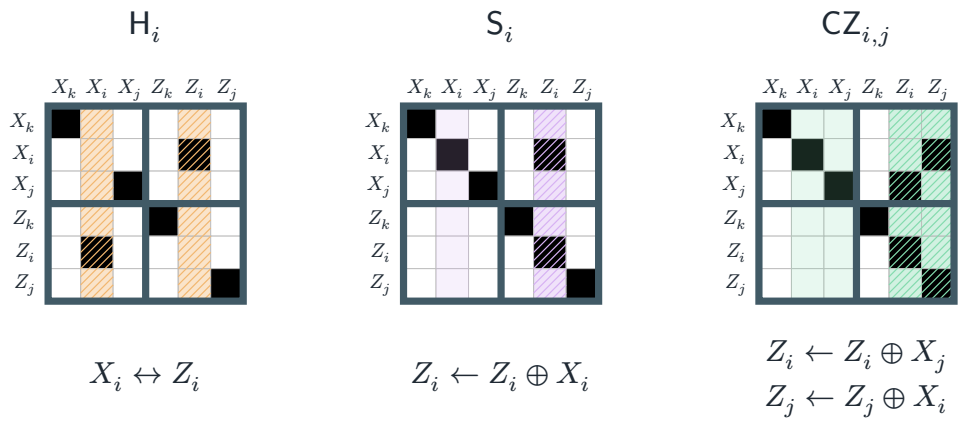


Figure 27: Column operations induced by the Clifford generators H_i, S_i, and $\mathsf{CZ}_{i,j}$.

The same column operations give the complete three-qubit reduction in Figure 28. We verify its optimality by exhaustive breadth-first search over all 1,451,520 elements of $\mathrm{Sp}(6, \mathbb{F}_2)$, exploring all free H and S moves at each cost before adding another CZ gate. The target is first reached at a cost of six CZ gates.

The corresponding classical synthesis methods are reviewed in Section 6.3.2. The experiments in Section 6.10 compare against the scalable baselines and use optimal six-qubit circuits to assess exact recovery.

6.2.3 *Recovering the sign vector*

To recover the Pauli correction, apply the phase-free reduction sequence to the full signed tableau using the sign-aware Clifford updates of Aaronson and Gottesman [28], reviewed in Section 2.3.2. The symplectic-matrix updates do not depend on the sign column, so the same sequence reduces the matrix to the identity. The remaining signs determine the final single-qubit Pauli correction.

Concretely, let the initial signed-tableau be $(M|s)$, and suppose that the phase-free reduction uses generators $G_1, ..., G_L$ such that

$$MG_1...G_L = I_{2n}. \tag{124}$$

Propagating the full tableau through these gates leaves a terminal tableau of the form

$$(I_{2n}|r), \quad r \in \mathbb{F}_2^{2n}. \tag{125}$$

The first n entries of r record the signs of the images of $X_1, ..., X_n$, and the final n record those of $Z_1, ..., Z_n$. Since X_i and Z_i anticommute, $r_i = 1$ calls for a Z_i clean-up, while $r_{i+n} = 1$ calls for an X_i clean-up. Up to global phase, the terminal map and its Pauli clean-up are therefore both

$$\prod_{i=1}^{n} X_i^{r_{i+n}} Z_i^{r_i}. \tag{126}$$

Figure 29 illustrates this construction.

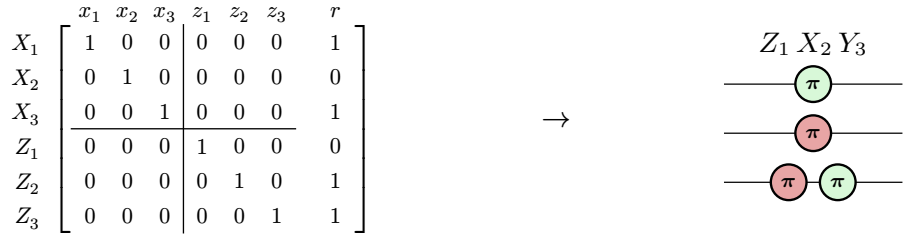


Figure 29: A terminal signed identity tableau and its Pauli clean-up $Z_1X_2Y_3$.

Applying this clean-up completes the tableau reduction; after reversal, it appears at the input end of the synthesised circuit.

6.2.4 *The Cayley-graph formulation*

The matrix-group synthesis problem can be cast as a minimum-cost path problem on a Cayley graph. For a matrix group $\mathcal{G}_n$ and generating set $\mathcal{A}_n$, the corresponding Cayley graph has vertex set $\mathcal{G}_n$. Each generator $G \in \mathcal{A}_n$ contributes a G-labelled edge from every $M \in \mathcal{G}_n$ to MG:

$$M \xrightarrow{G} MG, \quad M \in \mathcal{G}_n, \quad G \in \mathcal{A}_n. \tag{127}$$

We assign each G-labelled edge the cost $c_n(G)$. Paths from the target to the identity then encode reductions, with path cost equal to synthesis cost. Since every generator considered here is an involution, each edge can be traversed in either direction at the same cost, so the underlying weighted Cayley graph is undirected.

The Cayley graph is finite for each n, but the following proposition shows that its number of vertices grows super-exponentially in the number of qubits.

Proposition 52 *For every $n \geq 1$, the orders of the CNOT parity-map group and the phase-free Clifford tableau group are*

$$\begin{aligned} |\mathrm{GL}(n, \mathbb{F}_2)| &= \prod_{k=0}^{n-1} \left(2^n - 2^k\right), \\ |\mathrm{Sp}(2n, \mathbb{F}_2)| &= 2^{n^2} \prod_{j=1}^{n} (4^j - 1). \end{aligned} \tag{128}$$

In particular, both orders are $2^{\Theta(n^2)}$.

We derive both group orders in Section A.1. Bravyi, Latone, and Maslov [109] give the same expression for the binary symplectic group.

For the generating set in Equation 118, the CNOT Cayley graph has $n(n-1)$ neighbours at every vertex, one for each directed generator $\mathsf{CNOT}_{i,j}$ with $i \neq j$. For the generating set in Equation 121, the Clifford Cayley graph has

$$2n + \binom{n}{2} = \frac{n^2 + 3n}{2} \tag{129}$$

neighbours, comprising H_i and S_i for each qubit and one $\mathsf{CZ}_{i,j}$ for each unordered pair. An exhaustive breadth-first search may therefore visit and store $2^{\Theta(n^2)}$ vertices, examining $\Theta(n^2)$ generator moves from each vertex.

n	$\lvert\mathbf{GL}(n,\mathbb{F}_2)\rvert$	$\lvert\mathbf{Sp}(2n,\mathbb{F}_2)\rvert$
2	6	720
3	168	1,451,520
4	20,160	47,377,612,800
5	9,999,360	24,815,256,521,932,800
6	20,158,709,760	208,114,637,736,580,743,168,000

Table 2: Orders of the CNOT parity-map and phase-free Clifford tableau groups at two to six qubits.

The Cayley-graph viewpoint, illustrated for the CNOT and Clifford generating sets in Figures 30–31, also informs the curriculum-learning approach developed in Section 6.7.1. Since the weighted shortest-path distance from a target to the identity is its minimum synthesis cost, targets farther from the identity require more two-qubit gates and generally make harder training examples. We use this distance indirectly by sampling targets through random walks on the Cayley graph, starting at the identity, and increasing the walk length throughout training. Although walk length is only a proxy for shortest-path distance, it provides a controllable notion of difficulty. For three or more qubits, these random walks converge to the uniform distribution, as established in Theorem 54 and Theorem 55.

6.3 Comparison methods and related work

To assess the learned policies, we compare their circuits with those produced by scalable synthesis algorithms and, for small instances, with circuits of optimal entangling-gate count. We specify the implementations and cost conventions in Section 6.8.

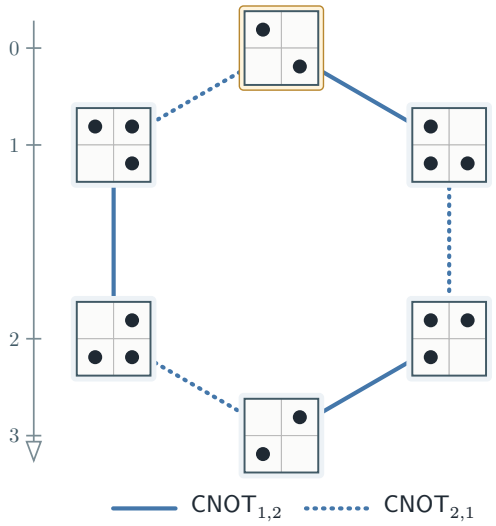


Figure 30: The complete Cayley graph of $\mathrm{GL}(2,\mathbb{F}_2)$ induced by the two directed two-qubit CNOT generators. Vertices are arranged by word distance from the identity: each lower level requires one additional CNOT.

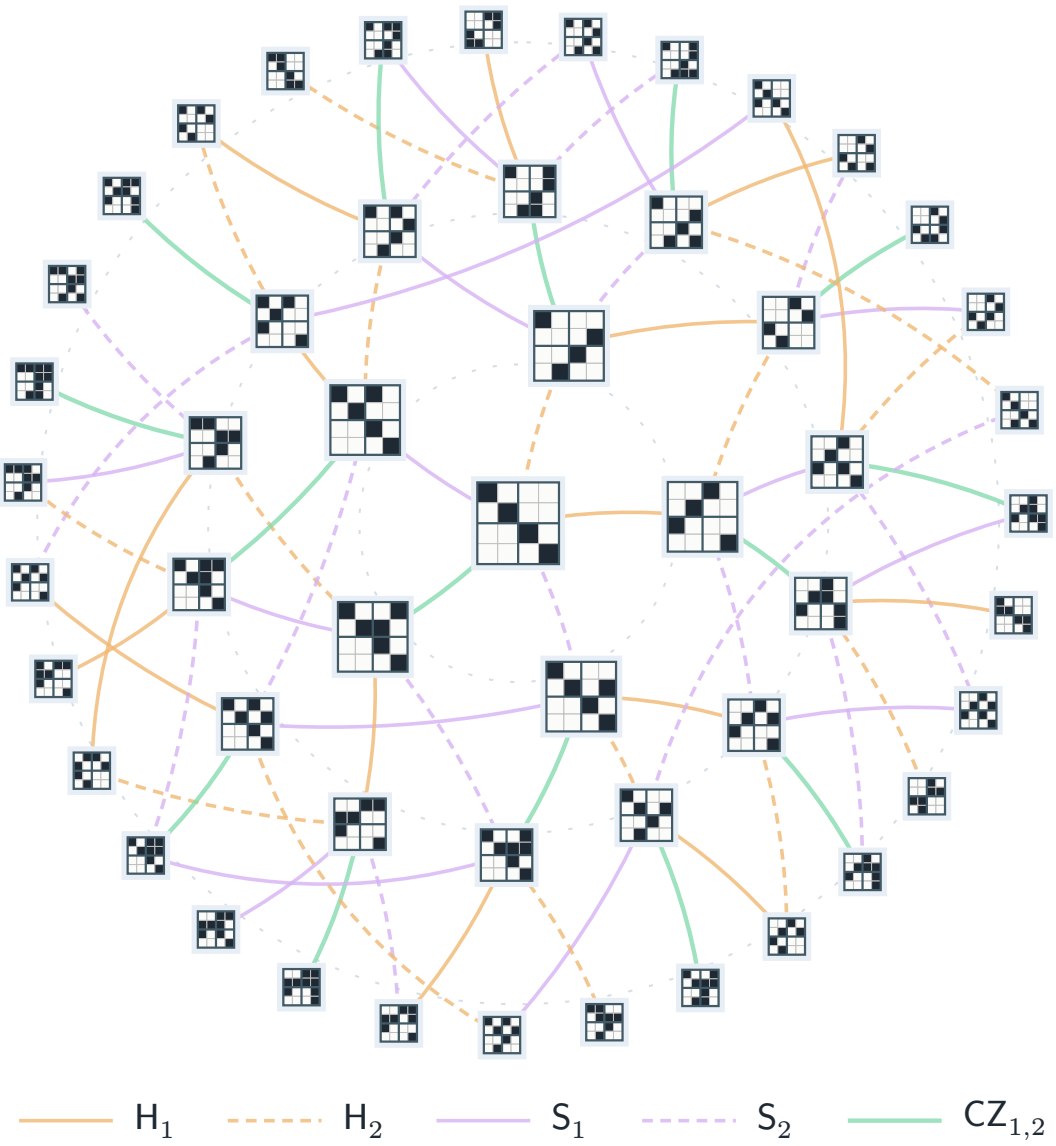


Figure 31: The radius-three ball about the identity in the Cayley graph generated by the two-qubit Clifford gates. Vertices are phase-free Clifford tableaus; edge colour and style identify the generator. The central identity tableau is the terminal state for reverse reduction and the starting state for random-walk target generation.

6.3.1 *CNOT comparison methods*

Patel–Markov–Hayes (PMH). We use PMH, the asymptotically optimal block-elimination algorithm described in Section 2.3.1.2.

Greedy Gaussian elimination (GGE) chooses row additions by matching portions of rows, so that a single addition clears several entries without disturbing columns already eliminated [110]. We use the FastGreedyGE variant, which first triangularises the matrix and then reduces it to the identity.

Alternating elimination with cost minimisation (AECM) applies operations from both the left and the right to clear a pivot row and column [111], adding gates at opposite ends of the circuit. It scores candidate sequences by the decrease

in the number of entries differing from the identity across both the matrix and its inverse, per added CNOT.

6.3.2 *Clifford comparison methods*

Aaronson–Gottesman [28] reduces a Clifford tableau to the identity one qubit at a time. We use Qiskit's `synth_clifford_ag` implementation, which has an $O(n^2)$ worst-case gate bound.

Qiskit greedy. We use the greedy compiler of Bravyi et al. [112], as implemented in Qiskit [113]. It reduces the tableau one qubit at a time, choosing at each step the qubit whose reduction requires the fewest CNOTs.

Bravyi–Maslov decomposes a Clifford into Hadamard-free components, a Hadamard layer, and a qubit permutation [114]. The Hadamard-free components are synthesised using CNOT, CZ, and single-qubit phase gates. We include the cost of implementing the permutation in the comparison.

6.3.3 *Exact references and other search methods*

For CNOT synthesis, we use the optimal synthesis results of Christensen et al. [115] through seven qubits. For Clifford synthesis, we use our own meet-in-the-middle solver [4] for the random-walk benchmarks and the subset of 1,003 six-qubit instances with known optimal CZ counts used by Bravyi et al. [112]. These comparisons are reported in Section 6.9 and Section 6.10.2.

Other approaches include heuristic shortest-path search and SAT-based synthesis. Webster et al. [116] guide search using matrix-based estimates, with fitted heuristics and bounded queues trading optimality guarantees for tractability. SAT-based methods search for a circuit within a prescribed cost bound and certify optimality by ruling out every lower cost, with applications to CNOT [117] and Clifford synthesis [118]. We discuss these approaches as related work but do not include them in our benchmarks.

6.3.4 *Connectivity and comparison scope*

Our experiments use all-to-all connectivity and fixed input and output labels, as specified in Section 2.3.6. Comparisons with architecture-aware methods require matching these connectivity and mapping conventions.

6.3.5 *Learned synthesis*

In concurrent work, AlphaCNOT combines a learned policy with Monte Carlo tree search to minimise CNOT count [119]. We compare it with our CNOT policy in Section 6.9.

6.4 Reinforcement learning for matrix-group synthesis

6.4.1 *Why reinforcement learning?*

We formulate matrix-group synthesis as a deterministic, single-player game in which a learned policy chooses gates to reduce the target matrix to the identity at minimum total cost.

In our setting, reinforcement learning offers an advantage over supervised learning by avoiding the need for labelled factorisations. Optimal labels are prohibitively expensive beyond small instances, while labels from an existing algorithm encode its limitations and select only one of the many valid factorisations of a target matrix. Reinforcement learning instead evaluates a trajectory by whether it reaches the identity and by the generator cost incurred, allowing it to discover alternative factorisations with the same or lower cost.

Reinforcement learning also offers an advantage over generative methods, including sequence-to-sequence ZX rewriting in Chapter 5 and diffusion-based circuit synthesis [120]. These methods produce a complete circuit or diagram whose semantic equivalence to the target must be verified after generation. In the matrix-group environment, every action has an exact algebraic effect, so any trajectory that reaches the identity certifies a valid factorisation, and hence a valid circuit, by Definition 51.

6.4.2 *Reinforcement-learning basics*

A reinforcement-learning problem can be formulated as a *Markov decision process* (MDP). In the deterministic setting used here, an MDP consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. At step t, an *agent* observes the state S_t and selects an action A_t, after which the *environment* returns

$$S_{t+1} = \mathcal{T}(S_t, A_t), \quad R_t = \mathcal{R}(S_t, A_t). \tag{130}$$

An *episode* is a sequence of interactions between the agent and the environment, from an initial state until termination. The agent selects actions according to a *policy* $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, which assigns each state a probability distribution over actions. Training adjusts the policy to maximise the expected discounted return

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\tau-1} \gamma^t R_t \right], \tag{131}$$

where τ is the episode length, bounded by a step cap, and $\gamma \in [0, 1)$ is the discount factor. A *value function* estimates the expected future return from a state under the policy. The policy-value networks in Section 6.6 therefore return both action scores and a scalar value estimate; their use during training and evaluation is described in Section 6.8.1 and Section 6.8.2.

6.4.3 *Matrix-group MDP*

For each qubit count n, let $\mathcal{G}_n$ be the matrix group generated by the available gate matrices $\mathcal{A}_n$. The corresponding MDP has state space $\mathcal{S}_n := \mathcal{G}_n$ and action space $\mathcal{A}_n$. Every action is available from every state, with deterministic transition

$$\mathcal{T}_n(M, G) := MG. \tag{132}$$

Thus the agent traverses the Cayley graph by right multiplication. The size-agnostic architectures in Section 6.6 share parameters θ across qubit counts, inducing

$$\pi_{\theta,n} : \mathcal{S}_n \to \mathcal{P}(\mathcal{A}_n) \quad \text{for every } n. \tag{133}$$

When the qubit count is clear, we omit n and write π_θ.

Each episode starts at $M_0 = M_{\text{target}}$ and ends successfully on reaching the identity, or unsuccessfully on reaching the step limit. The reward function can penalise generator costs, reward reaching the identity, and, where useful, provide a dense measure of progress. These shaping terms influence learning but do not determine whether a decoded circuit is accepted: only a trajectory that reaches the identity is a valid synthesis, and successful outputs are compared using the stated gate-cost objective.

6.4.4 *Clifford environment*

For Clifford synthesis, the state space is $\mathcal{S}_n := \mathrm{Sp}(2n, \mathbb{F}_2)$ and the action space is $\mathcal{A}_n^{\mathrm{Cl}}$, comprising the generator matrices from Equation 121: H_i and S_i for each qubit, and $\mathsf{CZ}_{i,j}$ for each unordered pair.

Each CZ action incurs a penalty of 1, while each H or S action incurs a smaller penalty of 0.01 to discourage cycles. We also give a bonus of 25 for reaching the identity and include a normalised Hamming-distance shaping term:

$$\begin{aligned}\mathcal{R}_n(M, G) := &-0.01\mathbb{1}(G = \mathsf{H}_i \ \text{ or } \ G = \mathsf{S}_i) - \mathbb{1}\big(G = \mathsf{CZ}_{i,j}\big) \\ &+25\mathbb{1}(MG = I_{2n}) - \frac{\|MG - I_{2n}\|_0}{8n^2}.\end{aligned} \tag{134}$$

Here $\mathbb{1}$ denotes an indicator function and $\|\cdot\|_0$ counts the non-zero matrix entries. The terminal bonus discourages the policy from applying only low-penalty one-qubit gates, while the final term provides a dense measure of progress towards the identity.

6.4.5 *CNOT environment*

For CNOT synthesis, the state space is $\mathcal{S}_n := \mathrm{GL}(n, \mathbb{F}_2)$ and the action set is $\mathcal{A}_n^{\mathrm{CNOT}}$, consisting of the $n(n-1)$ directed generators $\mathsf{CNOT}_{i,j}$ with $i \neq j$ from Equation 118. Since every action is a costly two-qubit gate, we use

$$\mathcal{R}_n(M, G) := \begin{cases} 24 & \text{if } \ MG = I_n \\ -1 & \text{otherwise.} \end{cases} \tag{135}$$

Thus every generator receives reward -1, while reaching the identity gives a $+25$ success bonus. A Hamming-weight shaping term may also be added as a warm-up signal; AlphaCNOT [119] reports that this works well in practice. In our CNOT runs, cross-size curriculum learning replaces this warm-up, so we omit dense Hamming-weight shaping.

6.5 Permutation symmetry

Our cost model allows gates between every pair of qubits and assigns costs independently of qubit labels. Such connectivity is available, for example, on trapped-ion systems such as Quantinuum H2 [53]. Under this cost model, the qubit labels are interchangeable: relabelling the qubits in a target matrix simply relabels the gates in a synthesis without changing its cost. The same symmetry carries over to the reinforcement-learning problem, where relabelling the state should relabel the policy's action probabilities while leaving its value estimate unchanged. These are the equivariance and invariance properties formalised below.

Denote by $\mathrm{Sym}(n)$ the group of permutations of n elements, i.e. its elements are bijections of the form $\sigma : \{1, ..., n\} \to \{1, ..., n\}$. If P_σ is the permutation matrix associated with σ, its action on the CNOT state space $\mathrm{GL}(n, \mathbb{F}_2)$ and generators is

$$\sigma \cdot M \coloneqq P_\sigma M P_\sigma^T, \quad \sigma \cdot \mathsf{CNOT}_{i,j} \coloneqq \mathsf{CNOT}_{\sigma(i),\sigma(j)}. \tag{136}$$

For Clifford synthesis, define $\Pi_\sigma \coloneqq \begin{pmatrix} P_\sigma & 0 \\ 0 & P_\sigma \end{pmatrix}$, which applies P_σ to both the X and Z coordinates. Its action on $\mathrm{Sp}(2n, \mathbb{F}_2)$ and the Clifford generators is

$$\begin{aligned} &\sigma \cdot M \coloneqq \Pi_\sigma M \Pi_\sigma^T, \\ &\sigma \cdot \mathsf{H}_i \coloneqq \mathsf{H}_{\sigma(i)}, \quad \sigma \cdot \mathsf{S}_i \coloneqq \mathsf{S}_{\sigma(i)}, \quad \sigma \cdot \mathsf{CZ}_{i,j} \coloneqq \mathsf{CZ}_{\sigma(i),\sigma(j)}. \end{aligned} \tag{137}$$

After relabelling, we write each CZ pair with the smaller index first. These actions induce an action on distributions over generators by pushforward:

$$(\sigma \cdot \mu)(G) \coloneqq \mu(\sigma^{-1} \cdot G) \quad \text{for} \quad \mu \in \mathcal{P}(\mathcal{A}). \tag{138}$$

For either matrix group, the transition function is equivariant and the reward function is invariant:

$$\mathcal{T}(\sigma \cdot M, \sigma \cdot G) = \sigma \cdot \mathcal{T}(M, G), \quad \mathcal{R}(\sigma \cdot M, \sigma \cdot G) = \mathcal{R}(M, G). \tag{139}$$

The following result is a standard consequence of finite-MDP symmetry [121, 122]. It shows that we can impose equivariance on the agent's policy without ruling out an optimal policy.

Proposition 53 *For either the CNOT or Clifford synthesis MDP with discount factor $\gamma \in [0, 1)$, there exists an optimal stationary stochastic Markov policy $\pi^* : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ whose corresponding optimal value function $V^* : \mathcal{S} \to \mathbb{R}$ satisfies*

$$\pi^*(\sigma \cdot M) = \sigma \cdot \pi^*(M), \quad V^*(\sigma \cdot M) = V^*(M). \tag{140}$$

The proof is given in Section C.1.1.

6.6 Size-agnostic equivariant neural architectures

A neural synthesiser should ideally be *size-agnostic*, so that a single set of learned parameters can be reused across problem sizes. A familiar example of a size-agnostic architecture is a convolutional neural network, whose shared filters can be applied to images of different dimensions without changing its learned parameters.

Existing neural synthesis methods often train models for a particular device architecture, qubit count, or target tableau. For example, Zen et al. [123] and Doherty et al. [124] both train separate agents for individual stabiliser codes. Kremer et al. [125] likewise train separate synthesis models for each circuit class, width, and connectivity graph. Romanello et al. [126] do reuse an eight-qubit CNOT agent at other widths. This reuses a fixed-width model by padding smaller circuits or splitting larger circuits into eight-qubit sections, rather than making the model itself size-agnostic.

This raises a natural question: can a single network be trained once and then applied directly across qubit counts, without retraining or reducing each input to a fixed width?

To construct policies that are both size-agnostic and equivariant, we begin by associating each qubit i with an embedding $h_i^{(0)} \in \mathbb{R}^h$. These embeddings are updated by permutation-equivariant neural layers whose parameters do not depend on the number of qubits. Relabelling the qubits then simply permutes the embeddings, so the final features can be read out as equivariant logits for gates on individual qubits or pairs of qubits. Symmetric pooling of the embeddings gives an invariant value estimate.

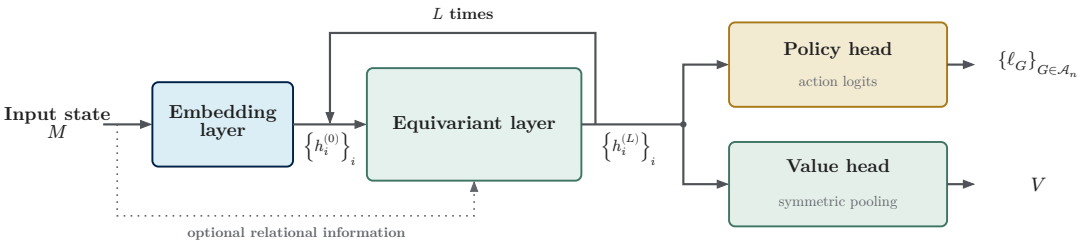


Figure 32: General form of the size-agnostic equivariant actor–critic architectures used below.

Figure 32 shows the general construction, implemented for Clifford and CNOT synthesis in Figure 33 and Figure 34, respectively. Their layers draw on Deep Sets [127], Set Transformers [128], relation-aware Transformers [129], and relational graph

neural networks [130]. The Clifford experiments use RelTransformer and RelGNN as their principal variants, while the CNOT experiments use BiRel; the Clifford ablation additionally tests an ordinary Transformer, an equivariant MLP, and an unconstrained FlatMLP. Definitions of these ingredients are given in Section C.2. Their cross-size generalisation is tested in Section 6.10 and Section 6.9.

For either circuit family, the actor produces one logit $\ell_{\theta,G}(M)$ for every valid generator $G \in \mathcal{A}_n$. The logits define the stochastic policy

$$\pi_{\theta,n}(G \mid M) := \frac{\exp\big(\ell_{\theta,G}(M)\big)}{\sum_{G' \in \mathcal{A}_n} \exp\big(\ell_{\theta,G'}(M)\big)}.$$

The critic independently returns a scalar $V_\theta(M)$ estimating the expected discounted return. Because softmax commutes with permutations, relabelling the action logits permutes the corresponding action probabilities, while symmetric pooling leaves the value estimate unchanged.

6.6.1 *Clifford policy*

Input. Recall that each row of the symplectic matrix M records the phase-free image of one Pauli generator under Clifford conjugation: the first n rows record the images of X_i, and the final n rows record the images of Z_i. Grouping the rows and columns by qubit arranges M into an $n \times n$ grid of 2×2 blocks. The directed effect of qubit i on qubit j is captured by four bits: the X_j and Z_j components of the image of X_i, and the X_j and Z_j components of the image of Z_i. More explicitly, the corresponding descriptor and learned pair feature are

$$d_{ij}(M) := \big(M_{ij}, M_{i,n+j}, M_{n+i,j}, M_{n+i,n+j}\big),$$
$$r_{ij} := \mathrm{Embed}\big(d_{ij}(M), \mathbb{1}(i = j)\big).$$

At the end of the reduction, diagonal blocks must equal I_2, while off-diagonal blocks must be zero. The diagonal flag $\mathbb{1}(i = j)$ tells the shared embedding map which case applies. We then initialise each qubit embedding using symmetric row and column summaries:

$$h_i^{(0)} := \varphi_0\big(\mathrm{mean}_j\, r_{ij}, \max_j r_{ij}, \mathrm{mean}_j\, r_{ji}, \max_j r_{ji}, r_{ii}, \eta_i\big),$$

Here η_i collects equivariant rank-based count features, such as the fractions of non-zero, rank-one, and rank-two blocks in block row and column i.

Equivariant layers. Starting from the initial qubit embeddings, we apply L permutation-equivariant updates. Any equivariant set architecture can be used here; for example, an ordinary Transformer without positional encodings is permutation-equivariant. We consider two equivariant relational variants designed to improve information flow through the qubit embeddings. Both use feature reinjection: the

pair features r_{ij} are supplied at every layer, rather than only during initialisation. This gives later layers access to pairwise information that may be lost when the features are pooled into qubit embeddings. Schematically, one such update is

$$h_i^{(\ell+1)} := \Phi_\ell\Big(h_i^{(\ell)}, \text{aggregate}_j\Big[\Psi_\ell\Big(h_i^{(\ell)}, h_j^{(\ell)}, r_{ij}, r_{ji}\Big)\Big], \eta_i\Big).$$

RelTransformer uses relation-aware attention, with r_{ij} conditioning the attention weights and values, while RelGNN uses r_{ij} and r_{ji} for edge-conditioned message passing. Because the maps are shared across qubits and the aggregation is symmetric, relabelling the qubits simply relabels the updated embeddings. Definitions of the RelTransformer and RelGNN layers are given in Section C.2, and their shared data flow is shown in Figure 33.

Readout. After L layers, shared heads score each one-qubit gate using the corresponding qubit embedding, while a symmetric head scores each CZ gate using the corresponding pair of qubit embeddings. Writing $h_i := h_i^{(L)}$ and letting g be an invariant global summary,

$$\begin{aligned} g &:= \text{pool}_i(h_i, \eta_i), \\ \ell_i^{\mathsf{H}} &:= \rho_{\mathsf{H}}(h_i, r_{ii}, g), \quad \ell_i^{\mathsf{S}} := \rho_{\mathsf{S}}(h_i, r_{ii}, g), \\ \ell_{ij}^{\mathsf{CZ}} &:= \rho_{\mathsf{CZ}}\big(h_i + h_j, h_i \odot h_j, |h_i - h_j|, r_{ij} + r_{ji}, r_{ij} \odot r_{ji}, g\big) = \ell_{ji}^{\mathsf{CZ}}, \quad i < j, \\ V_\theta(M) &:= \rho_V(g). \end{aligned}$$

The sums, elementwise products, and absolute difference supplied to the CZ head are unchanged when i and j are exchanged. We use this deliberately broad set of symmetric features, although some may be redundant.

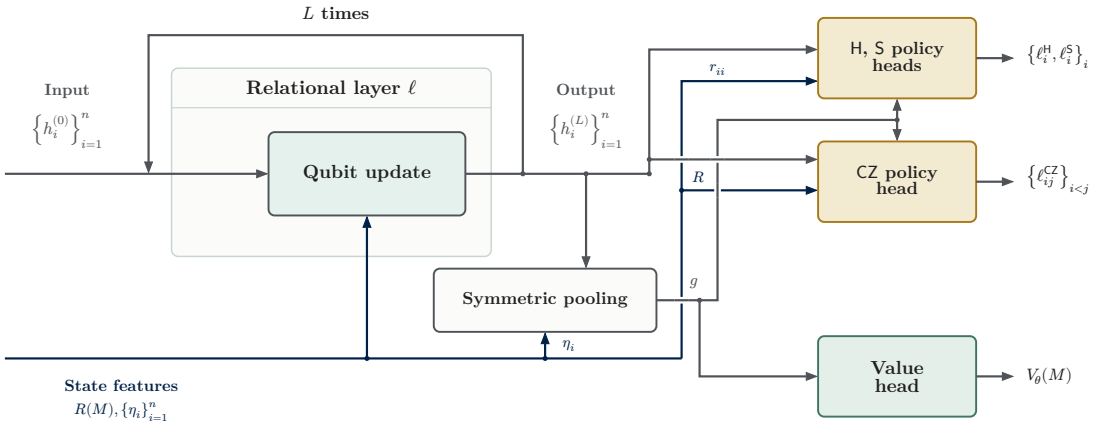


Figure 33: Shared Clifford architecture for RelTransformer and RelGNN, using relation-aware attention and message passing, respectively.

Table 6 later compares these relational models with simpler controls.

6.6.2 *CNOT policy*

We first tested RelTransformer and RelGNN architectures, both of which learned effective policies for this problem. However, the source and target qubits of a CNOT gate are not interchangeable in the matrix update, so the policy must be able to assign different scores to $\mathsf{CNOT}_{i,j}$ and $\mathsf{CNOT}_{j,i}$. We found that the model benefited from preserving this distinction throughout the network by maintaining two embeddings for each qubit: one for its role as source and one for its role as target. This led us to the BiRel layer, which we now describe.

Input. For each ordered pair of qubits (i, j), the two directed matrix entries and a diagonal flag are packed into a single lookup index:

$$c_{ij} := M_{ij} + 2M_{ji} + 4\mathbb{1}(i = j), \quad r_{ij} := \mathrm{Embed}\big(c_{ij}\big).$$

Pooling these pair features by row and column initialises two embeddings for each qubit: $s_i^{(0)}$ for the source stream and $t_i^{(0)}$ for the target stream.

Equivariant layer. A BiRel layer maintains two embeddings for every qubit: a source embedding $s_i^{(\ell)}$ and a target embedding $t_i^{(\ell)}$. It updates the source stream using relation-conditioned aggregates of the target stream, and vice versa. These names follow the gate convention: in $\mathsf{CNOT}_{i,j}$, i is the source (control) and j is the target. Because the generator acts on the right, the corresponding matrix reduction sends $C_i \leftarrow C_i \oplus C_j$: the source-indexed column is therefore updated using the target-indexed column.
Schematically, one layer may be written

$$\mu_i^{(\ell),\mathrm{src}} := \mathrm{pool}_{j\neq i}\, \varphi_{\mathrm{src}}^{(\ell)}\Big(s_i^{(\ell)}, t_j^{(\ell)}, r_{ij}, r_{ji}\Big), \quad s_i^{(\ell+1)} := \psi_{\mathrm{src}}^{(\ell)}\Big(s_i^{(\ell)}, \mu_i^{(\ell),\mathrm{src}}\Big),$$

$$\mu_i^{(\ell),\mathrm{tgt}} := \mathrm{pool}_{j\neq i}\, \varphi_{\mathrm{tgt}}^{(\ell)}\Big(s_j^{(\ell)}, t_i^{(\ell)}, r_{ji}, r_{ij}\Big), \quad t_i^{(\ell+1)} := \psi_{\mathrm{tgt}}^{(\ell)}\Big(t_i^{(\ell)}, \mu_i^{(\ell),\mathrm{tgt}}\Big).$$

Readout. After L layers, a nonsymmetric shared head scores each ordered generator:

$$g := \mathrm{pool}_i\Big(s_i^{(L)}, t_i^{(L)}\Big),$$

$$\ell_{ij}^{\mathsf{CNOT}} := \rho_{\mathsf{CNOT}}\Big(s_i^{(L)}, t_j^{(L)}, r_{ij}, r_{ji}, g\Big), \quad i \neq j,$$

$$V_\theta(M) := \rho_V(g).$$

Thus $\ell_{ij}^{\mathsf{CNOT}}$ and $\ell_{ji}^{\mathsf{CNOT}}$ may differ, while relabelling the qubits still permutes the ordered logits equivariantly. Symmetric pooling of both streams gives the invariant value estimate. The complete data flow is shown in Figure 34.

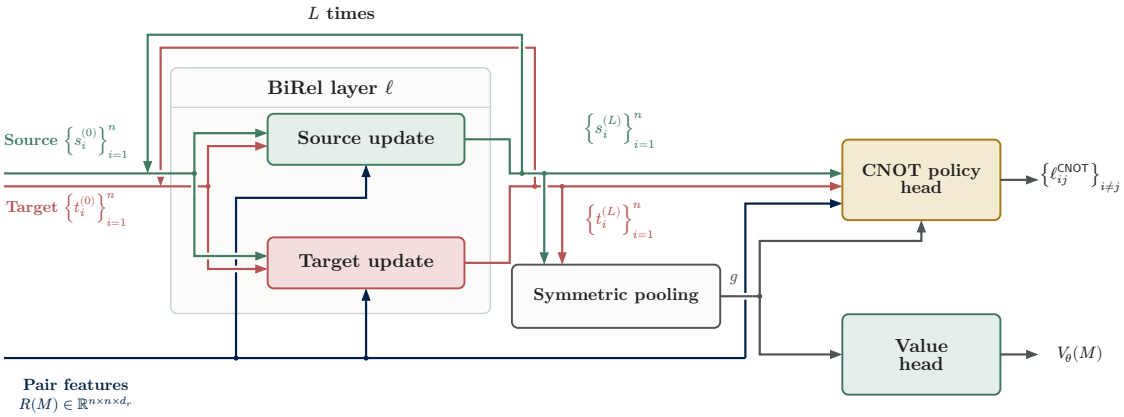


Figure 34: BiRel architecture for CNOT synthesis. The coupled source and target updates are stacked across L layers. Pair features condition every layer and the policy head; symmetric pooling supplies the global summary g to both heads.

Because the network parameters do not depend on the number of qubits, the same trained CNOT policy can be applied directly at different widths. Table 3 shows its transfer from training on up to 20 qubits to targets of up to 30 qubits.

Further architectural details are given in the corresponding papers [4] and [3]. The training configurations used for the principal reported experiments are summarised in Section 6.8.1.

6.7 Curriculum

So far we have not defined how the initial target for each training episode is chosen. Starting at a uniformly sampled group element, an untrained, randomly initialised policy is very unlikely to reach the identity within the step limit. Apart from the Hamming-distance shaping term used for Clifford synthesis, intermediate rewards penalise gate use without indicating which actions move the state closer to the identity. Successful trajectories therefore provide an important learning signal, but would be extremely rare if training used only uniformly sampled targets.

We therefore use curriculum learning [131, 132], inspired by reverse curriculum generation for sparse-reward goal-reaching problems [133] and DeepCubeA's approach of generating training instances by working backwards from the goal in combinatorial puzzles [134]. Kremer et al. [125] apply a related approach to constrained Clifford synthesis.

6.7.1 *Two-dimensional random-walk curriculum*

We sample elements from both matrix groups using the following random-walk method. Write I for the identity element of $\mathcal{G}_n$. For $n \in \mathbb{N}$ and $d \geq 0$, define $\mathcal{P}_{n,d}$ to be the distribution over $\mathcal{G}_n$ obtained by drawing

$$L = \lfloor d \rfloor + B, \quad \text{where} \quad B \sim \text{Bernoulli}(d - \lfloor d \rfloor). \tag{141}$$

then sampling $G_1, ..., G_L$ independently and uniformly from $\mathcal{A}_n$ and setting

$$M^{(0)} = I, \quad M^{(k)} = M^{(k-1)} G_k, \quad M_{\text{target}} = M^{(L)}.$$

Reversing the sampled sequence gives a valid reduction, so every target drawn from $\mathcal{P}_{n,d}$ can be solved within L steps. Although d is not the target's exact Cayley-graph distance, it is the expected sampled walk length and therefore serves as a proxy for target difficulty. The Bernoulli interpolation allows finer adjustments to the expected walk length, particularly for short walks, while preserving $\mathbb{E}[L] = d$ and recovering a fixed-length walk whenever d is an integer.

The training curriculum has two axes: the qubit count n and the expected random-walk length d. At a fixed qubit count, we increase d once rollout success reaches 90% for CNOT synthesis or 100% for Clifford synthesis. We increase the qubit count after a fixed number of training steps or once a prescribed difficulty is reached, depending on the experiment. The size-agnostic architecture lets us continue training at the larger width directly from the preceding checkpoint, reusing the learned parameters unchanged.

6.7.2 *Random-walk limits*

The following theorems show that the random walks used in our experiments converge to the uniform distribution, consistent with our aim of training a synthesiser that can handle any group element. We use the finite-state Markov-chain ergodic theorem: an irreducible, aperiodic chain converges to its unique stationary distribution. Here $\mathcal{A}_n$ generates $\mathcal{G}_n$, so each walk is irreducible, while right multiplication permutes $\mathcal{G}_n$, so the uniform distribution is stationary. It remains only to establish aperiodicity.

> Theorem 54 *(CNOT random-walk limit.) For $n \geq 3$, the walk on $\mathrm{GL}(n, \mathbb{F}_2)$ that right-multiplies by a uniformly sampled generator from $\mathcal{A}_n^{\mathrm{CNOT}}$ converges from the identity to the uniform distribution. For $n = 2$, the corresponding fixed-length walk has period two and does not converge.*

The proof is given in Section C.1.2.

> Theorem 55 *(Clifford random-walk limit.) For $n \geq 3$, the walk on $\mathrm{Sp}(2n, \mathbb{F}_2)$ that right-multiplies by a uniformly sampled generator from $\{\mathsf{H}_i, \mathsf{S}_i, \mathsf{CZ}_{i,j}\}$ converges from the identity to the uniform distribution. For $n = 1$ and $n = 2$, the corresponding fixed-length walk has period two and does not converge.*

The proof is given in Section C.1.3.

6.8 Training and evaluation methodology

6.8.1 *Training*

We jointly train the policy and value networks using proximal policy optimisation (PPO), a standard policy-gradient algorithm [135]. PPO samples trajectories from

the current policy and uses them to estimate advantages $\hat{A}_t$, which measure whether an action's return was better or worse than the value network predicted. The policy update therefore increases the probability of actions with positive estimated advantage and decreases that of actions with negative estimated advantage. PPO compares the updated policy with the policy that generated the trajectories through the probability ratio

$$r_t(\theta) := \frac{\pi_\theta(A_t \mid S_t)}{\pi_{\theta_{\text{old}}}(A_t \mid S_t)}, \quad L^{\text{clip}}(\theta) := \mathbb{E}_t\Big[\min\Big(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\Big)\Big].$$

Maximising this clipped surrogate objective limits the benefit of moving the ratio beyond $[1-\varepsilon, 1+\varepsilon]$, which discourages excessively large policy updates. We use the PPO implementation provided by PufferLib [136]. The training settings and checkpoint choices for the Clifford RelGNN, Clifford RelTransformer and CNOT BiRel experiments are given in Section C.3.

6.8.2 *Policy-guided decoding*

Policy-guided reduction uses two action-selection rules. During PPO training, actions are sampled from the policy. We initially used separate greedy evaluations to decide when to advance the curriculum. In later experiments, the sampled training rollouts already achieved 100% success at the advancement point, so we used their success rate directly and avoided the additional evaluation. For the main evaluations, we use *greedy decoding*, applying the generator with the highest policy probability at each step until reaching the identity or the step limit. The exception is the exact six-qubit recovery experiment, which begins with a greedy pass and then samples trajectories at several temperatures under a fixed compute budget, retaining the best successful circuit. A successful reduction gives a circuit for the target using the circuit-order conventions described in Section 6.2.

For Clifford synthesis, we also use the inverse-tableau trick of Gidney [137]: we decode both M_{target} and M_{target}^{-1}, convert the latter solution back to a circuit for M_{target}, and keep the shorter successful circuit. The CNOT experiments do not use this comparison. For larger Clifford instances, we avoid revisiting a tableau whenever an alternative action is available. We verify every trajectory and accept it as a valid synthesis only if it reaches the identity.

The detailed candidate-ranking, target-representation, and search protocols are recorded in Section C.4.

6.8.3 *Benchmarks and baselines*

We evaluate the learned policies as practical synthesis heuristics on finite random-walk targets and uniformly sampled group elements. For both CNOT and Clifford synthesis, we begin at small qubit counts where exact optima are known, then compare with scalable heuristics once exhaustive search becomes infeasible.

We begin with length-n^2 random walks in Table 3 and Table 4. Theorem 54 and Theorem 55 show that these walks converge to the uniform distribution; Figure 35 and Figure 36 evaluate the policies directly on uniformly sampled targets. Table 6 compares the Clifford architectures on uniform six-qubit targets.

For small instances, we compare the resulting gate counts with certified optima. We use the exact CNOT solver of Christensen et al. [115] through seven qubits and our own meet-in-the-middle Clifford solver [4] through six qubits. Breadth-first search and SAT [138] could provide equivalent exact references.

For larger instances, we compare against the scalable methods introduced in Section 6.3. We use Qiskit's `synth_clifford_layers` implementation for the Bravyi–Maslov baseline. The Qiskit greedy baseline excludes the template and symbolic-Pauli optimisations of Bravyi et al. [112], [113]. We also compare with AlphaCNOT [119] where comparable results are available.

The Qiskit baselines use the same two-qubit-cost convention: CZ and CX each count as one entangling gate, while SWAP counts as three.

6.9 CNOT synthesis results

The CNOT experiments minimise CNOT count. The finite-walk experiments evaluate 100 targets generated by random walks of length n^2 at each qubit count.

The exact CNOT references are generated with the optimal CNOT synthesis solver of Christensen et al. [115], which certifies minimum-size circuits through $n = 7$ qubits. AlphaCNOT reports aggregate results but does not release the synthesised circuits behind them, so its published results cannot be compared with ours target by target. We therefore rerun its method on our targets from five qubits onwards.

n	Optimum	AlphaCNOT	Our policies				Scalable baselines		
			Checkpoint	Δ_{opt}	Final model	Δ_{opt}	AECM	GGE	PMH
4	5.35	–	**5.35**	0.00	5.95	+0.60	5.44	6.58	6.65
5	7.70	7.75	**7.71**	+0.01	8.42	+0.72	8.07	9.88	10.40
6	10.81	11.17	**11.10**	+0.29	12.22	+1.41	12.03	14.76	15.46
7	13.50	14.88	**14.23**	+0.73	16.00	+2.50	15.65	19.56	20.99
8	–	21.38	**18.42**	–	19.83	–	19.98	25.21	27.56
10	–	–	**27.76**	–	28.79	–	30.24	38.47	41.73
12	–	–	**37.90**	–	39.44	–	42.63	53.88	60.22
14	–	–	**50.95**	–	51.95	–	57.41	71.20	81.19
16	–	–	**65.28**	–	65.68	–	74.52	90.17	104.36
18	–	–	**80.86**	–	80.98	–	94.10	111.39	128.61
20	–	–	**97.78**	–	**97.78**	–	114.08	135.36	159.21
25	–	–	–	–	**150.63**	–	181.35	201.51	241.94
30	–	–	–	–	**221.12**	–	263.16	278.89	334.34

Table 3: Mean CNOT counts over 100 length-n^2 random-walk targets at each qubit count. **Checkpoint** denotes the model obtained after training the curriculum up to that row's qubit count; training ended at 20 qubits. **Final model** is the 20-qubit checkpoint transferred without further training. Each Δ_{opt} is the adjacent policy count minus the optimum. Boldface marks the best learned policy in each row; lower is better.

6.9.1 *Exact-reference regime*

The four-qubit curriculum checkpoint matches the certified mean optimum. Through seven qubits, each checkpoint remains within 0.73 CNOTs of the certified mean optimum at its corresponding qubit count. The final 20-qubit model also solves every target, although its performance at smaller qubit counts deteriorates as curriculum training progresses to larger qubit counts.

From five qubits onwards, our equivariant, size-agnostic policy matches or improves on AlphaCNOT-100-mix using a single greedy decode. AlphaCNOT uses Monte Carlo tree search with 50 simulations per move and keeps the best of 100 stochastic attempts per target. Our method uses no tree search during either training or inference: evaluation simply selects the highest-logit gate at each step.

6.9.2 *Transfer beyond the training range*

Trained on at most 20 qubits, the policy generalises without further training to 30 qubits, the largest size tested. At 30 qubits, it solves every instance and uses 221.12 CNOTs on average, compared with 278.89 for GGE and 334.34 for PMH. At qubit counts seen during training, the final CNOT model uses at most 1.77 more gates on average than the corresponding earlier checkpoint. Curiously, the Clifford policy shows the opposite behaviour on seven-qubit targets: extending training to ten qubits reduces its mean CZ count from 11.03 to 10.65 (Section 6.10.1). These results are specific to the stated benchmark.

6.9.3 *Uniform-target generalisation*

We next test elements sampled uniformly from $\mathrm{GL}(n, \mathbb{F}_2)$ rather than from finite-length random walks. By Theorem 54, the uniform distribution is the limit of these walks. Exact optima are unavailable at these qubit counts, so we compare solution quality against scalable baselines.

The model succeeds on every uniformly sampled $\mathrm{GL}(n, \mathbb{F}_2)$ target through 30 qubits. From 10 qubits onwards, it outperforms AECM, GGE, and PMH, including beyond its training range.

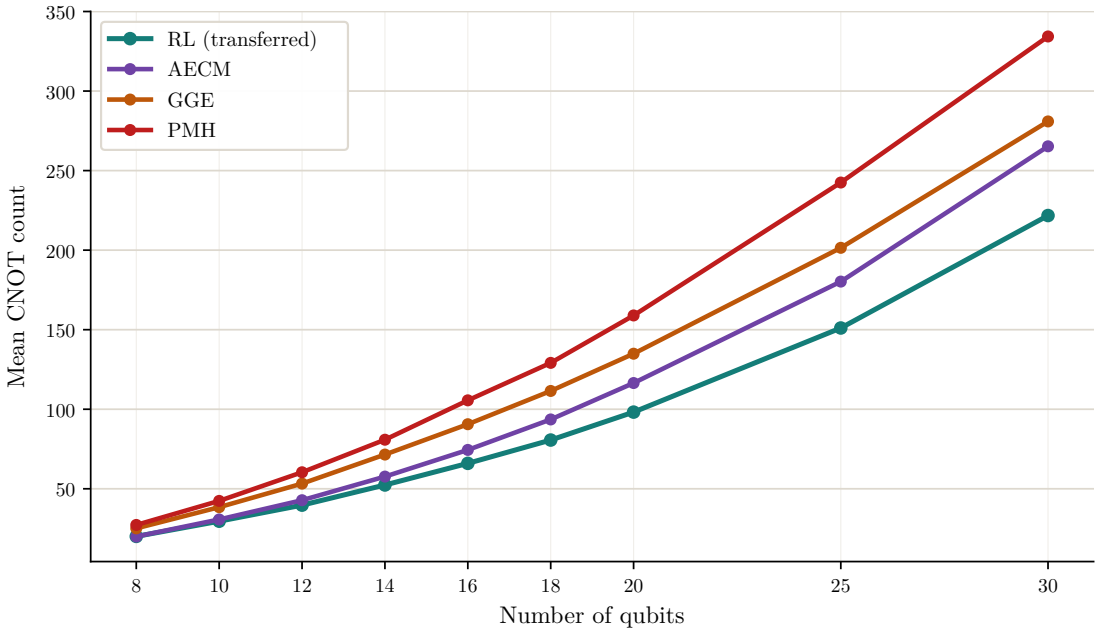


Figure 35: Mean CNOT counts for the transferred policy, AECM [111], GGE, and PMH [23] on 100 uniformly sampled $\mathrm{GL}(n, \mathbb{F}_2)$ targets at each qubit count [4].

At 30 qubits, the transferred policy uses 221.75 CNOTs on average, compared with 265.26 for AECM, 280.90 for GGE, and 334.31 for PMH. Its advantage therefore persists under uniform sampling from the full group.

6.10 Clifford synthesis results

The Clifford experiments minimise two-qubit gate count, reported as the number of CZ gates. We report two sets of experiments. The RelTransformer is evaluated on target tableaus sampled by length-n^2 random walks, measuring mean circuit quality and transfer across qubit counts [4]. The RelGNN is evaluated on uniformly sampled targets to measure circuit quality and reliability, and on the 1003-instance six-qubit benchmark of Bravyi et al. [112] to test recovery of certified optima [3]. The six-qubit ablation in Table 6 directly compares the two architectures and shows that they achieve comparable performance.

Kremer et al. [125] also use reinforcement learning for Clifford synthesis, but train models for devices with restricted connectivity. Their synthesis problem is therefore more constrained than ours, so we do not include their gate counts in the quantitative comparisons.

6.10.1 *RelTransformer: mean cost and transfer across qubit counts*

At each qubit count, the RelTransformer experiments evaluate 100 targets sampled by length-n^2 random walks. We implement a meet-in-the-middle solver which certifies the optimum through six qubits, beyond which its time and memory requirements become prohibitive.

n	Optimum	Our policies				Scalable baselines		
		Checkpoint	Δ_{opt}	Final model	Δ_{opt}	Qiskit greedy	B–M	A–G
4	3.03	–	–	**3.03**	0.00	3.48	4.12	6.78
5	5.07	–	–	**5.10**	+0.03	6.30	8.42	11.65
6	7.47	7.73	+0.26	**7.68**	+0.21	10.48	13.26	20.24
7	–	11.03	–	**10.65**	–	16.33	20.49	29.13
8	–	–	–	**13.86**	–	22.65	28.48	39.05
10	–	**22.24**	–	**22.24**	–	37.03	48.21	66.79
12	–	–	–	**32.57**	–	55.78	79.16	102.27
20	–	–	–	**106.46**	–	171.83	295.21	318.85
30	–	–	–	**296.71**	–	397.44	793.79	747.56

Table 4: Mean CZ counts over 100 length-n^2 random-walk targets at each qubit count [4]. **Checkpoint** is shown where a model was trained up to that row's qubit count: six, seven, or ten qubits. **Final model** uses the later checkpoint from the six-qubit curriculum in the exact-reference rows and the ten-qubit checkpoint in the transfer rows, so the entries coincide at ten qubits. B–M denotes Bravyi–Maslov [114]. Each Δ_{opt} is the adjacent policy count minus the optimum. Boldface marks the best learned policy in each row; lower is better.

After training on six-qubit examples, the model uses an average of only 0.26 CZ gates more than the certified optimum on six-qubit targets. A later checkpoint from continued training on the same six-qubit curriculum matches the mean optimum at four qubits and exceeds it by only 0.03 and 0.21 CZ gates at five and six qubits. Every reported policy result in this regime improves on the scalable baselines.

After extending the curriculum from six- to ten-qubit examples, the model solves all 100 held-out ten-qubit targets with a mean of 22.24 CZ gates. Without further training or reparametrisation, the same checkpoint can be applied at every larger qubit count. It also retains its performance at smaller qubit counts: on the seven-qubit batch, it averages 10.65 CZ gates, compared with 11.03 for the checkpoint saved after training up to seven qubits.

At 30 qubits, the transferred policy uses 296.71 CZ gates on average, compared with 397.44 for Qiskit greedy, 747.56 for Aaronson–Gottesman, and 793.79 for Bravyi–Maslov. The same checkpoint thus improves on every listed scalable baseline at three times the largest qubit count used in training. Results for uniformly sampled targets at lower qubit counts are given in Section C.6.

6.10.2 *RelGNN: exact six-qubit recovery*

Since the preceding experiments show an average gap of less than one CZ gate from the optimum, we next test whether policy-guided search can recover the certified optimum for individual targets. Bravyi et al. [112] studied a benchmark of 1003 six-qubit tableaus with known optimal CZ counts, drawn from the exhaustive database of Bravyi, Latone, and Maslov [109]. We emphasise that the agent is trained only on targets sampled by the curriculum described in Section 6.7.1, without access to these tableaus or their optimal solutions. Accidental overlap is unlikely: even restricting to six-qubit Clifford operators with optimal CZ count five leaves about 1.3×10^{14} possibilities [109].

The RelGNN policy returns a valid circuit for every instance and recovers 995 of the 1003 optima (99.2%); each of the remaining eight circuits uses one additional CZ gate. A single greedy pass over the full suite takes 0.83 seconds and recovers 507 optima. Brief policy-guided search raises this to 610 optima in 21 seconds, with every circuit then within one CZ gate of optimal.

Decoder stage	Optima	Elapsed	Maximum gap	Mean gap
Greedy pass	507/1003	0.83 s	3	0.692
Brief search	610/1003	21 s	1	0.392
Extended search	982/1003	21.6 min	1	0.021
Extended search	995/1003	183.1 min	1	0.008

Table 5: Decoding milestones on the 1003-instance exact six-qubit benchmark [3]. Gaps are learned CZ count minus the certified optimum, with means taken over all 1003 instances. Elapsed times cover the full suite; extended-search values record the first time each recovery level is reached during a separate whole-suite search.

For comparison, the symbolic peephole optimiser of Bravyi et al. [112] recovered 982 of the 1003 optima after 217 hours and did not recover the remaining 21 after a further 576 hours [139]. Although wall-clock timings are not directly comparable because the experiments ran on different hardware, our search recovered 982 optima in 21.6 minutes and 995 in 183.1 minutes. This recovery experiment is separate from the 100-target mean-cost experiment above; Section C.4.1 gives the complete rollout schedules, timing definitions, reduction counts, and hardware details.

6.10.3 *RelGNN: uniform-target generalisation*

We evaluate the RelGNN checkpoint trained up to ten qubits on 100 uniformly sampled phase-free Clifford tableaus at each qubit count. Among the targets it solves, the policy produces lower mean CZ counts than both Qiskit baselines throughout the displayed range, but its solve rate falls at the largest qubit counts. The dotted curves show the Qiskit means on the subset of targets successfully solved by the model.

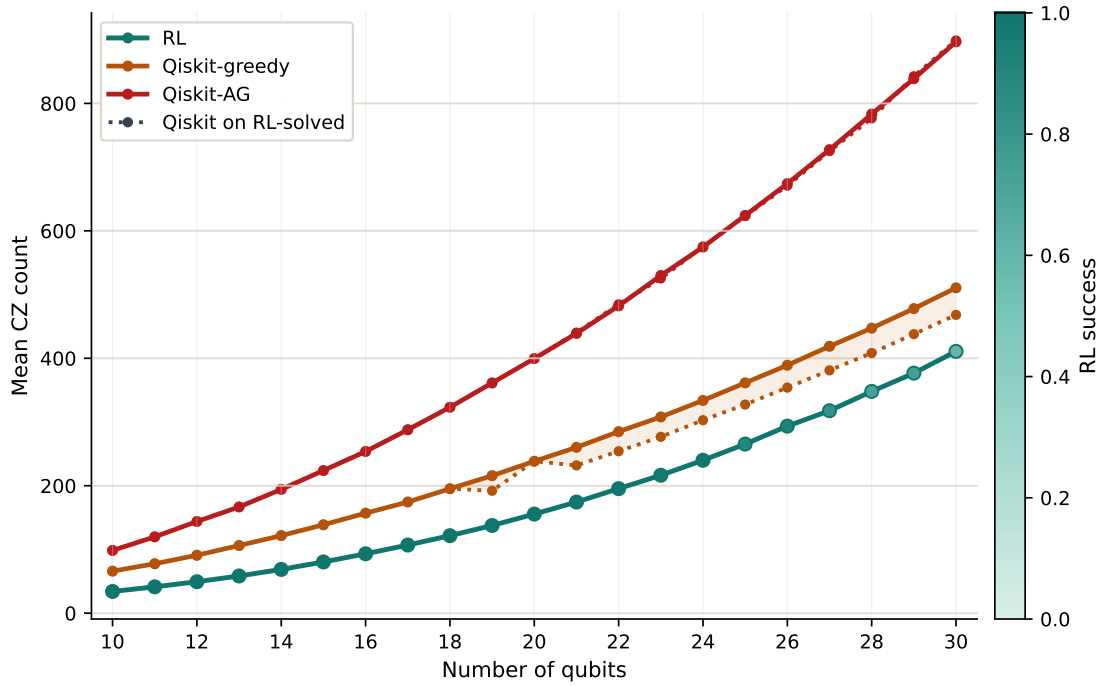


Figure 36: Uniform-target CZ-count comparison for the ten-qubit-trained RelGNN checkpoint under the $6n^2$ rollout budget with the no-loop safeguard. The benchmark contains 100 uniformly sampled phase-free Clifford tableaus at each qubit count. Learned means are conditional on successful synthesis, and marker colour gives the solve rate. Solid curves show Qiskit's mean CZ counts over all targets, and dotted curves restrict these means to the targets solved by the learned policy [3].

At 30 qubits, the checkpoint solves 59 of the 100 targets, using 410.8 CZ gates on average. Both Qiskit baselines use substantially more CZ gates on these same 59 targets, as shown by the dotted curves.

6.10.4 *Architecture ablation*

Finally, we test whether the relational architecture itself accounts for the improvement on uniformly sampled six-qubit targets. **Transformer** uses ordinary self-attention [102], **MLP** removes communication between qubits, and **FlatMLP** applies an unconstrained MLP to the flattened tableau.

Policy family	CZ count
RelGNN	13.49 ± 0.01
RelTransformer	13.47 ± 0.06
Transformer	14.12 ± 0.28
MLP	15.88 ± 0.05
FlatMLP	15.19 ± 0.16

Table 6: Architecture ablation on 100 uniformly sampled six-qubit Clifford targets. Mean CZ counts under deterministic no-loop greedy decoding are reported as mean ± standard deviation over three training seeds; all models solve every target [3].

RelGNN and RelTransformer are essentially tied and outperform the non-relational controls. This suggests that explicitly incorporating pairwise tableau information

matters more than the choice between attention and message passing. Evaluation details are recorded in Section C.4.3.

6.11 Circuit extraction

The matrix-group synthesis framework also provides a circuit-extraction procedure for reduced ZX-diagrams. We begin with the simpler CNOT case, in which the parity map is read directly from the diagram, before generalising to Clifford circuits using tableaus obtained from Pauli webs.

6.11.1 *CNOT GSLC extraction*

The parity map is the biadjacency matrix between the two vertex sets of the CNOT GSLC normal form introduced in Theorem 15. In Figure 37, rows x_i and columns y_j index these sets, and a black marker records each connection. Matrix reduction and graphical extraction then proceed in lockstep: each column addition to the parity grid corresponds to extracting a CNOT onto the output boundary.

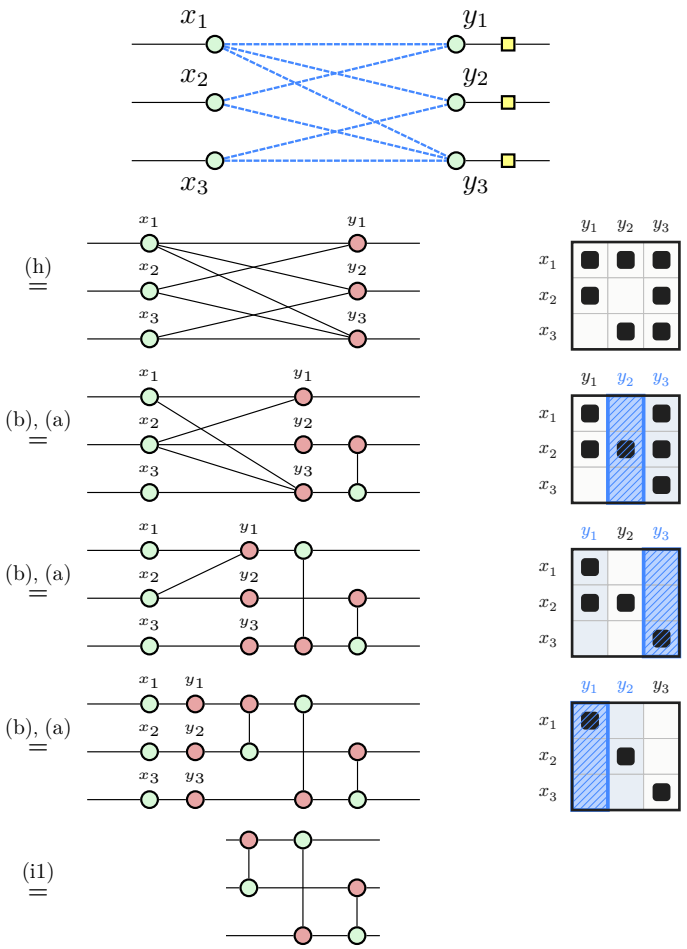


Figure 37: Extraction of a three-qubit circuit from a CNOT GSLC diagram.

6.11.2 *Clifford GSLC extraction*

The phase-free operation tableau can be obtained from a Clifford GSLC diagram using the Pauli webs introduced in Section 3.7. Edges between the y_j vertices represent CZ gates, while the Clifford phases on those vertices represent powers of S. In Figure 38, tableau reduction and graphical extraction proceed in tandem: each right multiplication extracts the corresponding Clifford gate onto the output boundary.

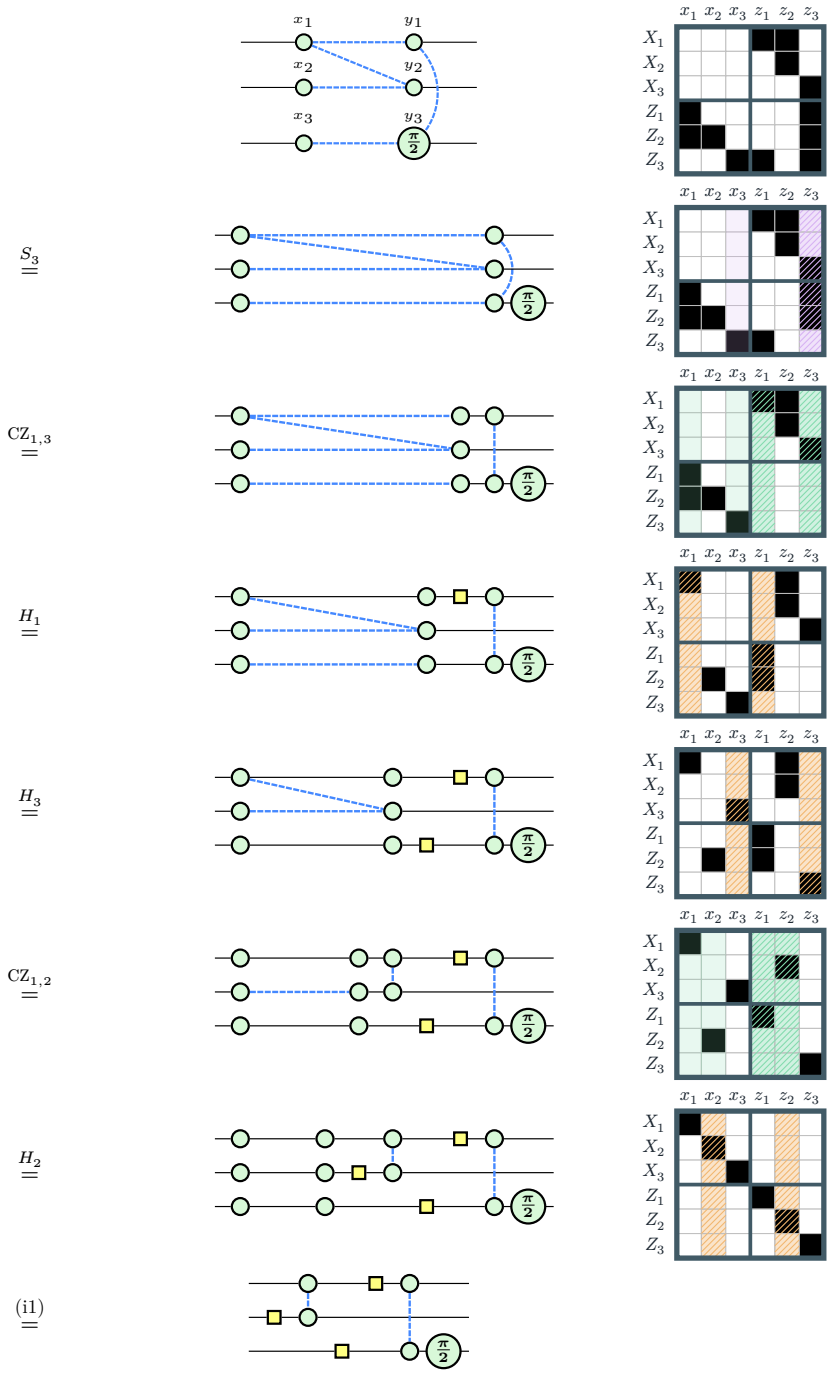


Figure 38: Extraction of a three-qubit Clifford circuit from a Clifford GSLC diagram.

The Clifford policy can also support circuit extraction from graph-like ZX-diagrams through the ZX-flow framework of Kissinger and van de Wetering [74]. Ordering the phase gadgets consistently with the diagram's flow gives phase-gadget layers interleaved with Clifford skeletons. Each skeleton can then be converted to a tableau and resynthesised by the learned policy.

6.12 Discussion and limitations

This chapter presented reusable neural policies for CNOT and Clifford circuit synthesis. On the reported random-walk benchmarks, they achieve state-of-the-art results through 30 qubits. A single model for each problem can be applied to unseen targets and qubit counts without retraining. This is achieved using custom equivariant relational architectures that reflect the matrix structure of the state and the directed or symmetric gate actions.

The learned Clifford policy also recovers 995 of the 1003 certified optima in the exact six-qubit benchmark, with each of the remaining eight circuits one CZ gate above optimum. These references came from an exhaustive database that required substantial computational effort to construct and is not fully publicly available. Policy-guided search thus recovers almost all of these optima without constructing an exhaustive database.

Empirically, the learned policies give the best results among the methods evaluated on the stated benchmarks. Beyond the widths for which exact references are available, however, we cannot certify that a returned circuit is optimal, even when it happens to be so. We also cannot guarantee that a policy rollout reaches the identity. If a rollout reaches the step limit, a deterministic synthesis algorithm can complete the reduction of the residual tableau. This guarantees a valid circuit even when the learned policy does not reach the identity on its own.

Scalability presents a further limitation. Each policy step processes $O(n^2)$ pair features and candidate gates, so even a rollout of linear length costs $O(n^3)$; longer rollouts cost more. A factorised policy could avoid scoring every pair jointly, while hierarchical multi-gate actions could reduce the number of policy steps.

Finally, the architectures and two-dimensional curricula may transfer to related synthesis problems. Future work on circuit extraction could integrate the synthesisers with PyZX and, more ambitiously, train an agent directly on a Pauli-dependency DAG or Pauli semiweb. Such an agent could jointly choose the phase-gadget order and optimise the intervening Clifford regions, targeting the two-qubit count of the whole circuit rather than each region independently.

Chapter VII

Conclusions

This thesis studies the compilation of quantum circuits using tools from the ZX-calculus and machine learning, with the aim of reducing the resources required for large-scale quantum computation. We focus mainly on two compilation objectives: the minimisation of parametrised rotation gates and the minimisation of entangling gate count. We prove parameter-count optimality under a restricted class of reductions and learn synthesis policies that approach known minimum entangling-gate counts. Together, these results illustrate the complementary roles of graphical reasoning and machine learning in circuit compilation. Graphical calculi expose algebraic structure, while neural networks learn to exploit such structure to guide optimisation. Our contributions are as follows:

Parametrised optimisation (Chapter 4). We prove that phase teleportation minimises parameter count among affine parsimonious reductions of parametrised Clifford circuits in which each parameter occurs once. This result extends to measurement-based computations with gflow and distinct measurement parameters. Allowing fixed T gates or repeated parameters makes parameter optimisation NP-hard in post-selected Clifford circuits.

Learning ZX rewriting (Chapter 5). We train an encoder–decoder Transformer to learn ZX-diagram transformations from examples generated by PyZX's `full_reduce`, without supplying rewrite rules or the diagrams' underlying semantics. The results show that Transformers can learn complex graph transformations from examples alone, providing evidence that they capture stabiliser structure. However, accuracy decreases as sequence length increases, limiting generalisation to larger circuits.

Matrix-group synthesis (Chapter 6). We train reinforcement learning agents to synthesise unseen Clifford and CNOT circuits of varying size and depth. Their mean gate counts are within one gate of the optimum on benchmarks with known optimal solutions, and they outperform existing baselines on larger circuits. We introduce a two-dimensional curriculum that increases both circuit size and synthesis difficulty, supported by equivariant, size-agnostic architectures that allow the same policy to operate across qubit counts.

7.1 Further Work on Parametrised Optimisation

One of our motivations for studying the compilation of PQCs was the coincidence in T-count between PyZX's phase teleportation algorithm [20] and Zhang and Chen's [40] TOpt algorithm. Since our work first appeared, Vandaele, Perdrix and Vuillot [87] have built on it to establish the corresponding parameter-count optimality result for the phase-merging procedure used by TOpt. This explains why the two algorithms reach the same optimum for parametrised Clifford circuits with independent parameters in the reduction model studied here.

We have established optimality when the parameters are independent, and shown that allowing parameters to repeat makes the optimisation problem NP-hard in the post-selected setting. However, this still leaves room for optimisation algorithms that exploit further rewrite rules involving repeated parameters. For example, Mori, Hakoshima and Fujii's [140] multiproduct commutation relations allow whole groups of Pauli rotations to commute even when pairwise commutation cannot achieve the same rearrangement. Such rules could expose further opportunities for merging and cancelling parametrised rotations beyond those found by existing phase-folding algorithms. Future work could characterise these rules, express them graphically, and incorporate them into a compilation algorithm.

The relationship between affine and parsimonious reductions also remains to be fully understood. In our paper [1], we showed that an affine reduction with non-trivial input parameters can be replaced by a parsimonious reduction whenever each input parameter is copied to at most three output locations. The proof is not reproduced in this thesis. We conjecture that the same holds without a bound on the number of copies, which would extend our optimality result to all affine reductions.

7.2 Further Work on Learned Circuit Synthesis

We formulate CNOT and Clifford synthesis as matrix-group synthesis problems. Both involve binary matrices, but the same framework suggests extensions to other groups. For instance, the shortest-word problem over the group S_n of $n \times n$ permutation matrices, with allowed swaps as generators, corresponds to synthesising a swap network with the fewest gates for a given permutation. Synthesis over the special orthogonal group $\mathrm{SO}(2n)$ corresponds to implementing a fermionic Gaussian transformation using elementary Gaussian gates, with the matrix describing its action on $2n$ Majorana operators up to global phase [141]. Synthesis over the unitary group $\mathrm{U}(n)$ corresponds to implementing a passive n-mode optical interferometer using beam splitters and phase shifters [142].

We expect the equivariant, size-agnostic architectures and two-dimensional curriculum to be useful in these settings when the available operations and their costs respect mode relabelling. A single trained model could then solve new instances

across different qubit or mode counts without retraining. For continuous groups, the synthesiser would also need to choose gate parameters, either analytically or through learning.

A broader direction is to use matrix-group synthesisers as building blocks for general-purpose compilers, extending beyond pure CNOT and Clifford circuits to phase polynomials and Clifford+T circuits. ZX-flow [74] and Pauli Dependency DAGs (PDDAGs) [39] offer two possible approaches: they expose Clifford transformations within a larger computation while retaining the constraints on the ordering of non-Clifford rotations. The extraction interfaces in Section 6.11 provide a starting point for integrating our synthesisers with these representations. For Pauli networks, Dubal et al. [143] train resynthesisers for four-, five- and six-qubit subgraphs of the heavy-hex topology, then apply them to larger circuits through peephole optimisation. Our equivariant RL approach generalises to at least 30 qubits for CNOT and Clifford synthesis, suggesting that a single size-agnostic model could also tackle larger Pauli networks directly. This could improve circuit quality by exploiting optimisation opportunities across the boundaries of small peephole blocks.

A further question is how to allocate computation between training and inference-time search. Brown et al. [144] demonstrated the power of combining reinforcement learning with search, while Jones [145] measured a trade-off between training and search compute for AlphaZero agents playing Hex. AlphaCNOT [119] already uses planning and tree search for CNOT synthesis. Combining these techniques with our equivariant, size-agnostic approach would allow us to study how gate count and solve rate depend on training budget, search budget and qubit count, and when additional search is more effective than further training.

7.3 Outlook

Recent demonstrations of below-threshold quantum error correction on superconducting processors [51] and fault-tolerant operations on neutral-atom devices [146] mark substantial progress towards large-scale fault-tolerant quantum computation. Alongside advances in hardware, better compilation can help close the gap between the machines we can build and the resources our algorithms require.

The tools available for this task are also evolving. The ZX-calculus increasingly supports reasoning about fault-tolerant computation, through Pauli webs [72] and distance-preserving rewrites [147] that connect graphical reasoning with error protection. Meanwhile, advances in large language models and agentic AI are expanding the possibilities for automated algorithmic discovery: systems such as AlphaEvolve already propose, evaluate and refine algorithms through computational feedback [148].

This thesis shows how graphical reasoning and reinforcement learning can work together to reduce the resources required by quantum circuits. This is only a

beginning. Combining machine learning with diagrammatic techniques can push the frontier of quantum computation, uncovering new mathematical results and enabling new tools for reasoning, discovery and compilation.

Appendices

Appendix A

Algebraic Foundations of Circuit Synthesis

This appendix collects the group-order calculations and synthesis-bound proofs used in Chapter 2 and Chapter 6. We first derive the orders of the binary general linear and symplectic groups, then prove the CNOT and Clifford synthesis bounds under the cost models stated in the main text. The CNOT bounds follow Patel, Markov and Hayes [23]; the Clifford upper bound uses the canonical form of Aaronson and Gottesman [28].

A.1 Orders of the synthesis groups

Proposition 52 *For every $n \geq 1$, the orders of the CNOT parity-map group and the phase-free Clifford tableau group are*

$$|\mathrm{GL}(n, \mathbb{F}_2)| = \prod_{k=0}^{n-1} \left(2^n - 2^k\right),$$

$$|\mathrm{Sp}(2n, \mathbb{F}_2)| = 2^{n^2} \prod_{j=1}^{n} \left(4^j - 1\right).$$

In particular, both orders are $2^{\Theta(n^2)}$.

Proof. The columns of an invertible binary matrix form an ordered basis of $\mathbb{F}_2^n$. After k independent columns have been chosen, their span contains 2^k vectors, leaving $2^n - 2^k$ choices for the next column. Thus

$$|\mathrm{GL}(n, \mathbb{F}_2)| = \prod_{k=0}^{n-1} \left(2^n - 2^k\right) \geq 2^{n(n-1)}. \tag{142}$$

The counting argument for $\mathrm{Sp}(2n, \mathbb{F}_2)$ is similar, but must also account for the symplectic restriction. An ordered basis $(v_1, w_1, \ldots, v_n, w_n)$ is symplectic when $\langle v_i, v_j \rangle_{\mathrm{sp}} = \langle w_i, w_j \rangle_{\mathrm{sp}} = 0$ and $\langle v_i, w_j \rangle_{\mathrm{sp}} = \delta_{ij}$. The first vector v_1 may be any non-zero vector, giving $2^{2n} - 1$ choices. Its partner w_1 must satisfy $\langle v_1, w_1 \rangle_{\mathrm{sp}} = 1$, which holds for exactly half of the vectors in $\mathbb{F}_2^{2n}$, giving 2^{2n-1} choices. After k pairs have been chosen,

all subsequent vectors must lie in the symplectic complement of their span, which has dimension $2(n-k)$. Thus v_{k+1} has $2^{2(n-k)}-1$ choices and w_{k+1} has $2^{2(n-k)-1}$ choices. Hence

$$\begin{aligned} |\mathrm{Sp}(2n,\mathbb{F}_2)| &= \prod_{k=0}^{n-1} 2^{2(n-k)-1}\left(2^{2(n-k)}-1\right) \\ &= 2^{n^2}\prod_{j=1}^{n}(4^j-1). \end{aligned} \tag{143}$$

For $n \geq 2$, bounding each general-linear factor between 2^{n-1} and 2^n gives $2^{n(n-1)} \leq |\mathrm{GL}(n,\mathbb{F}_2)| < 2^{n^2}$. Similarly, $2^{2j-1} \leq 4^j - 1 < 2^{2j}$ gives $2^{2n^2} \leq |\mathrm{Sp}(2n,\mathbb{F}_2)| < 2^{2n^2+n}$. These bounds establish the claimed $2^{\Theta(n^2)}$ growth. □

A.2 CNOT counting lower bound

Proposition 1 *Let $d_{\mathrm{CNOT}}(n)$ be the maximum, over invertible $n \times n$ binary targets, of the minimum CNOT count for ancilla-free synthesis with all-to-all connectivity and fixed input and output labels. For $n \geq 2$,*

$$d_{\mathrm{CNOT}}(n) \geq \frac{n(n-1)}{\log_2(n(n-1)+1)} = \Omega(n^2/\log n).$$

Proof. By Equation 142, there are at least $2^{n(n-1)}$ invertible binary targets.

There are $n(n-1)$ directed CNOT placements. Adding an identity symbol lets us pad any circuit of at most d gates to a word of length d, so such circuits realise at most $(n(n-1)+1)^d$ different targets. If every target is realisable within d gates, then

$$\begin{aligned} (n(n-1)+1)^d &\geq |\mathrm{GL}(n,\mathbb{F}_2)| \geq 2^{n(n-1)}, \\ d &\geq \frac{n(n-1)}{\log_2(n(n-1)+1)} = \Omega(n^2/\log n). \end{aligned}$$

□

A.3 Clifford counting lower bound

Proposition 2 *With single-qubit Clifford gates free, ancilla-free Clifford synthesis on all-to-all connectivity requires $\Omega(n^2/\log n)$ CNOTs in the worst case. The same bound holds when CZ is the entangling gate.*

Proof. A single-qubit Clifford, up to global phase, is determined by its images of X and Z. The first image has six signed Pauli choices; the second has four choices anticommuting with the first. Thus there are at most 24 possible local factors.

Fix a sequence of ℓ CNOT placements. On each wire, commute local gates past operations on other wires and combine all local gates between successive CNOTs touching that wire. If wire i participates in a_i CNOTs, it has $a_i + 1$ local factors. Since $\sum_i a_i = 2\ell$, there are at most $n + 2\ell$ factors altogether. Let $N(n, d)$ count the distinct operators implementable with at most d CNOTs. To cover all Clifford targets, Equation 142 requires

$$\begin{aligned} 2^{n(n-1)} \leq N(n,d) &\leq \sum_{\ell=0}^{d} (n(n-1))^{\ell} 24^{n+2\ell} \\ &\leq 24^n \left(1 + 24^2 n(n-1)\right)^d. \end{aligned}$$

Hence

$$d \geq \frac{n(n-1) - n\log_2 24}{\log_2(1 + 24^2 n(n-1))} = \Omega(n^2/\log n).$$

Free Hadamards convert each CZ to one CNOT and conversely, so the same argument applies to CZ count. □

A.4 Matching Clifford upper bound

The construction in Section 2.3.2.3 gives $d_{\text{Cliff}}(n) = \Theta(n^2/\log n)$ for ancilla-free synthesis with all-to-all connectivity, fixed wire labels, and free single-qubit Clifford gates, whether CNOT or CZ is used as the entangling gate.

Proof. Apply the PMH bound [23] separately to the five CNOT stages of the Aaronson–Gottesman canonical form. Each costs $O(n^2/\log n)$ CNOTs, and their number is independent of n. The remaining stages use only free single-qubit gates. This gives the upper bound, while Proposition 2 gives the lower bound. Replacing CNOTs with locally equivalent CZs preserves the entangling count. □

Appendix B

Technical Results for Parametrised Compilation

B.1 Parameter-dependent scalars

This appendix proves Proposition 27, the scalar-constancy result used in Chapter 4. We first establish the one-parameter case, then extend the argument to all parameter inputs. Non-triviality is as defined in Definition 24.

> Lemma 56 *Let $D_1[\alpha]$ and $D_2[\alpha]$ be parametrised diagrams with a single parameter α, which is non-trivial in D_1. Suppose*
>
> $$D_1[\alpha] = \lambda(\alpha) D_2[\alpha] \tag{144}$$
>
> *for every α, with $\lambda(\alpha) \neq 0$. Then λ is constant and $D_1' = \mu D_2'$ for some $\mu \in \mathbb{C} \setminus \{0\}$.*

Proof. Choose distinct α, β for which the two diagrams are non-zero. Such a pair exists because a non-trivial one-parameter diagram can vanish at no more than one value. Non-triviality makes $D_1[\alpha]$ and $D_1[\beta]$ non-proportional, and therefore also makes $D_2[\alpha]$ and $D_2[\beta]$ non-proportional. For any third value γ, write

$$|+_\gamma\rangle = a|+_\alpha\rangle + b|+_\beta\rangle \tag{145}$$

using Lemma 21. When γ differs from α and β, both a and b are non-zero. Linearity gives

$$D_1[\gamma] = a\lambda(\alpha) D_2[\alpha] + b\lambda(\beta) D_2[\beta]. \tag{146}$$

Applying the proportionality at γ and expanding $D_2[\gamma]$ in the same basis gives

$$D_1[\gamma] = a\lambda(\gamma) D_2[\alpha] + b\lambda(\gamma) D_2[\beta]. \tag{147}$$

Subtracting,

$$a(\lambda(\alpha) - \lambda(\gamma)) D_2[\alpha] = -b(\lambda(\beta) - \lambda(\gamma)) D_2[\beta]. \tag{148}$$

The two diagrams are non-proportional, so both coefficients vanish. Hence $\lambda(\alpha) = \lambda(\beta) = \lambda(\gamma)$. Since γ was arbitrary, $\lambda = \mu$ is constant. Since $|+_\alpha\rangle$ and $|+_\beta\rangle$ form a basis, $D_1' = \mu D_2'$. □

Corollary 57 *Let $D_1[\alpha]$ and $D_2[\alpha]$ be parametrised diagrams with a single parameter α, which is non-trivial in D_1. If for each $r \in \{0,1,2,3\}$ there is a non-zero scalar λ_r with*

$$D_1\left[r\frac{\pi}{2}\right] = \lambda_r D_2\left[r\frac{\pi}{2}\right], \tag{149}$$

then $D_1' = \mu D_2'$ for a constant non-zero scalar μ. At any Clifford value where the diagrams are non-zero, the corresponding λ_r equals μ.

Proof. At most one of the four evaluations can vanish, since two zero evaluations would make the parameter trivial. Choose three non-zero Clifford evaluations. The argument of Lemma 56 shows that their scalars are equal. Equality on any two of the corresponding basis states gives $D_1' = \mu D_2'$. Any scalar assigned at a common zero is immaterial. □

B.1.1 *Scalar constancy for all parameters*

Proposition 27 *Let D_1 and D_2 be parametrised diagrams with the same parameters, all non-trivial in D_1. If*

$$D_1[\vec{\alpha}] = \lambda(\vec{\alpha}) D_2[\vec{\alpha}]$$

for every $\vec{\alpha}$ and some function $\lambda : \mathbb{R}^k \to \mathbb{C} \setminus \{0\}$, then there is a constant $\lambda' \in \mathbb{C} \setminus \{0\}$ such that $D_1' = \lambda' D_2'$.

Proof. First leave the input wire for α_1 open and fix every other parameter to an arbitrary Clifford assignment $\vec{\delta} \in \{0, \frac{\pi}{2}, \pi, 3\frac{\pi}{2}\}^{k-1}$. The resulting one-parameter slice is non-trivial by Definition 24, so Lemma 56 gives

$$\left(D_1\left[-, \vec{\delta}\right]\right)' = \mu_{\vec{\delta}}\left(D_2\left[-, \vec{\delta}\right]\right)'. \tag{150}$$

Now leave the α_2 wire open as well. For fixed Clifford values of $\alpha_3, ..., \alpha_k$, the displayed equality is known at all four Clifford values of α_2. The second parameter remains non-trivial after the first wire is opened: proportional open maps would remain proportional after plugging any Clifford value into that first wire, contrary to Definition 24. Therefore Corollary 57 applies and removes the dependence of $\mu_{\vec{\delta}}$ on α_2.

Repeat this argument for each remaining parameter. At each step, Corollary 57 shows that the scalar is independent of that parameter and that proportionality holds with its wire left open. After all k wires have been opened,

$$D_1' = \mu D_2' \tag{151}$$

for a single non-zero constant μ. □

Appendix C

Supplementary Details for Matrix-Group Reinforcement Learning

This appendix collects proofs about equivariant policies and random-walk curricula, neural-layer definitions, complete training configurations, and the detailed decoding and evaluation protocols supporting Chapter 6. The group-order and synthesis-bound proofs used by that chapter and Chapter 2 are collected in Appendix A.

C.1 Technical results

C.1.1 *Equivariant optimal policy*

Proposition 53 *For either the CNOT or Clifford synthesis MDP with discount factor $\gamma \in [0, 1)$, there exists an optimal stationary stochastic Markov policy $\pi^* : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ whose corresponding optimal value function $V^* : \mathcal{S} \to \mathbb{R}$ satisfies*

$$\pi^*(\sigma \cdot M) = \sigma \cdot \pi^*(M), \quad V^*(\sigma \cdot M) = V^*(M).$$

Proof. For either MDP, let V^* be the unique solution of its discounted Bellman optimality equation,

$$V^*(M) = \max_{G \in \mathcal{A}} [\mathcal{R}(M, G) + \gamma V^*(\mathcal{T}(M, G))]. \tag{152}$$

For a fixed $\sigma \in \mathrm{Sym}(n)$, define $W_\sigma(M) \coloneqq V^*(\sigma \cdot M)$. Applying Equation 152 and reindexing the maximum by the bijection $G \mapsto \sigma \cdot G$ gives

$$\begin{aligned} W_\sigma(M) &= \max_{G' \in \mathcal{A}} [\mathcal{R}(\sigma \cdot M, G') + \gamma V^*(\mathcal{T}(\sigma \cdot M, G'))] \\ &= \max_{G \in \mathcal{A}} [\mathcal{R}(\sigma \cdot M, \sigma \cdot G) + \gamma V^*(\mathcal{T}(\sigma \cdot M, \sigma \cdot G))] \\ &= \max_{G \in \mathcal{A}} [\mathcal{R}(M, G) + \gamma V^*(\sigma \cdot \mathcal{T}(M, G))] \\ &= \max_{G \in \mathcal{A}} [\mathcal{R}(M, G) + \gamma W_\sigma(\mathcal{T}(M, G))]. \end{aligned} \tag{153}$$

Hence W_σ is another fixed point of the Bellman optimality operator. Uniqueness gives $W_\sigma = V^*$, and therefore

$$V^*(\sigma \cdot M) = V^*(M). \tag{154}$$

Now define the optimal action-value function and the set of optimal actions by

$$\begin{aligned} Q^*(M, G) &:= \mathcal{R}(M, G) + \gamma V^*(\mathcal{T}(M, G)), \\ \mathcal{A}^*(M) &:= \arg\max_{G \in \mathcal{A}} Q^*(M, G). \end{aligned} \tag{155}$$

The equivariance of $\mathcal{T}$, invariance of $\mathcal{R}$, and Equation 154 imply

$$Q^*(\sigma \cdot M, \sigma \cdot G) = Q^*(M, G), \tag{156}$$

so

$$\mathcal{A}^*(\sigma \cdot M) = \sigma \cdot \mathcal{A}^*(M). \tag{157}$$

Let $\pi^*(\cdot \mid M)$ be the uniform distribution over $\mathcal{A}^*(M)$. It is optimal because it is supported only on maximising actions. Since the group action bijectively maps the maximising set at M to the maximising set at $\sigma \cdot M$, it also satisfies

$$\pi^*(\sigma \cdot G \mid \sigma \cdot M) = \pi^*(G \mid M), \tag{158}$$

which is equivalent to $\pi^*(\sigma \cdot M) = \sigma \cdot \pi^*(M)$.

If the objective includes a fixed rollout limit, augment the state with the number of remaining steps. Qubit permutations leave this coordinate unchanged, so the same proof applies to the augmented state space. □

C.1.2 *CNOT random-walk convergence*

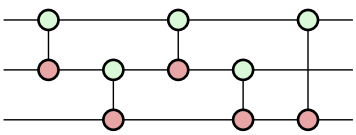
> Theorem 54 *(CNOT random-walk limit.) For $n \geq 3$, the walk on $\mathrm{GL}(n, \mathbb{F}_2)$ that right-multiplies by a uniformly sampled generator from $\mathcal{A}_n^{\mathrm{CNOT}}$ converges from the identity to the uniform distribution. For $n = 2$, the corresponding fixed-length walk has period two and does not converge.*

Proof. CNOT gates generate $\mathrm{GL}(n, \mathbb{F}_2)$, so the walk is irreducible. Right multiplication permutes its elements, making the uniform distribution stationary. By the finite-state Markov-chain ergodic theorem, it therefore suffices to establish aperiodicity for $n \geq 3$.

Every CNOT generator is an involution by Equation 119, so applying it twice gives a two-step return to the identity. On qubits 1, 2, and 3, the identity

$$\mathsf{CNOT}_{1,2}\mathsf{CNOT}_{2,3}\mathsf{CNOT}_{1,2}\mathsf{CNOT}_{2,3}\mathsf{CNOT}_{1,3} = I_n$$

gives a five-step return whenever $n \geq 3$:



The period divides both two and five and is therefore one.

For $n = 2$, direct enumeration shows that even-length walks reach three group elements and odd-length walks reach the other three. The walk therefore has period two and does not converge. □

C.1.3 *Clifford random-walk convergence*

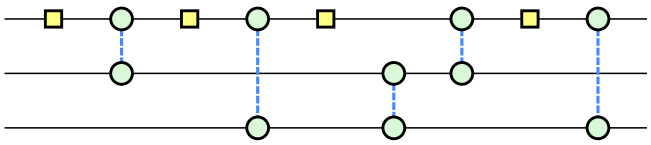
> **Theorem 55** *(Clifford random-walk limit.) For $n \geq 3$, the walk on $\mathrm{Sp}(2n, \mathbb{F}_2)$ that right-multiplies by a uniformly sampled generator from $\{\mathsf{H}_i, \mathsf{S}_i, \mathsf{CZ}_{i,j}\}$ converges from the identity to the uniform distribution. For $n = 1$ and $n = 2$, the corresponding fixed-length walk has period two and does not converge.*

Proof. The H, S and CZ gates generate the binary symplectic group, so the walk is irreducible. Right multiplication preserves the uniform distribution. As in the CNOT case, the finite-state Markov-chain ergodic theorem reduces convergence for $n \geq 3$ to aperiodicity.

Every generator is an involution, giving a return of length two. For $n \geq 3$, the relation

$$\mathsf{H}_1\mathsf{CZ}_{1,2}\mathsf{H}_1\mathsf{CZ}_{1,3}\mathsf{H}_1\mathsf{CZ}_{2,3}\mathsf{CZ}_{1,2}\mathsf{H}_1\mathsf{CZ}_{1,3} = I$$

is represented by the circuit



This relation involves only qubits 1, 2 and 3, giving a nine-step return to the identity for any $n \geq 3$. The period divides both two and nine and is therefore one. Direct enumeration gives period two for $n = 1, 2$. □

C.2 Equivariant layer variants

This section gives schematic definitions of the neural-layer variants used in the Clifford experiments and architecture ablation. Write $H^{(\ell)} = \left\{h_i^{(\ell)}\right\}_i$ for the qubit embeddings entering layer ℓ, and write $R = \left\{r_{ij}\right\}_{i,j}$ for the fixed pair features obtained from the input tableau. Every equivariant variant shares its learned maps across qubit indices.

C.2.1 *Deep Sets*

Deep Sets [127] process unordered inputs using shared elementwise maps and symmetric aggregation:

$$s^{(\ell)} := \operatorname{pool}_j \psi_\ell\Big(h_j^{(\ell)}\Big),$$
$$h_i^{(\ell+1)} := \varphi_\ell\Big(h_i^{(\ell)}, s^{(\ell)}\Big).$$

The pooled summary $s^{(\ell)}$ is invariant to relabelling, so the updated embeddings remain equivariant. Using sum, mean or maximum pooling allows the same layer to process any number of qubits.

C.2.2 *Set Transformer*

Set Transformers [128] use self-attention without positional encodings to process unordered inputs. For one attention head, let $q_i := W_Q h_i^{(\ell)}$, $k_j := W_K h_j^{(\ell)}$, and $v_j := W_V h_j^{(\ell)}$. Then

$$\alpha_{ij} := \frac{\exp\big(q_i^\top k_j/\sqrt{d}\big)}{\sum_m \exp\big(q_i^\top k_m/\sqrt{d}\big)},$$
$$u_i := h_i^{(\ell)} + \sum_j \alpha_{ij} v_j, \quad h_i^{(\ell+1)} := u_i + \mathrm{FF}_\ell(u_i).$$

Multi-head attention applies several such updates in parallel before a shared projection. Relabelling the inputs simply relabels the outputs. This is the ordinary Transformer used in the Clifford architecture ablation.

C.2.3 *RelTransformer*

The RelTransformer uses the relation-aware attention of Shaw et al. [129]. For one attention head, let $q_i := W_Q h_i^{(\ell)}$, $k_j := W_K h_j^{(\ell)}$, and $v_j := W_V h_j^{(\ell)}$. The pair features provide learned key and value corrections $a_{ij}^K := \kappa_\ell(r_{ij})$ and $a_{ij}^V := \nu_\ell(r_{ij})$, giving

$$s_{ij} := \frac{q_i^\top\big(k_j + a_{ij}^K\big)}{\sqrt{d}}, \quad \alpha_{ij} := \frac{\exp(s_{ij})}{\sum_m \exp(s_{im})},$$
$$u_i := h_i^{(\ell)} + \sum_j \alpha_{ij}\big(v_j + a_{ij}^V\big), \quad h_i^{(\ell+1)} := u_i + \mathrm{FF}_\ell(u_i).$$

Multi-head attention applies several such updates in parallel before a shared projection. With no positional encoding, relabelling the qubits relabels the outputs, while the pair features allow the attention to depend on the tableau.

C.2.4 *RelGNN*

The RelGNN uses message passing as described by Gilmer et al. [130], with qubits as nodes and pair features as edge features:

$$m_{i\leftarrow j}^{(\ell)} := \varphi_{\mathrm{msg}}^{(\ell)}\Big(h_i^{(\ell)}, h_j^{(\ell)}, r_{ij}, r_{ji}\Big),$$
$$\mu_i^{(\ell)} := \operatorname{aggregate}_j m_{i\leftarrow j}^{(\ell)}, \quad h_i^{(\ell+1)} := \varphi_{\mathrm{upd}}^{(\ell)}\Big(h_i^{(\ell)}, \mu_i^{(\ell)}\Big).$$

Because the maps are shared and aggregation ignores message order, relabelling the qubits simply relabels the updated embeddings. Hartford et al. [149] give a closely related treatment of pair-indexed arrays. In the Clifford implementation, the node update also receives the rank-based progress features described in Section 6.6.1.

C.2.5 *MLP controls*

The equivariant **MLP** removes communication between qubits. Each embedding is updated independently by the same learned map,

$$h_i^{(\ell+1)} := \varphi_\ell\Big(h_i^{(\ell)}\Big).$$

Finally, **FlatMLP** flattens the tableau into one fixed-dimensional vector and applies an unconstrained MLP. It is neither permutation-equivariant nor size-agnostic and therefore serves as the unstructured control.

C.3 Training configurations

This section gives the training configurations for three sets of experiments: the Clifford RelGNN experiments and RelTransformer architecture ablation from our Clifford-synthesis paper [3], and the CNOT BiRel experiments from our subsequent matrix-group synthesis paper [4]. All three used PPO with a random-walk curriculum. The settings and checkpoint choices below combine the published details with archived training records.

C.3.1 *Clifford RelGNN*

We trained the RelGNN on six-qubit targets, then continued training on ten-qubit targets. Table 7 gives the six-qubit training settings. Curriculum difficulty denotes the expected random-walk length defined in Equation 141.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Training steps	1.5×10^9	Parallel environments	2048
Rollout length	256	Curriculum range	difficulties 1–1000
Advance threshold	success = 1.00	Learning rate	2.5×10^{-4}
Batch size	8192	Epochs per update	5
Discount γ	0.99	GAE λ	0.95
Policy clip	0.15	Value clip	0.2
Hamming-shaping scale	0.5	One-qubit cost	0.01
Terminal bonus	25	Algorithm	PPO

Table 7: Six-qubit Clifford RelGNN training configuration [3].

The reported results use one checkpoint at each width. The training seed, checkpoint-selection rule, training time and hardware were not recorded in the paper.

C.3.2 *Clifford RelTransformer*

For the architecture ablation, we trained three six-qubit RelTransformer models with seeds 0, 1 and 2. Each used two relation-aware attention layers, four attention heads and 128-dimensional qubit representations. Figures 8–9 give the model, PPO and curriculum settings. The ablation reports means and standard deviations over the three seeds (Table 6).

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Training topology	Six-qubit curriculum	Seeds	0, 1, 2
Hidden dimension	128	Relation dimension	128
Attention layers	2	Attention heads	4
Training budget	1.5×10^9 steps	Parallel environments	2048
Rollout length	256	Minibatch size	8192
PPO epochs per update	4	Learning rate	2.5×10^{-4}
Discount γ	0.99	GAE λ	0.95
Initial entropy coefficient	0.08	Entropy floor	30% of initial
Value-loss coefficient	0.5	Maximum gradient norm	1.5
Policy clip	0.15	Value clip	0.2
Optimiser	Adam	Adam (β_1, β_2)	$(0.9, 0.999)$
Adam ε	10^{-5}	Reward clipping	disabled
Target-KL stopping	disabled	Training precision	bfloat16

Table 8: Model and PPO settings for the six-qubit Clifford RelTransformer ablation.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Initial difficulty	1	Maximum difficulty	1000
Difficulty increment	0.25	Advance threshold	success ≥ 0.90
Progression source	training rollouts	Progression probe episodes	200
Near-threshold margin	0.02	Plateau-probe delay	20 checks
Plateau-probe interval	10 checks	Minimum advance evaluations	1
Direct-advance window	5	Evaluation frequency	98,304 steps
Evaluation episodes	200	Reward mode	Hamming shaping
Hamming-shaping scale	0.5	One-qubit cost	0.01
Terminal bonus	25	Repetition penalty	0
Minimum episode limit	32	Limit from difficulty 12	1000
Episode-limit headroom	2.2	Episode-limit slack	64
Plateau window	300 checks	Plateau gate	success = 1.00
Plateau stopping EMA	0.95	Entropy-rescue patience	3 checks
Entropy-rescue multiplier	2	Optimiser reset on rescue	disabled

Table 9: Curriculum, reward and evaluation settings for the Clifford RelTransformer ablation.

Each run had a maximum budget of 1.5×10^9 environment steps. The ablation used the following checkpoints, saved before that budget was reached.

Seed	**Environment steps**	**Difficulty**
0	531,103,744	250.75
1	522,190,848	246.75
2	530,579,456	250.50

C.3.3 *CNOT BiRel*

We trained a two-layer BiRel actor–critic with hidden dimension 96. All runs used 2048 parallel environments and two PPO epochs per update, with $\gamma = 0.99$ and GAE $\lambda = 0.95$. The entropy and value coefficients were 0.01 and 0.5, respectively, and gradients were clipped at norm 1. The curriculum advanced at a success rate of 0.90.

The 8- and 10-qubit runs started from random initialisation. The 12-, 14- and 16-qubit runs continued from the 10-qubit checkpoint, while the 18- and 20-qubit runs continued from the 12-qubit checkpoint. Table 10 gives the settings that varied with qubit count.

Qubits	Rollout length	Batch size	Learning rate	PPO clip	Difficulty step
8	256	8192	2.5×10^{-4}	0.2	0.1
10–14	512	16,384	2.5×10^{-4}	0.2	0.5
16	512	16,384	1.25×10^{-4}	0.1	0.5
18–20	256	8192	1.25×10^{-4}	0.1	0.5

Table 10: CNOT BiRel training settings that vary with qubit count.

All runs used seed zero, so these experiments do not measure variability across independent training runs. Each run had a maximum budget of 20,000 updates, but plateau stopping ended training between updates 700 and 900.

C.4 Decoding and evaluation protocols

This section gives the Clifford decoding and evaluation procedures used in Section 6.10. We represent targets and check successful synthesis using phase-free binary symplectic tableaus. Uniform targets are generated with Qiskit's `random_clifford`, retaining only its `symplectic_matrix` and discarding the Pauli signs.

We rank candidate circuits by CZ count, using total gate count to break ties. In the combined exact-recovery search, we compare candidates only by their CZ count above the known optimum.

C.4.1 *Exact-recovery search*

Greedy decoding is applied once to the target tableau and once to its inverse. Of the successful circuits, we keep the one with lower CZ count, breaking ties by total gate count. The extended search also uses both tableaus, with a limit of 512 generator applications per trajectory and 2048 parallel environments.

We combine the results from four temperature schedules: *t40*, *t40→greedy*, *t40→t25* and *t40→t12*. The *t40* schedule uses a fixed temperature of 4.0, while *t40→t25* varies the temperature from 4.0 to 2.5. The *t40→t12* schedule linearly decreases the temperature from 4.0 to 1.2, while *t40→greedy* decreases it from 4.0 to 0.05. Each schedule runs 4096 trials per circuit for each of the target and inverse tableaus.

For each instance, we report the time at which any temperature schedule first finds an optimal circuit during the search over the full benchmark suite. The search ran on a single NVIDIA A100 80 GB GPU using the native CUDA rollout backend. These timings describe this implementation and hardware, and do not establish a controlled speed comparison with earlier methods.

C.4.2 *Larger-width generalisation*

At larger qubit counts, decoding uses the no-loop safeguard and a limit of $6n^2$ actions per trajectory. This is an evaluation budget, rather than a worst-case synthesis guarantee. The learned policy is evaluated on 100 uniform targets at each qubit count. For 21–30 qubits, the Qiskit means over all targets, shown as solid curves in Figure 36, come from earlier evaluations on 200 targets.

C.4.3 *Architecture-ablation evaluation*

The architecture ablation uses deterministic decoding with the no-loop safeguard. The evaluation was configured to sample 32 additional trajectories, but the filter re-ranked all actions at every step. These trajectories therefore reproduced the deterministic result and did not change the selected circuit.

The ordinary Transformer result is reproduced from the published table. Archived per-run summaries are available for the other four architectures, but not for this row.

C.5 Supplementary CNOT results

The following tables supplement Section 6.9 with the published comparison against known optima, transfer results at every evaluated qubit count, and comparisons on uniformly sampled targets.

C.5.1 *Comparison with known optima*

Table 11 reproduces the small-instance comparison from [4]. The main chapter includes later baseline evaluations, so some baseline values differ. The learned-policy and optimal counts are unchanged. **Gap** is the learned mean minus the optimal mean.

n	**Optimum**	**Learned**	**Gap**	**AlphaCNOT**	**AECM**	**GGE**	**PMH**
4	5.35	5.35	0.00	5.32	6.61	7.94	6.98
5	7.70	7.71	+0.01	8.16	10.58	12.34	11.07
6	10.81	11.10	+0.29	11.10	15.34	17.49	16.40
7	13.50	14.23	+0.73	15.41	21.08	23.40	22.62

Table 11: Complete mean CNOT counts on 100 length-n^2 random-walk targets in the certified-optimum range [4]. AlphaCNOT uses different samples from the same target distribution.

C.5.2 *Transfer across qubit counts*

Table 12 compares the 20-qubit policy with the checkpoint trained up to each target width, where available, and with elimination baselines. **Transferred** denotes

the 20-qubit policy, **Specialist** the checkpoint for that qubit count, and **Δ** the transferred policy's mean CNOT count minus the specialist's.

n	**Transferred**	**Specialist**	**Δ**	**GGE**	**PMH**	**GE**
8	19.83	18.42	+1.41	25.21	27.56	35.44
10	28.79	27.76	+1.03	38.47	41.73	53.80
12	39.44	37.90	+1.54	53.88	60.22	76.87
14	51.95	50.95	+1.00	71.20	81.19	104.88
16	65.68	65.28	+0.40	90.17	104.36	135.26
18	80.98	80.86	+0.12	111.39	128.61	168.77
20	97.78	97.78	0.00	135.36	159.21	209.05
25	150.63	–	–	201.51	241.94	325.54
30	221.12	–	–	278.89	334.34	463.59

Table 12: Complete mean CNOT counts on 100 length-n^2 random-walk targets in the scalable range [4]. GE denotes plain Gaussian elimination.

C.5.3 *Uniform-target evaluation*

Table 13 gives the results underlying Figure 35, including the comparison with checkpoints trained at each width. These targets are sampled uniformly from $\mathrm{GL}(n, \mathbb{F}_2)$, testing generalisation beyond the finite random walks used in training.

n	**Transferred**	**Specialist**	**Δ**	**GGE**	**PMH**	**GE**
8	20.07	18.54	+1.53	25.12	27.27	34.93
10	29.61	28.08	+1.53	38.44	42.38	54.47
12	39.71	38.33	+1.38	53.25	60.38	78.12
14	52.38	50.99	+1.39	71.51	80.84	103.80
16	65.98	65.37	+0.61	90.58	105.59	135.69
18	80.63	80.41	+0.22	111.49	129.14	168.47
20	98.19	98.19	0.00	134.90	158.95	208.67
25	151.06	–	–	201.46	242.50	323.90
30	221.75	–	–	280.90	334.31	462.37

Table 13: Mean CNOT counts on 100 uniformly sampled $\mathrm{GL}(n, \mathbb{F}_2)$ targets. The transferred policy was trained up to 20 qubits, and Δ is its mean CNOT count minus that of the corresponding specialist. The learned policies successfully synthesise every target.

C.6 Supplementary Clifford results

This section supplements Section 6.10 with the complete finite-random-walk comparisons and results across a broader range of target difficulties.

C.6.1 *Finite-random-walk comparisons*

Table 14 compares the RelTransformer with known optima and the Qiskit baselines. **Gap** is the learned mean CZ count minus the optimal mean, and **A–G** denotes Aaronson–Gottesman synthesis.

n	**Optimum**	**Learned**	**Gap**	**Qiskit greedy**	**A–G**
4	3.03	3.05	+0.02	3.48	6.78
5	5.07	5.13	+0.06	6.30	11.65
6	7.47	7.73	+0.26	10.48	20.24

Table 14: Mean CZ counts on 100 length-n^2 random-walk targets with known optimal CZ counts, certified by meet-in-the-middle search [4].

Table 15 gives the transfer results at every evaluated width, extending the four widths discussed in Section 6.10.1.

n	**Learned**	**Gap**	**Qiskit greedy**	**A–G**
7	10.65	−5.68	16.33	29.13
8	13.86	−8.79	22.65	39.05
12	32.57	−23.21	55.78	102.27
16	62.39	−44.38	106.77	194.42
20	106.46	−65.37	171.83	318.85
24	166.45	−83.56	250.01	465.78
30	296.71	−100.73	397.44	747.56

Table 15: Mean CZ counts on 100 length-n^2 random-walk targets at larger qubit counts, where exact optima are unavailable [4]. **Gap** is the learned policy's mean CZ count minus that of Qiskit greedy.

C.6.2 *Generalisation across target difficulties*

The RelGNN experiment evaluates 100 held-out targets for each $n = 7, ..., 30$ and $d \in \{16, 32, 64, 128, 256, 512, 1024, \infty\}$. For finite d, targets are generated by applying d uniformly sampled Clifford generators to the identity. For $d = \infty$, targets are sampled uniformly from $\mathrm{Sp}(2n, \mathbb{F}_2)$, the limiting distribution established in Theorem 55. The learned policy and both Qiskit baselines use the same 100 finite-difficulty targets. The uniform-target sample sizes and trajectory limit are given in Section C.4.2.

Among the evaluated checkpoints, the one trained on ten qubits generally gives the lowest mean CZ count on successfully solved targets at finite difficulties. At 30 qubits and difficulty 1024, it solves all 100 targets with a mean of 323.3 CZ gates, compared with 447.5 for Qiskit greedy and 783.4 for Aaronson–Gottesman [3]. For uniformly sampled targets, however, its solve rate falls from 99% at 24 qubits to 59% at 30 qubits. Comparing mean gate counts on identical instances therefore requires restricting the baselines to the targets solved by the learned policy.

References

[1] J. van de Wetering, R. Yeung, T. Laakkonen, and A. Kissinger, 'Optimal Compilation of Parametrised Quantum Circuits', *Quantum*, vol. 9, p. 1828, 2025, doi: 10.22331/Q-2025-08-27-1828.

[2] R. Yeung, K. Meichanetzidis, A. Krajenbrink, and F. Charton, 'Teaching Small Transformers to Rewrite ZX Diagrams', in *NeurIPS 2023 Workshop on Mathematical Reasoning and AI*, 2023. [Online]. Available: https://mathai2023.github.io/papers/34.pdf

[3] R. Yeung, A. Kissinger, and R. Cornish, 'Equivariant Reinforcement Learning for Clifford Quantum Circuit Synthesis'. [Online]. Available: https://arxiv.org/abs/2605.10910

[4] R. Yeung, A. Kissinger, and R. Cornish, 'Reusable Equivariant Neural Compilers for Matrix-Group Quantum Circuit Synthesis', in *Proceedings of the 2026 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2026. [Online]. Available: https://sites.google.com/view/qce-workshop-ai4qc/program

[5] E. Bernstein and U. Vazirani, 'Quantum Complexity Theory', *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1411–1473, 1997, doi: 10.1137/S0097539796300921.

[6] P. W. Shor, 'Algorithms for Quantum Computation: Discrete Logarithms and Factoring', in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, IEEE, 1994, pp. 124–134. doi: 10.1109/SFCS.1994.365700.

[7] C. Gidney and M. Ekerå, 'How to Factor 2048 Bit RSA Integers in 8 Hours Using 20 Million Noisy Qubits', *Quantum*, vol. 5, p. 433, 2021, doi: 10.22331/q-2021-04-15-433.

[8] C. Gidney, 'How to Factor 2048 Bit RSA Integers with Less than a Million Noisy Qubits', *arXiv preprint arXiv:2505.15917*, 2025.

[9] J. Preskill, 'Quantum Computing in the NISQ Era and Beyond', *Quantum*, vol. 2, p. 79, 2018, doi: 10.22331/q-2018-08-06-79.

[10] J. van de Wetering and M. Amy, 'Optimising Quantum Circuits is Generally Hard'. [Online]. Available: https://arxiv.org/abs/2310.05958

[11] B. Coecke and R. Duncan, 'Interacting Quantum Observables: Categorical Algebra and Diagrammatics', *New Journal of Physics*, vol. 13, no. 4, p. 43016, 2011, doi: 10.1088/1367-2630/13/4/043016.

[12] E. Jeandel, S. Perdrix, and R. Vilmart, 'Completeness of the ZX-Calculus', *Logical Methods in Computer Science*, vol. 16, no. 2, p. 11, 2020, doi: 10.23638/LMCS-16(2:11)2020.

[13] J. Jumper *et al.*, 'Highly Accurate Protein Structure Prediction with AlphaFold', *Nature*, vol. 596, pp. 583–589, 2021, doi: 10.1038/s41586-021-03819-2.

[14] A. Davies *et al.*, 'Advancing Mathematics by Guiding Human Intuition with AI', *Nature*, vol. 600, pp. 70–74, 2021, doi: 10.1038/s41586-021-04086-x.

[15] R. Sutton, 'The Bitter Lesson'. [Online]. Available: http://www.incompleteideas.net/IncIdeas/BitterLesson.html

[16] D. Silver *et al.*, 'A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play', *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018, doi: 10.1126/science.aar6404.

[17] F. J. R. Ruiz *et al.*, 'Quantum Circuit Optimization with AlphaTensor', *Nature Machine Intelligence*, vol. 7, pp. 374–385, 2025, doi: 10.1038/s42256-025-01001-1.

[18] A. Kissinger and J. van de Wetering, 'PyZX: Large Scale Automated Diagrammatic Reasoning', in *Proceedings 16th International Conference on Quantum Physics and Logic*, in Electronic Proceedings in Theoretical Computer Science, vol. 318. 2020, pp. 229–241. doi: 10.4204/EPTCS.318.14.

[19] R. Duncan, A. Kissinger, S. Perdrix, and J. van de Wetering, 'Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus', *Quantum*, vol. 4, p. 279, 2020, doi: 10.22331/q-2020-06-04-279.

[20] A. Kissinger and J. van de Wetering, 'Reducing the Number of Non-Clifford Gates in Quantum Circuits', *Physical Review A*, vol. 102, no. 2, p. 22406, 2020, doi: 10.1103/PhysRevA.102.022406.

[21] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th Anniversary. Cambridge University Press, 2010. doi: 10.1017/CBO9780511976667.

[22] T. Häner, D. S. Steiger, K. Svore, and M. Troyer, 'A Software Methodology for Compiling Quantum Programs', *Quantum Science and Technology*, vol. 3, no. 2, p. 20501, 2018, doi: 10.1088/2058-9565/aaa5cc.

[23] K. N. Patel, I. L. Markov, and J. P. Hayes, 'Optimal Synthesis of Linear Reversible Circuits', *Quantum Information and Computation*, vol. 8, no. 3–4, pp. 282–294, 2008, doi: 10.26421/QIC8.3-4-4.

[24] C. Jordan, *Traité des substitutions et des équations algébriques*. Paris: Gauthier-Villars, 1870. [Online]. Available: https://archive.org/details/traitdessubsti00jorduoft

[25] Y. Lafont, 'Towards an Algebraic Theory of Boolean Circuits', *Journal of Pure and Applied Algebra*, vol. 184, no. 2–3, pp. 257–310, 2003, doi: 10.1016/S0022-4049(03)00069-0.

[26] D. Gottesman, 'The Heisenberg Representation of Quantum Computers'. [Online]. Available: https://arxiv.org/abs/quant-ph/9807006

[27] J. Dehaene and B. De Moor, 'Clifford Group, Stabilizer States, and Linear and Quadratic Operations over GF(2)', *Physical Review A*, vol. 68, no. 4, p. 42318, 2003, doi: 10.1103/PhysRevA.68.042318.

[28] S. Aaronson and D. Gottesman, 'Improved Simulation of Stabilizer Circuits', *Physical Review A*, vol. 70, no. 5, p. 52328, 2004, doi: 10.1103/PhysRevA.70.052328.

[29] P. Selinger, 'Generators and Relations for n-Qubit Clifford Operators', *Logical Methods in Computer Science*, vol. 11, no. 2, p. 10, 2015, doi: 10.2168/LMCS-11(2:10)2015.

[30] A. Meijer-van de Griend and R. Duncan, 'Architecture-Aware Synthesis of Phase Polynomials for NISQ Devices', in *Proceedings of the 19th International Conference on Quantum Physics and Logic*, in Electronic Proceedings in Theoretical Computer Science, vol. 394. 2023, pp. 116–140. doi: 10.4204/EPTCS.394.8.

[31] M. Amy, P. Azimzadeh, and M. Mosca, 'On the Controlled-NOT Complexity of Controlled-NOT–Phase Circuits', *Quantum Science and Technology*, vol. 4, no. 1, p. 15002, 2018, doi: 10.1088/2058-9565/aad8ca.

[32] M. Amy and M. Mosca, 'T-Count Optimization and Reed–Muller Codes', *IEEE Transactions on Information Theory*, vol. 65, no. 8, pp. 4771–4784, 2019, doi: 10.1109/TIT.2019.2906374.

[33] L. E. Heyfron and E. T. Campbell, 'An Efficient Quantum Compiler that Reduces T Count', *Quantum Science and Technology*, vol. 4, no. 1, p. 15004, 2019, doi: 10.1088/2058-9565/aad604.

[34] N. J. Ross and P. Selinger, 'Optimal ancilla-free Clifford+T approximation of z-rotations', *Quantum Information and Computation*, vol. 16, no. 11–12, pp. 901–953, 2016, [Online]. Available: https://arxiv.org/abs/1403.2975

[35] M. Amy, 'Towards Large-Scale Functional Verification of Universal Quantum Circuits', in *Proceedings of the 15th International Conference on Quantum Physics and Logic*, in Electronic Proceedings in Theoretical Computer Science, vol. 287. 2019, pp. 1–21. doi: 10.4204/EPTCS.287.1.

[36] M. Amy, J. Chen, and N. J. Ross, 'A Finite Presentation of CNOT-Dihedral Operators', in *Proceedings of the 14th International Conference on Quantum Physics and Logic*, in Electronic Proceedings in Theoretical Computer Science, vol. 266. 2018, pp. 84–97. doi: 10.4204/EPTCS.266.5.

[37] X. Bian and P. Selinger, 'Generators and Relations for 2-Qubit Clifford+T Operators', *Electronic Proceedings in Theoretical Computer Science*, vol. 394, pp. 13–28, 2023, doi: 10.4204/EPTCS.394.2.

[38] M. Cerezo *et al.*, 'Variational Quantum Algorithms', *Nature Reviews Physics*, vol. 3, pp. 625–644, 2021, doi: 10.1038/s42254-021-00348-9.

[39] W. Simmons, 'Relating Measurement Patterns to Circuits via Pauli Flow', in *Proceedings 18th International Conference on Quantum Physics and Logic*, C. Heunen and M. Backens, Eds, in Electronic Proceedings in Theoretical Computer Science, vol. 343. 2021, pp. 50–101. doi: 10.4204/EPTCS.343.4.

[40] F. Zhang and J. Chen, 'Optimizing T Gates in Clifford+T Circuit as Pi/4 Rotations around Paulis'. [Online]. Available: https://arxiv.org/abs/1903.12456

[41] A. Cowtan, W. Simmons, and R. Duncan, 'A Generic Compilation Strategy for the Unitary Coupled Cluster Ansatz'. [Online]. Available: https://arxiv.org/abs/2007.10515

[42] S. Bravyi, G. Smith, and J. A. Smolin, 'Trading Classical and Quantum Computational Resources', *Physical Review X*, vol. 6, p. 21043, 2016, doi: 10.1103/PhysRevX.6.021043.

[43] D. Litinski, 'A Game of Surface Codes: Large-Scale Quantum Computing with Lattice Surgery', *Quantum*, vol. 3, p. 128, 2019, doi: 10.22331/q-2019-03-05-128.

[44] A. T. Schmitz, N. P. D. Sawaya, S. Johri, and A. Y. Matsuura, 'Graph Optimization Perspective for Low-Depth Trotter–Suzuki Decomposition', *Physical Review A*, vol. 109, p. 42418, 2024, doi: 10.1103/PhysRevA.109.042418.

[45] T. Tomesh *et al.*, 'Optimized Quantum Program Execution Ordering to Mitigate Errors in Simulations of Quantum Systems', in *2021 International Conference on Rebooting Computing (ICRC)*, 2021. doi: 10.1109/ICRC53822.2021.00013.

[46] Q. Huang, D. Winderl, A. Meijer-van de Griend, and R. Yeung, 'Redefining Lexicographical Ordering: Optimizing Pauli String Decompositions for Quantum Compiling'. [Online]. Available: https://arxiv.org/abs/2408.00354

[47] A. Clément, N. Heurtel, S. Mansfield, S. Perdrix, and B. Valiron, 'A Complete Equational Theory for Quantum Circuits', in *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2023. doi: 10.1109/LICS56636.2023.10175801.

[48] A. Clément, N. Delorme, and S. Perdrix, 'Minimal Equational Theories for Quantum Circuits'. [Online]. Available: https://arxiv.org/abs/2311.07476v2

[49] E. Jeandel, S. Perdrix, and R. Vilmart, 'Diagrammatic Reasoning beyond Clifford+T Quantum Mechanics'. 2018. doi: 10.1145/3209108.3209139.

[50] IBM Quantum, 'Processor types'. [Online]. Available: https://quantum.cloud.ibm.com/docs/en/guides/processor-types

[51] Google Quantum AI and Collaborators, 'Quantum error correction below the surface code threshold', *Nature*, vol. 638, pp. 920–926, 2025, doi: 10.1038/s41586-024-08449-y.

[52] G. Li, Y. Ding, and Y. Xie, 'Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices'. [Online]. Available: https://arxiv.org/abs/1809.02573

[53] Quantinuum, 'Quantinuum System Model H2 Product Data Sheet'. 2025. [Online]. Available: https://docs.quantinuum.com/systems/_static/assets/data_sheets/Quantinuum%20H2%20Product%20Data%20Sheet.pdf

[54] A. Kissinger and A. Meijer-van de Griend, 'CNOT Circuit Extraction for Topologically-Constrained Quantum Memories', *Quantum Information and Computation*, vol. 20, no. 7–8, pp. 581–596, 2020, doi: 10.26421/QIC20.7-8-4.

[55] S. Martiel and T. Goubault de Brugière, 'Architecture aware compilation of quantum circuits via lazy synthesis', *Quantum*, vol. 6, p. 729, 2022, doi: 10.22331/q-2022-06-07-729.

[56] D. Winderl, Q. Huang, A. Meijer-van de Griend, and R. Yeung, 'Architecture-Aware Synthesis of Stabilizer Circuits from Clifford Tableaus'. [Online]. Available: https://arxiv.org/abs/2309.08972

[57] A. Meijer-van de Griend and S. M. Li, 'Dynamic Qubit Routing with CNOT Circuit Synthesis for Quantum Compilation', *Electronic Proceedings in Theoretical Computer Science*, vol. 394, pp. 363–399, 2023, doi: 10.4204/EPTCS.394.18.

[58] P. Selinger, 'Quantum circuits of T-depth one', *Physical Review A*, vol. 87, p. 42302, 2013, doi: 10.1103/PhysRevA.87.042302.

[59] C. Gidney, 'Halving the cost of quantum addition', *Quantum*, vol. 2, p. 74, 2018, doi: 10.22331/q-2018-06-18-74.

[60] J. van de Wetering, 'ZX-Calculus for the Working Quantum Computer Scientist'. [Online]. Available: https://arxiv.org/abs/2012.13966

[61] A. Kissinger and J. van de Wetering, *Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation*. 2026. [Online]. Available: https://github.com/zxcalc/book/releases/tag/v1.3.0

[62] M. Backens, 'Making the Stabilizer ZX-Calculus Complete for Scalars', in *Proceedings of the 12th International Workshop on Quantum Physics and Logic*, in Electronic Proceedings in Theoretical Computer Science, vol. 195. 2015, pp. 17–32. doi: 10.4204/EPTCS.195.2.

[63] S. X. Cui, D. Gottesman, and A. Krishna, 'Diagonal Gates in the Clifford Hierarchy', *Physical Review A*, vol. 95, no. 1, p. 12329, 2017, doi: 10.1103/PhysRevA.95.012329.

[64] R. Duncan and S. Perdrix, 'Graph States and the Necessity of Euler Decomposition', in *Mathematical Theory and Computational Practice*, K. Ambos-Spies,

B. Löwe, and W. Merkle, Eds, in Lecture Notes in Computer Science, vol. 5635. Springer, 2009, pp. 167–177. doi: 10.1007/978-3-642-03073-4_18.

[65] R. Vilmart, 'A Near-Minimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics', in *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, 2019, pp. 1–10. doi: 10.1109/LICS.2019.8785765.

[66] M. Backens, 'The ZX-Calculus is Complete for Stabilizer Quantum Mechanics', *New Journal of Physics*, vol. 16, no. 9, p. 93021, 2014, doi: 10.1088/1367-2630/16/9/093021.

[67] A. Cowtan, S. Dilkes, R. Duncan, W. Simmons, and S. Sivarajah, 'Phase Gadget Synthesis for Shallow Circuits', in *Proceedings of the 16th International Conference on Quantum Physics and Logic*, in Electronic Proceedings in Theoretical Computer Science, vol. 318. 2020, pp. 213–228. doi: 10.4204/EPTCS.318.13.

[68] D. E. Gottesman, 'Stabilizer Codes and Quantum Error Correction', Doctoral dissertation, 1997. doi: 10.7907/rzr7-dt72.

[69] M. Hein, J. Eisert, and H. J. Briegel, 'Multi-Party Entanglement in Graph States', *Physical Review A*, vol. 69, no. 6, p. 62311, 2004, doi: 10.1103/PhysRevA.69.062311.

[70] M. Van den Nest, J. Dehaene, and B. De Moor, 'Graphical Description of the Action of Local Clifford Transformations on Graph States', *Physical Review A*, vol. 69, no. 2, p. 22316, 2004, doi: 10.1103/PhysRevA.69.022316.

[71] T. McElvanney and M. Backens, 'Complete Flow-Preserving Rewrite Rules for MBQC Patterns with Pauli Measurements', in *Proceedings 19th International Conference on Quantum Physics and Logic*, in Electronic Proceedings in Theoretical Computer Science, vol. 394. 2023, pp. 66–82. doi: 10.4204/EPTCS.394.5.

[72] H. Bombin, D. Litinski, N. Nickerson, F. Pastawski, and S. Roberts, 'Unifying flavors of fault tolerance with the ZX calculus', *Quantum*, vol. 8, p. 1379, June 2024, doi: 10.22331/q-2024-06-18-1379.

[73] B. Rodatz, B. Poór, and A. Kissinger, 'Fault Tolerance by Construction'. [Online]. Available: https://arxiv.org/abs/2506.17181

[74] A. Kissinger and J. van de Wetering, 'ZX-Flow: A Flexible Criterion for Deterministic Computation with ZX-Diagrams'. [Online]. Available: https://arxiv.org/abs/2603.09580

[75] A. Peruzzo *et al.*, 'A Variational Eigenvalue Solver on a Photonic Quantum Processor', *Nature Communications*, vol. 5, p. 4213, 2014, doi: 10.1038/ncomms5213.

[76] E. Farhi, J. Goldstone, and S. Gutmann, 'A Quantum Approximate Optimization Algorithm'. 2014. doi: 10.48550/arXiv.1411.4028.

[77] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, 'Quantum Circuit Learning', *Physical Review A*, vol. 98, no. 3, p. 32309, 2018, doi: 10.1103/PhysRevA.98.032309.

[78] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, 'Parameterized Quantum Circuits as Machine Learning Models', *Quantum Science and Technology*, vol. 4, no. 4, p. 43001, 2019, doi: 10.1088/2058-9565/ab4eb5.

[79] M. Larocca, N. Ju, D. García-Martín, P. J. Coles, and M. Cerezo, 'Theory of Overparametrization in Quantum Neural Networks', *Nature Computational Science*, vol. 3, pp. 542–551, 2023, doi: 10.1038/s43588-023-00467-6.

[80] T. Haug, K. Bharti, and M. S. Kim, 'Capacity and Quantum Geometry of Parametrized Quantum Circuits', *PRX Quantum*, vol. 2, no. 4, p. 40309, 2021, doi: 10.1103/PRXQuantum.2.040309.

[81] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, 'Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus', *PRX Quantum*, vol. 3, no. 1, p. 10313, 2022, doi: 10.1103/PRXQuantum.3.010313.

[82] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, 'Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms', *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, 2019, doi: 10.1002/qute.201900070.

[83] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, 'Evaluating Analytic Gradients on Quantum Hardware', *Physical Review A*, vol. 99, no. 3, p. 32331, 2019, doi: 10.1103/PhysRevA.99.032331.

[84] T. Stollenwerk and S. Hadfield, 'Diagrammatic Analysis for Parameterized Quantum Circuits', in *Proceedings of the 19th International Conference on Quantum Physics and Logic*, in Electronic Proceedings in Theoretical Computer Science, vol. 394. 2023, pp. 262–301. doi: 10.4204/EPTCS.394.15.

[85] T. Peham, L. Burgholzer, and R. Wille, 'Equivalence Checking of Parameterized Quantum Circuits', in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, 2023, pp. 702–708. doi: 10.1145/3566097.3567932.

[86] H. Miller-Bakewell, 'Finite Verification of Infinite Families of Diagram Equations', in *Proceedings 16th International Conference on Quantum Physics and Logic*, in Electronic Proceedings in Theoretical Computer Science, vol. 318. 2020, pp. 27–52. doi: 10.4204/EPTCS.318.3.

[87] V. Vandaele, S. Perdrix, and C. Vuillot, 'Optimal Number of Parametrized Rotations and Hadamard Gates in Parametrized Clifford Circuits with Non-repeated Parameters'. [Online]. Available: https://arxiv.org/abs/2407.07846

[88] R. Raussendorf and H. J. Briegel, 'A One-Way Quantum Computer', *Physical Review Letters*, vol. 86, no. 22, pp. 5188–5191, 2001, doi: 10.1103/PhysRevLett.86.5188.

[89] R. Raussendorf, D. E. Browne, and H. J. Briegel, 'Measurement-Based Quantum Computation on Cluster States', *Physical Review A*, vol. 68, no. 2, p. 22312, 2003, doi: 10.1103/PhysRevA.68.022312.

[90] M. Backens, H. Miller-Bakewell, G. de Felice, L. Lobski, and J. van de Wetering, 'There and Back Again: A Circuit Extraction Tale', *Quantum*, vol. 5, p. 421, 2021, doi: 10.22331/q-2021-03-25-421.

[91] D. E. Browne, E. Kashefi, M. Mhalla, and S. Perdrix, 'Generalized Flow and Determinism in Measurement-Based Quantum Computation', *New Journal of Physics*, vol. 9, no. 8, p. 250, 2007, doi: 10.1088/1367-2630/9/8/250.

[92] M. Backens and A. Kissinger, 'ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity', in *Proceedings 15th International Conference on Quantum Physics and Logic*, in Electronic Proceedings in Theoretical Computer Science, vol. 287. 2019, pp. 23–42. doi: 10.4204/EPTCS.287.2.

[93] M. Backens, A. Kissinger, H. Miller-Bakewell, J. van de Wetering, and S. Wolffs, 'Completeness of the ZH-calculus', *Compositionality*, vol. 5, no. 5, 2023, doi: 10.32408/compositionality-5-5.

[94] S. Kuijpers, J. van de Wetering, and A. Kissinger, 'Graphical Fourier Theory and the Cost of Quantum Addition'. [Online]. Available: https://arxiv.org/abs/1904.07551

[95] S. Perdrix and Q. Wang, 'Supplementarity Is Necessary for Quantum Diagram Reasoning', in *41st International Symposium on Mathematical Foundations of Computer Science*, in Leibniz International Proceedings in Informatics, vol. 58. 2016, pp. 76:1–76:14. doi: 10.4230/LIPIcs.MFCS.2016.76.

[96] T. Fösel, M. Y. Niu, F. Marquardt, and L. Li, 'Quantum Circuit Optimization with Deep Reinforcement Learning'. [Online]. Available: https://arxiv.org/abs/2103.07585

[97] L. Moro, M. G. A. Paris, M. Restelli, and E. Prati, 'Quantum Compiling by Deep Reinforcement Learning', *Communications Physics*, vol. 4, p. 178, 2021, doi: 10.1038/s42005-021-00684-3.

[98] R. Yeung, 'Diagrammatic Design and Study of Ansätze for Quantum Machine Learning', Master's thesis, 2020. [Online]. Available: https://arxiv.org/abs/2011.11073

[99] C. Zhao and X.-S. Gao, 'Analyzing the Barren Plateau Phenomenon in Training Quantum Neural Networks with the ZX-Calculus', *Quantum*, vol. 5, p. 466, 2021, doi: 10.22331/q-2021-06-04-466.

[100] I. Sutskever, O. Vinyals, and Q. V. Le, 'Sequence to Sequence Learning with Neural Networks', in *Advances in Neural Information Processing Systems*, 2014. [Online]. Available: https://papers.nips.cc/paper_files/paper/2014/hash/5a18e133cbf9f257297f410bb7eca942-Abstract.html

[101] D. Bahdanau, K. Cho, and Y. Bengio, 'Neural Machine Translation by Jointly Learning to Align and Translate', in *International Conference on Learning Representations*, 2015.

[102] A. Vaswani *et al.*, 'Attention Is All You Need', in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[103] G. Lample and F. Charton, 'Deep Learning for Symbolic Mathematics', in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=S1eZYeHFDS

[104] N. de Beaudrap, X. Bian, and Q. Wang, 'Fast and Effective Techniques for T-Count Reduction via Spider Nest Identities', in *15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020)*, S. T. Flammia, Ed., in Leibniz International Proceedings in Informatics (LIPIcs), vol. 158. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020, pp. 11:1–11:23. doi: 10.4230/LIPIcs.TQC.2020.11.

[105] D. Variš and O. Bojar, 'Sequence Length Is a Domain: Length-Based Overfitting in Transformer Models', in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 8246–8257. doi: 10.18653/v1/2021.emnlp-main.650.

[106] Z. Shao *et al.*, 'DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models'. [Online]. Available: https://arxiv.org/abs/2402.03300

[107] L. Ouyang *et al.*, 'Training Language Models to Follow Instructions with Human Feedback', in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2022, pp. 27730–27744. doi: 10.52202/068431-2011.

[108] B. Poór, A. B. Khesin, S. M. Li, B. Rodatz, J. van de Wetering, and R. Yeung, 'CSSCat: Scalable Fault-Tolerant CSS State Preparation'. [Online]. Available: https://quantum-compilers.github.io/iwqc2026/talks.html

[109] S. Bravyi, J. A. Latone, and D. Maslov, '6-qubit optimal Clifford circuits', *npj Quantum Information*, vol. 8, no. 1, p. 79, 2022, doi: 10.1038/s41534-022-00583-7.

[110] T. Goubault de Brugière, M. Baboulin, B. Valiron, S. Martiel, and C. Allouche, 'Gaussian Elimination versus Greedy Methods for the Synthesis of Linear Reversible Circuits', *ACM Transactions on Quantum Computing*, vol. 2, no. 3, pp. 1–26, 2021, doi: 10.1145/3474226.

[111] B. Schaeffer and M. Perkowski, 'A Cost Minimization Approach to Synthesis of Linear Reversible Circuits'. [Online]. Available: https://arxiv.org/abs/1407.0070

[112] S. Bravyi, R. Shaydulin, S. Hu, and D. Maslov, 'Clifford Circuit Optimization with Templates and Symbolic Pauli Gates', *Quantum*, vol. 5, p. 580, 2021, doi: 10.22331/q-2021-11-16-580.

[113] IBM Quantum and Qiskit contributors, 'GreedySynthesisClifford'. [Online]. Available: https://quantum.cloud.ibm.com/docs/en/api/qiskit/2.2/qiskit.transpiler.passes.synthesis.hls_plugins.GreedySynthesisClifford

[114] S. Bravyi and D. Maslov, 'Hadamard-Free Circuits Expose the Structure of the Clifford Group', *IEEE Transactions on Information Theory*, vol. 67, no. 7, pp. 4546–4563, 2021, doi: 10.1109/TIT.2021.3081415.

[115] J. E. Christensen, S. F. Jørgensen, A. Pavlogiannis, and J. van de Pol, 'On Exact Sizes of Minimal CNOT Circuits', in *Reversible Computation*, in Lecture Notes in Computer Science, vol. 15716. Springer, 2025, pp. 71–88. doi: 10.1007/978-3-031-97063-4_6.

[116] M. Webster, S. Koutsioumpas, and D. E. Browne, 'Heuristic and Optimal Synthesis of CNOT and Clifford Circuits'. [Online]. Available: https://arxiv.org/abs/2503.14660

[117] I. Shaik and J. van de Pol, 'Optimal Layout-Aware CNOT Circuit Synthesis with Qubit Permutation', in *ECAI 2024*, IOS Press, 2024, pp. 4207–4215. doi: 10.3233/FAIA240993.

[118] I. Shaik and J. van de Pol, 'CNOT-Optimal Clifford Synthesis as SAT', in *28th International Conference on Theory and Applications of Satisfiability Testing (SAT 2025)*, in Leibniz International Proceedings in Informatics (LIPIcs), vol. 341. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, pp. 28:1–28:21. doi: 10.4230/LIPIcs.SAT.2025.28.

[119] J. Cossio, D. L. Bosco, R. Romanello, G. Serra, and C. Piazza, 'AlphaCNOT: Learning CNOT Minimization with Model-Based Planning'. [Online]. Available: https://arxiv.org/abs/2604.13812v1

[120] F. Fürrutter, G. Muñoz-Gil, and H. J. Briegel, 'Quantum Circuit Synthesis with Diffusion Models', *Nature Machine Intelligence*, vol. 6, pp. 515–524, 2024, doi: 10.1038/s42256-024-00831-9.

[121] M. Zinkevich and T. R. Balch, 'Symmetry in Markov Decision Processes and its Implications for Single Agent and Multiagent Learning', in *Proceedings of the Eighteenth International Conference on Machine Learning*, Morgan Kaufmann, 2001, pp. 632–640. [Online]. Available: https://www.cs.cmu.edu/~maz/publications/symmetry7.pdf

[122] E. van der Pol, D. E. Worrall, H. van Hoof, F. A. Oliehoek, and M. Welling, 'MDP Homomorphic Networks: Group Symmetries in Reinforcement Learning', in *Advances in Neural Information Processing Systems*, 2020, pp. 4199–4210. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/2be5f9c2e3620eb73c2972d7552b6cb5-Paper.pdf

[123] R. Zen, J. Olle, L. Colmenarez, M. Puviani, M. Müller, and F. Marquardt, 'Quantum Circuit Discovery for Fault-Tolerant Logical State Preparation with Reinforcement Learning', *Physical Review X*, vol. 15, no. 4, p. 41012, 2025, doi: 10.1103/gqpr-dgz7.

[124] M. Doherty *et al.*, 'Fast Stabilizer State Preparation via AI-Optimized Graph Decimation'. [Online]. Available: https://arxiv.org/abs/2603.17743

[125] D. Kremer, V. Villar, H. Paik, I. Duran, I. Faro, and J. Cruz-Benito, 'Practical and Efficient Quantum Circuit Synthesis and Transpiling with Reinforcement Learning'. [Online]. Available: https://arxiv.org/abs/2405.13196v2

[126] R. Romanello *et al.*, 'CNOT Minimal Circuit Synthesis: A Reinforcement Learning Approach', in *2025 IEEE International Conference on Quantum Artificial Intelligence (QAI)*, IEEE, 2025, pp. 253–260. doi: 10.1109/QAI63978.2025.00047.

[127] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, 'Deep Sets', in *Advances in Neural Information Processing Systems*, 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html

[128] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, 'Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks', in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 3744–3753. [Online]. Available: https://proceedings.mlr.press/v97/lee19d.html

[129] P. Shaw, J. Uszkoreit, and A. Vaswani, 'Self-Attention with Relative Position Representations', in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 464–468. doi: 10.18653/v1/N18-2074.

[130] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, 'Neural Message Passing for Quantum Chemistry', in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 1263–1272. [Online]. Available: https://proceedings.mlr.press/v70/gilmer17a.html

[131] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, 'Curriculum learning', in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 41–48. doi: 10.1145/1553374.1553380.

[132] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, 'Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey', *Journal of Machine Learning Research*, vol. 21, no. 181, pp. 1–50, 2020, [Online]. Available: http://jmlr.org/papers/volume21/20-212/20-212.pdf

[133] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, 'Reverse Curriculum Generation for Reinforcement Learning', in *Conference on Robot Learning*, 2017, pp. 482–495. [Online]. Available: http://proceedings.mlr.press/v78/florensa17a/florensa17a.pdf

[134] F. Agostinelli, S. McAleer, A. Shmakov, and P. Baldi, 'Solving the Rubik's cube with deep reinforcement learning and search', *Nature Machine Intelligence*, vol. 1, no. 8, pp. 356–363, 2019, doi: 10.1038/s42256-019-0070-z.

[135] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, 'Proximal Policy Optimization Algorithms'. [Online]. Available: https://arxiv.org/abs/1707.06347v2

[136] J. Suarez, 'PufferLib: Making Reinforcement Learning Libraries and Environments Play Nice'. [Online]. Available: https://arxiv.org/abs/2406.12905v1

[137] C. Gidney, 'Inverting Clifford Tableaus'. [Online]. Available: https://algassert.com/post/2002

[138] S. Schneider, L. Burgholzer, and R. Wille, 'A SAT Encoding for Optimal Clifford Circuit Synthesis', in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ACM, 2023, pp. 190–195. doi: 10.1145/3566097.3567929.

[139] S. Bravyi, R. Shaydulin, S. Hu, and D. Maslov, 'Data to accompany Clifford Circuit Optimization with Templates and Symbolic Pauli Gates'. [Online]. Available: https://github.com/rsln-s/Clifford_Circuit_Optimization_with_Templates_and_Symbolic_Pauli_Gates

[140] Y. Mori, H. Hakoshima, and K. Fujii, 'Nontrivial Multiproduct Commutation Relation Toward Reducing T-Count in Sequential Pauli-Based Computation', *PRX Quantum*, vol. 7, no. 2, p. 20345, 2026, doi: 10.1103/qw5z-3bwz.

[141] R. Jozsa and A. Miyake, 'Matchgates and Classical Simulation of Quantum Circuits', *Proceedings of the Royal Society A*, vol. 464, pp. 3089–3106, 2008, doi: 10.1098/rspa.2008.0189.

[142] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, 'Optimal Design for Universal Multiport Interferometers', *Optica*, vol. 3, no. 12, pp. 1460–1465, 2016, doi: 10.1364/OPTICA.3.001460.

[143] A. Dubal, D. Kremer, S. Martiel, V. Villar, D. Wang, and J. Cruz-Benito, 'Pauli Network Circuit Synthesis with Reinforcement Learning'. [Online]. Available: https://arxiv.org/abs/2503.14448

[144] N. Brown, A. Bakhtin, A. Lerer, and Q. Gong, 'Combining Deep Reinforcement Learning and Search for Imperfect-Information Games', in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 17057–17069. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/c61f571dbd2fb949d3fe5ae1608dd48b-Abstract.html

[145] A. L. Jones, 'Scaling Scaling Laws with Board Games'. [Online]. Available: https://arxiv.org/abs/2104.03113

[146] D. Bluvstein *et al.*, 'A fault-tolerant neutral-atom architecture for universal quantum computation', *Nature*, vol. 649, pp. 39–46, 2026, doi: 10.1038/s41586-025-09848-5.

[147] B. Rodatz, B. Poór, and A. Kissinger, 'Floquetifying stabiliser codes with distance-preserving rewrites', *Quantum*, vol. 10, p. 2202, 2026, doi: 10.22331/q-2026-09-03-2202.

[148] A. Novikov *et al.*, 'AlphaEvolve: A coding agent for scientific and algorithmic discovery'. [Online]. Available: https://arxiv.org/abs/2506.13131

[149] J. Hartford, D. Graham, K. Leyton-Brown, and S. Ravanbakhsh, 'Deep Models of Interactions Across Sets', in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 1909–1918. [Online]. Available: https://proceedings.mlr.press/v80/hartford18a.html