Graph state preparation on arbitrary hardware

Tristan Cam,
in collaboration with Simon Martiel, Cyril Gavoille, Yvan Le Borgne

July 4, 2025
Reversible Computation 2025

Motivations

Introduction: quantum computing

- Some tasks are easier for a quantum computer;
- some quantum circuits are hard to simulate on a classical computer;
- high entanglement is a key ingredient for quantum speedup.

Introduction: quantum computing

- Some tasks are easier for a quantum computer;
- some quantum circuits are hard to simulate on a classical computer;
- high entanglement is a key ingredient for quantum speedup.

$\Rightarrow$ graph state framework

nearly hard to simulate; good measure of entanglement: rank-width

Introduction: preparing graph states

Graph state $|G\rangle$ on n qubits :

Stabilizer state represented by an undirected graph G on n vertices.

Recipe to prepare $|G\rangle$ on your hardware H:

Introduction: preparing graph states

Graph state $|G\rangle$ on n qubits :
Stabilizer state represented by an undirected graph G on n vertices.

Recipe to prepare $|G\rangle$ on your hardware H:

- First prepare the state $|+\rangle^{\otimes n} = |\overline{K_n}\rangle$ (*i.e.* a wall of Hadamard),

Introduction: preparing graph states

Graph state $|G\rangle$ on n qubits :
Stabilizer state represented by an undirected graph G on n vertices.

Recipe to prepare $|G\rangle$ on your hardware H:

- First prepare the state $|+\rangle^{\otimes n} = |\overline{K_n}\rangle$ (*i.e.* a wall of Hadamard),
- then for each edge $\{u, v\} \in E(G)$ *in any order* apply a CZ between qubits u, v.

Introduction: preparing graph states

Graph state $|G\rangle$ on n qubits :
Stabilizer state represented by an undirected graph G on n vertices.

Recipe to prepare $|G\rangle$ on your hardware H:

- First prepare the state $|+\rangle^{\otimes n} = |\overline{K_n}\rangle$ (*i.e.* a wall of Hadamard),
- then for each edge $\{u, v\} \in E(G)$ *in any order* apply a CZ between qubits u, v.

$\Rightarrow$ you used $|E(G)|$ two-qubit gates.

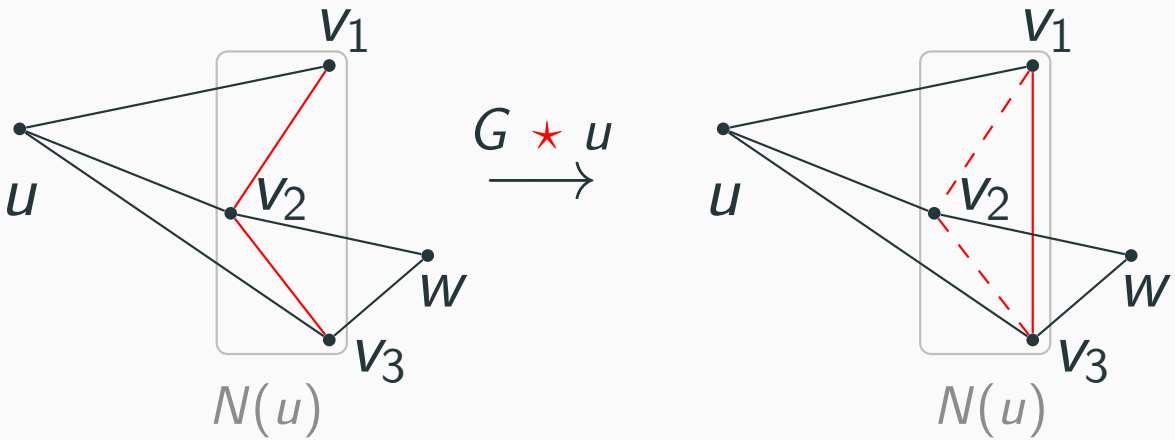
Given all-to-all connectivity

Local complementation(s)

Local complementation: as a graph operation

Definition

The *local complementation* of G according to u, written $G \star u$, is a graph which has the same vertices as G, but all the neighbors $v_1, v_2, \ldots$ of u are connected in $G \star u$ if and only if they are not connected in G. All other edges are unchanged.

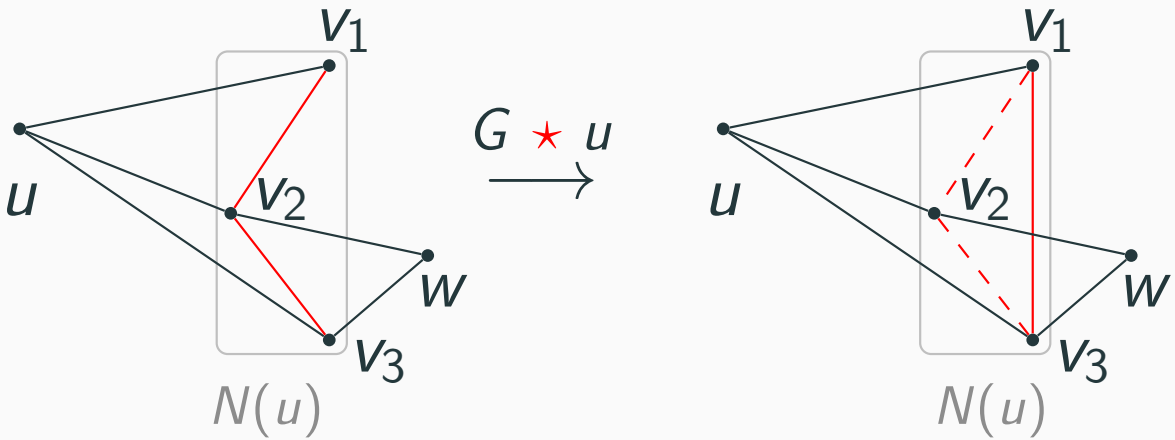


$$G \star u \star u = G$$

Local complementation: as a graph operation

Definition

The *local complementation* of G according to u, written $G \star u$, is a graph which has the same vertices as G, but all the neighbors $v_1, v_2, \ldots$ of u are connected in $G \star u$ if and only if they are not connected in G. All other edges are unchanged.



$$G \star u \star u = G$$

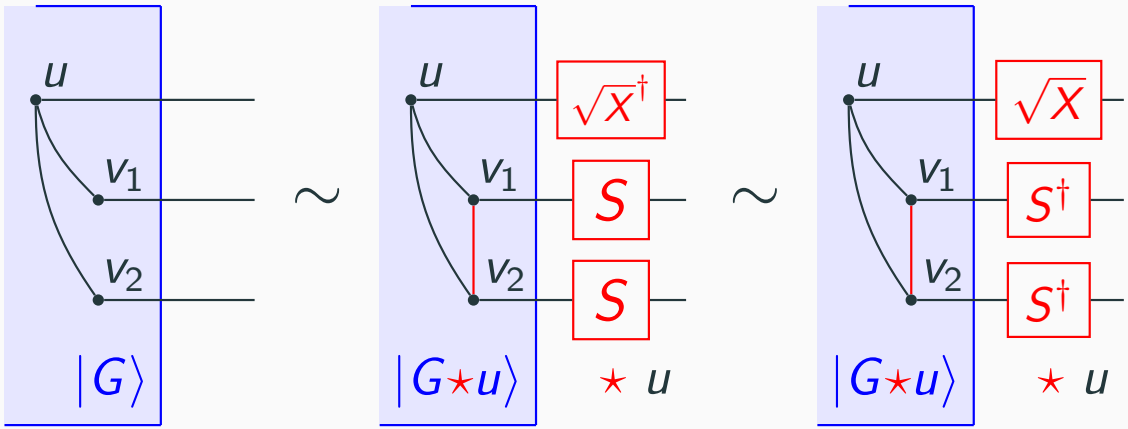
Note: this preserves rank-width (*i.e.* entanglement).

Local complementation: as a circuit operation

Proposition

Let $|G\rangle$ be a graph state, u a vertex of G and N_u the set of neighbors of u in G. Performing a local complementation on $|G\rangle$ according to u is done by:

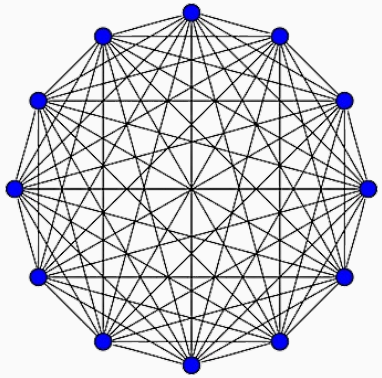
- *adding a $\sqrt{X}^\dagger$ gate on the qubit u,*
- *adding a S gate on every qubit in N_u.*



Replacing CZ by single qubit gate = less error

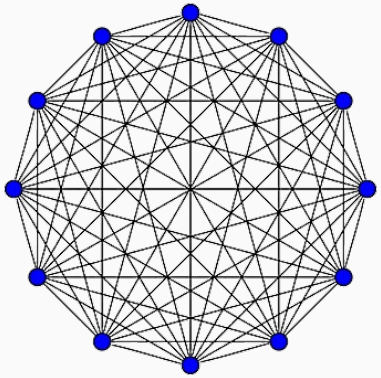
Local complementation: in practice

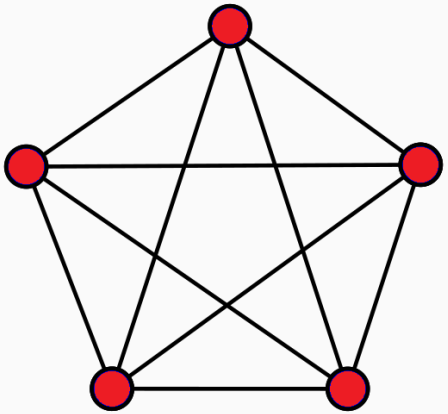
Your hardware: H, with all-to-all connectivity.



Local complementation: in practice

Your hardware: H, with all-to-all connectivity.



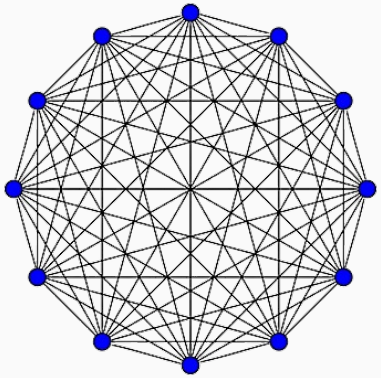


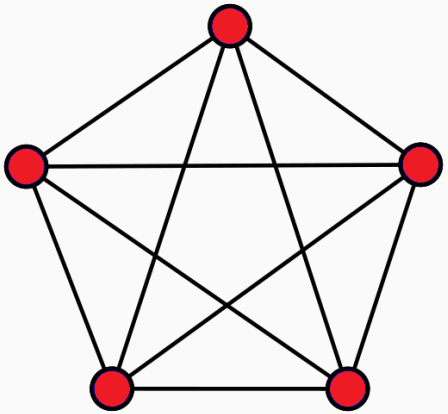
n=5

You want to prepare $|G\rangle$ with $G = K_5$

Local complementation: in practice

Your hardware: H, with all-to-all connectivity.





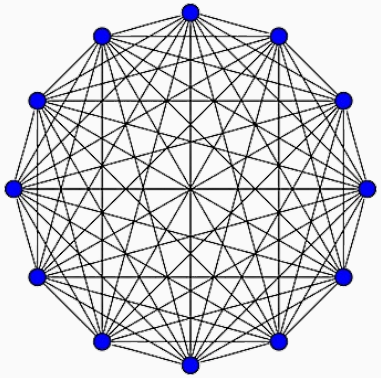
n=5

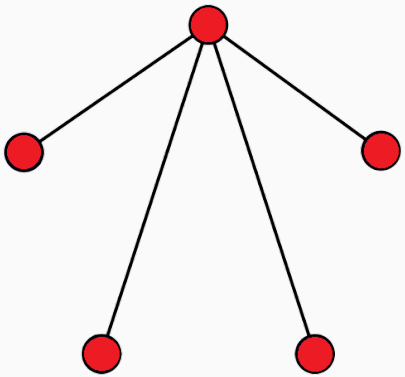
You want to prepare $|G\rangle$ with $G = K_5$

Naively uses $\theta(n^2)$ CZ gates.

Local complementation: in practice

Your hardware: H, with all-to-all connectivity.

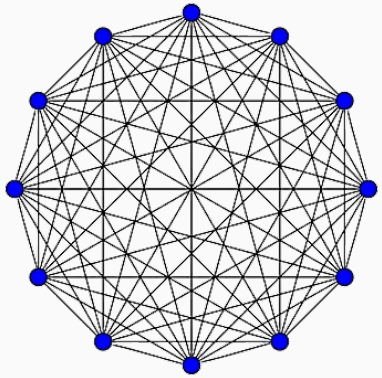


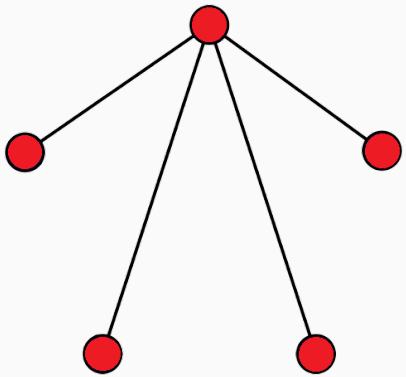


Notice that $K_5 \sim_{loc} K_{1,4}$

Local complementation: in practice

Your hardware: H, with all-to-all connectivity.





Notice that $K_5 \sim_{loc} K_{1,4}$

Preparing $|K_{1,4}\rangle$ uses $\mathcal{O}(n)$ CZ gates, then apply one local complementation to get $|G\rangle$.

One local complementation emulated 6 "virtual" CZ.

Local complementation: in practice

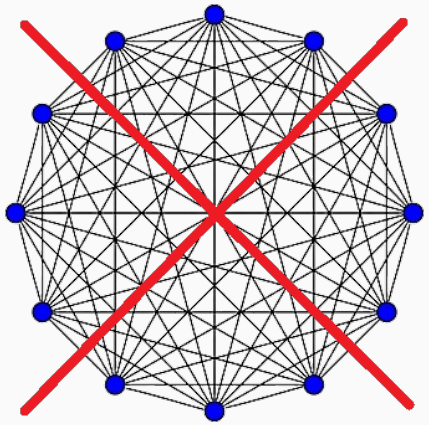
$\Rightarrow$ Local complementation enables optimizations in gate count

BUT...

Local complementation: in practice

$\Rightarrow$ Local complementation enables optimizations in gate count

BUT...

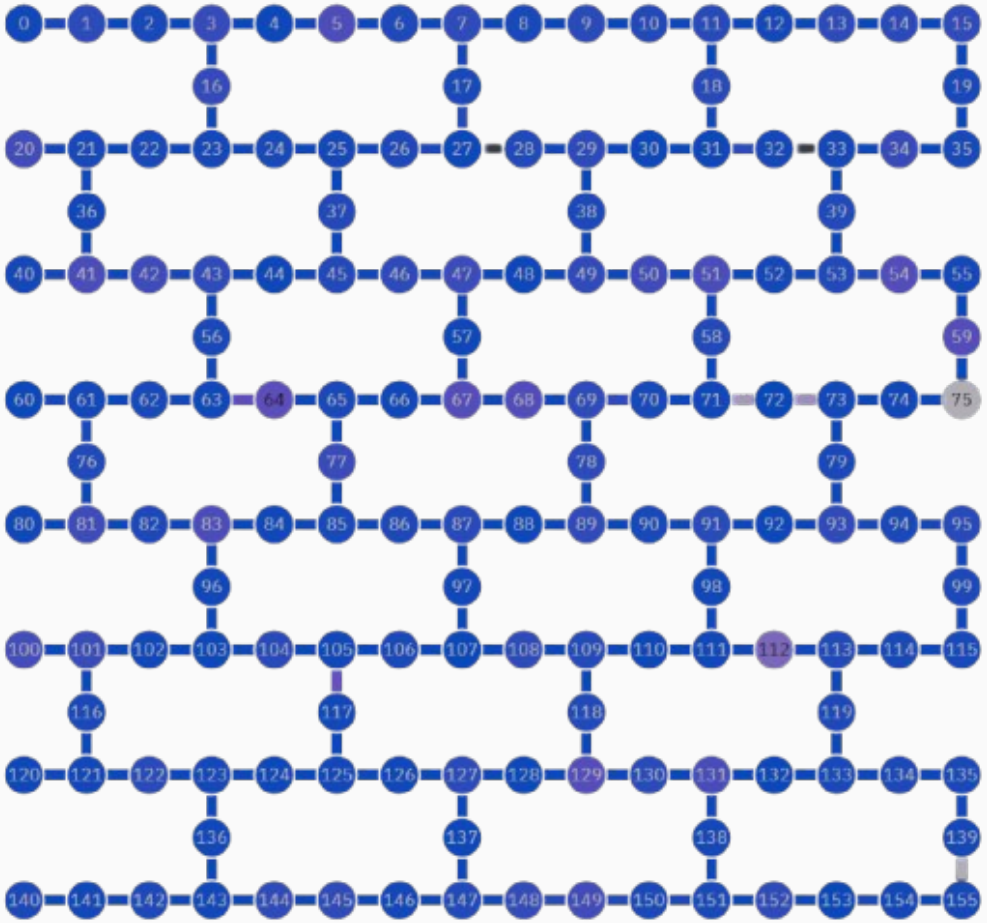


(your hardware may not be a complete graph.)

How about on arbitrary hardware?

Arbitrary hardware

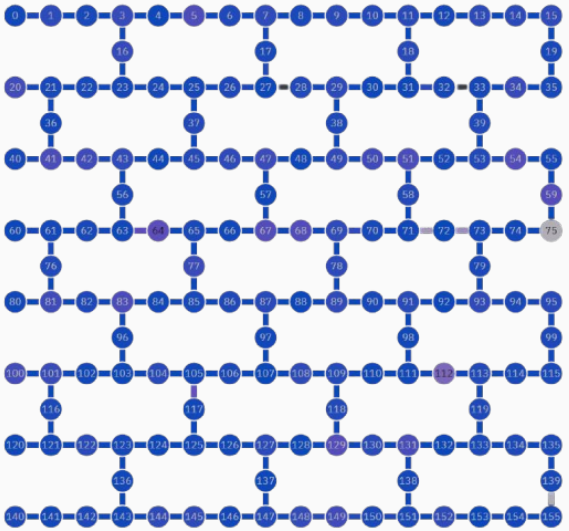
Given a graph H : your hardware.



IBM's 156 qubit processor, Heron r2

Arbitrary hardware

Given a graph H : your hardware.

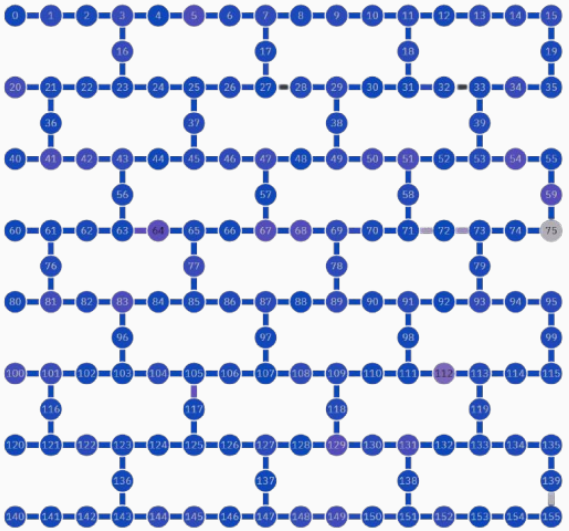


Problem

Which graph states can be prepared on H by only:

Arbitrary hardware

Given a graph H : your hardware.



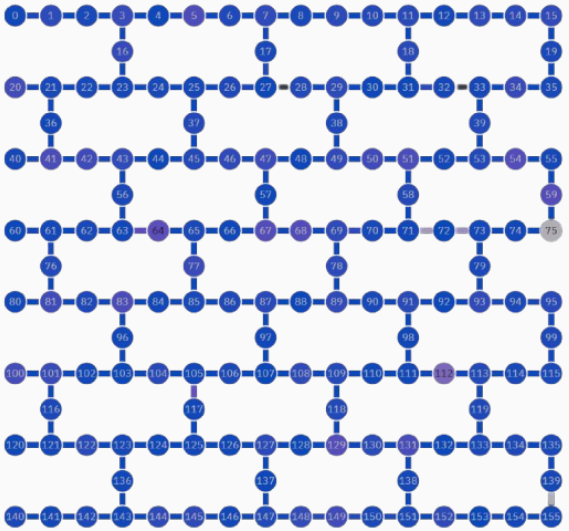
Problem

Which graph states can be prepared on H by only:

- adding CZ along edges of H (toggling),

Arbitrary hardware

Given a graph H : your hardware.



Problem

Which graph states can be prepared on H by only:

- adding CZ along edges of H (toggling),
- local complementations on vertices of H.

Our results I

Answer: all of them.

Theorem

Let H be a connected graph on n vertices.
Let G be any graph on n vertices.
The state $|G\rangle$ can be prepared by a circuit consisting of CZ gates along edges of H and local complementations.

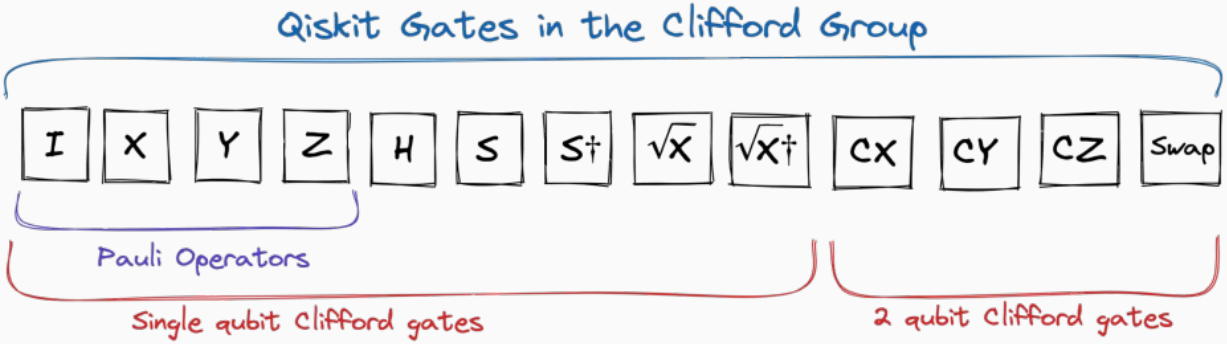
Our method

Useful fact

Proposition (Folklore)

Let H be the connected hardware graph.
Then any Clifford circuit can be implemented with local Clifford gates and CZ along edges of H.

i.e. any Clifford circuit is "compatible" with H if we allow all types of local Cliffords (naively using a $\mathcal{O}(n^3)$ overhead on CZ count).



Goal

C from Proposition,
Contains {CZ, Local Cliffords}.

C' from our method,
Contains {CZ, $\underbrace{S, \sqrt{X}^\dagger}_{\text{loc. comp.}}$}.

polynomial time algorithm

$$C\,|+\rangle \quad = \quad |G\rangle \quad = \quad C'\,|+\rangle$$

Goal

C from Proposition,
Contains {CZ, Local Cliffords}.

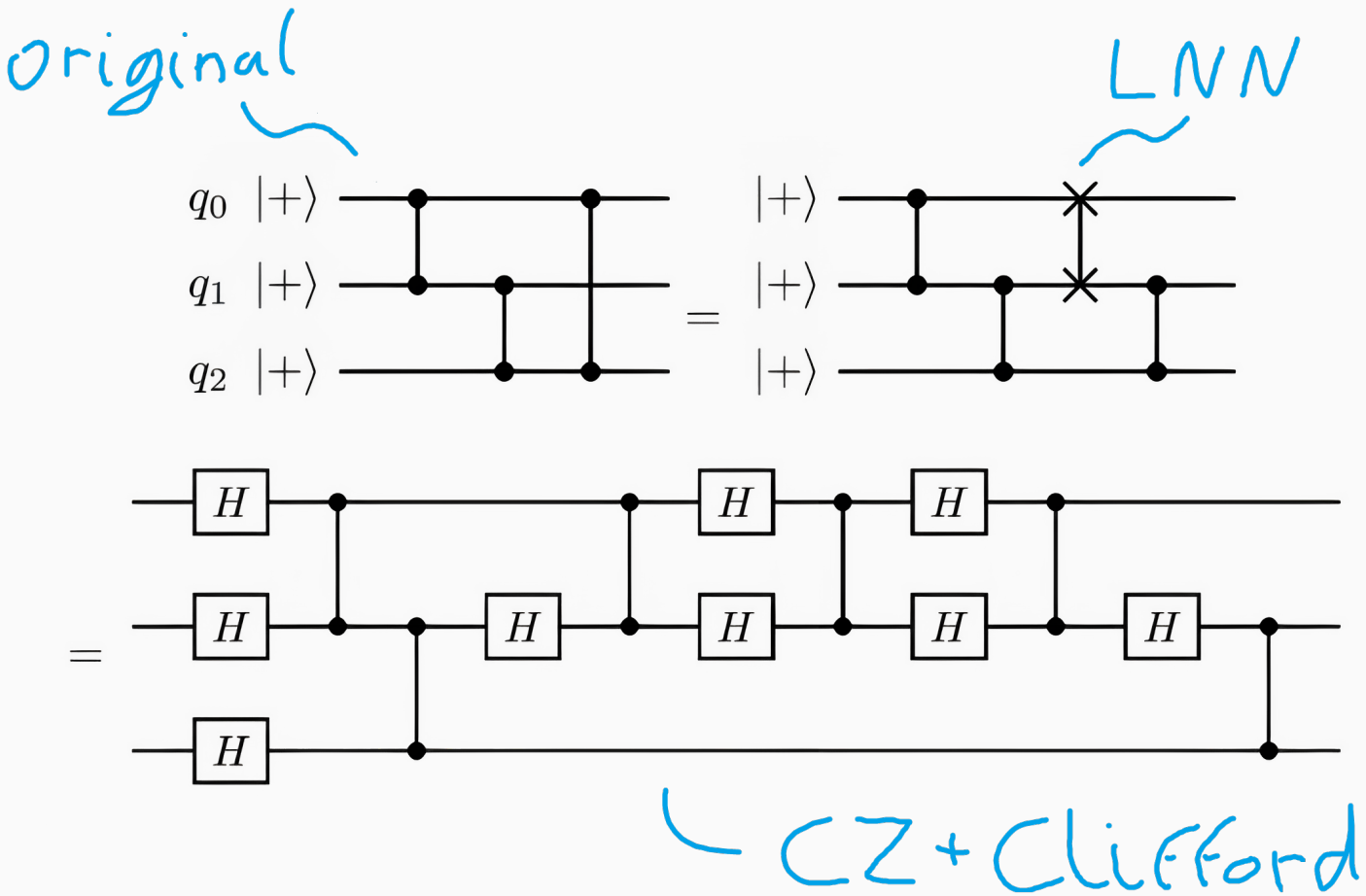
C' from our method,
Contains {CZ, $\underbrace{S, \sqrt{X}^{\dagger}}_{\text{loc. comp.}}$}.

polynomial time algorithm

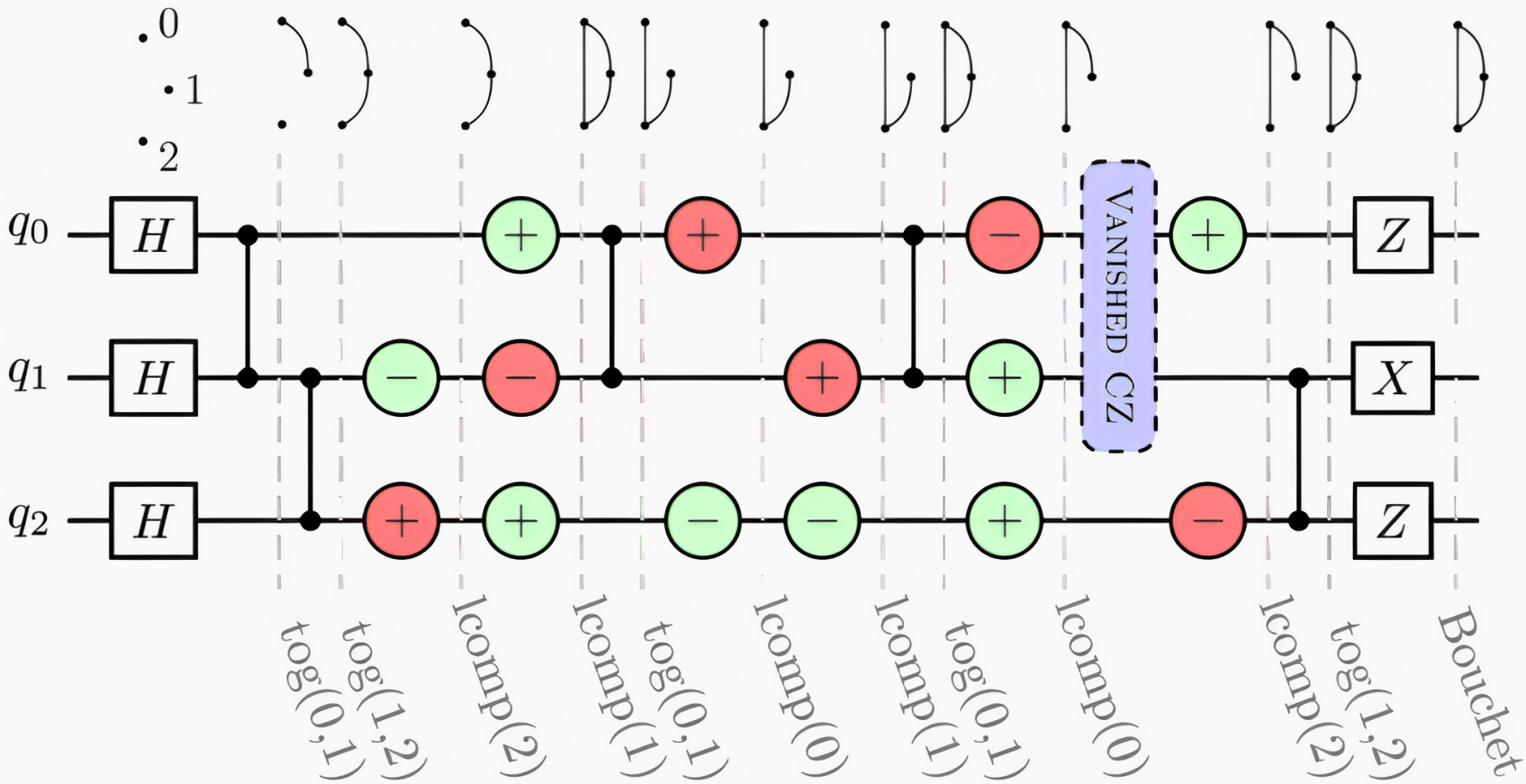
$$C\,|+\rangle \quad = \quad |G\rangle \quad = \quad C'\,|+\rangle$$

$$\#CZ(C) \quad = \quad m \quad \geq \quad \#CZ(C')$$

Overview

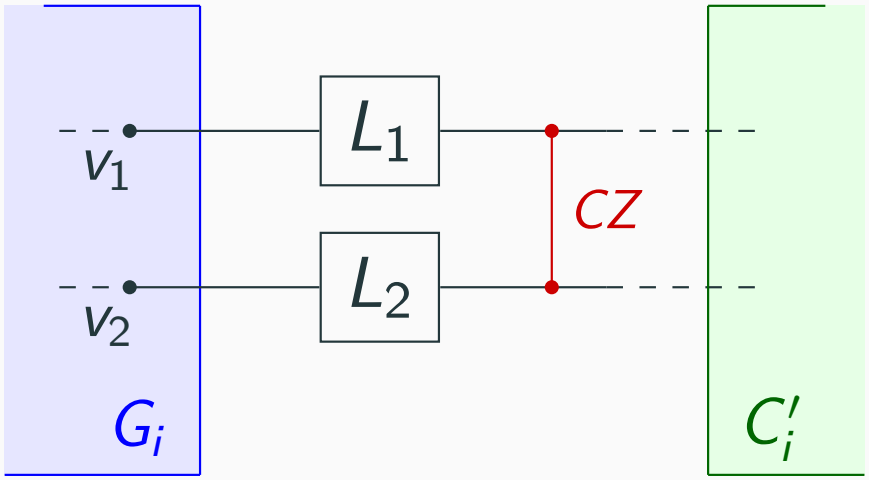

original
LNN
q_0 $|+\rangle$
q_1 $|+\rangle$
q_2 $|+\rangle$
=
$|+\rangle$
$|+\rangle$
$|+\rangle$
=
H
CZ + Clifford

Overview



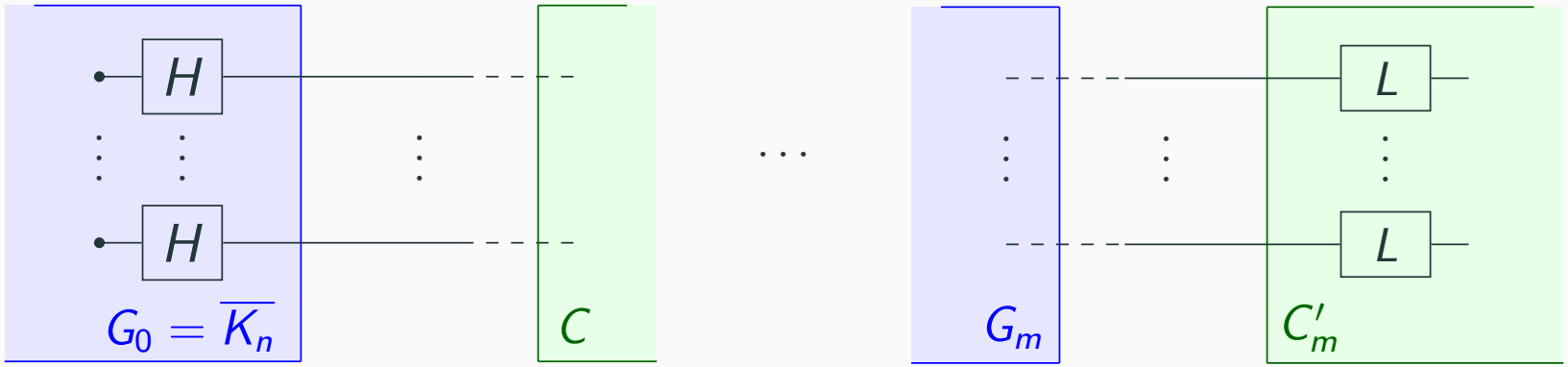
where $\oplus = \boxed{S}$, $\ominus = \boxed{S^\dagger}$, $\oplus = \boxed{\sqrt{X}}$, $\ominus = \boxed{\sqrt{X}^\dagger}$.

Our construction



Sketch of the i-th step.

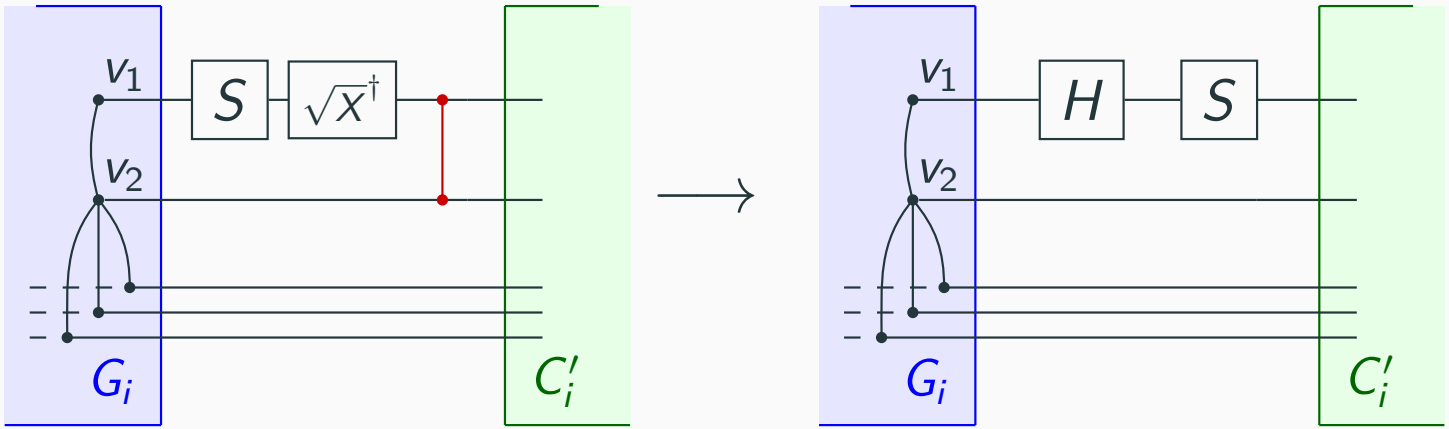
Our construction



First step

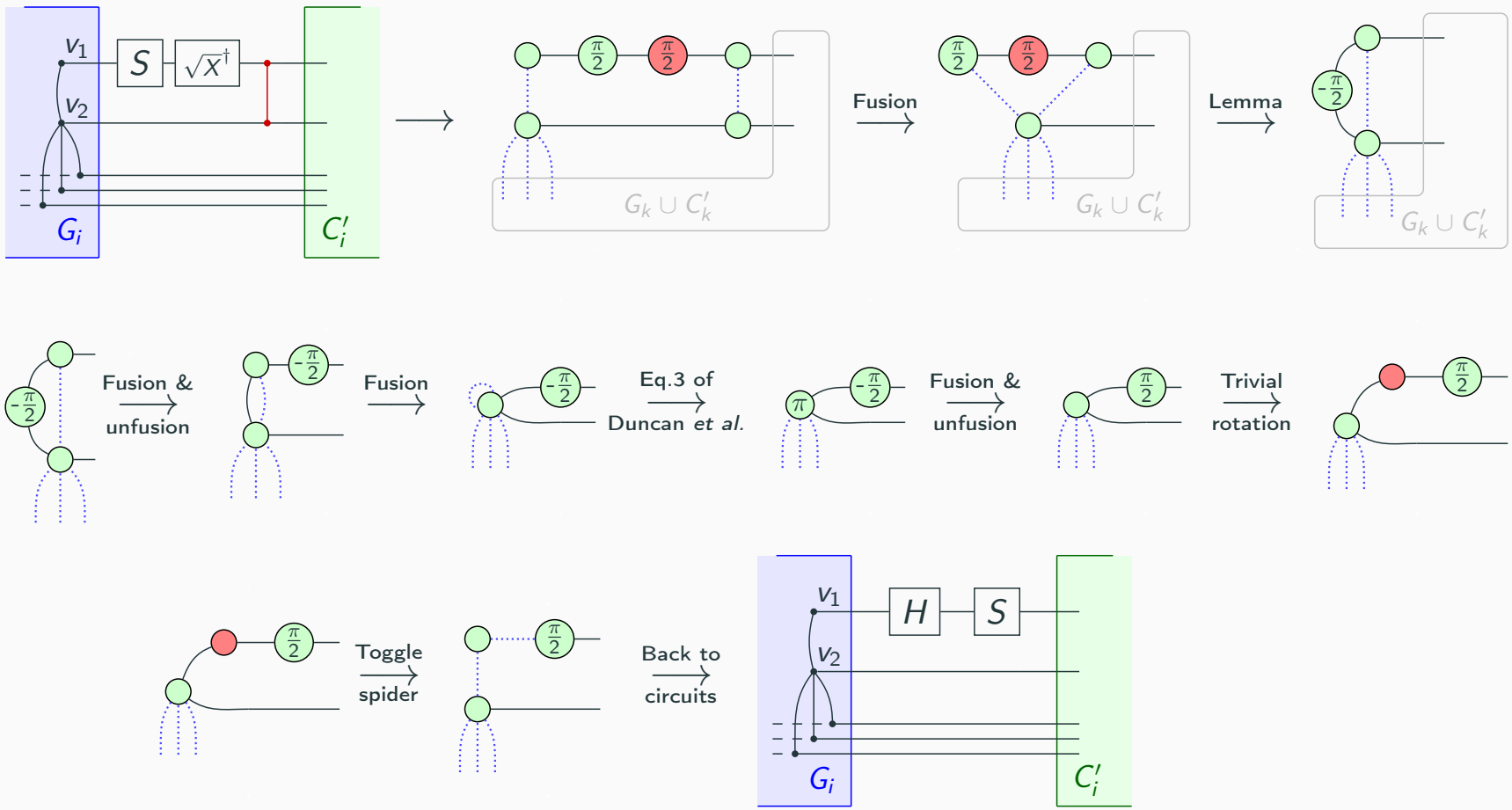
Last step

Preliminary Lemma 1: vanishing CZ

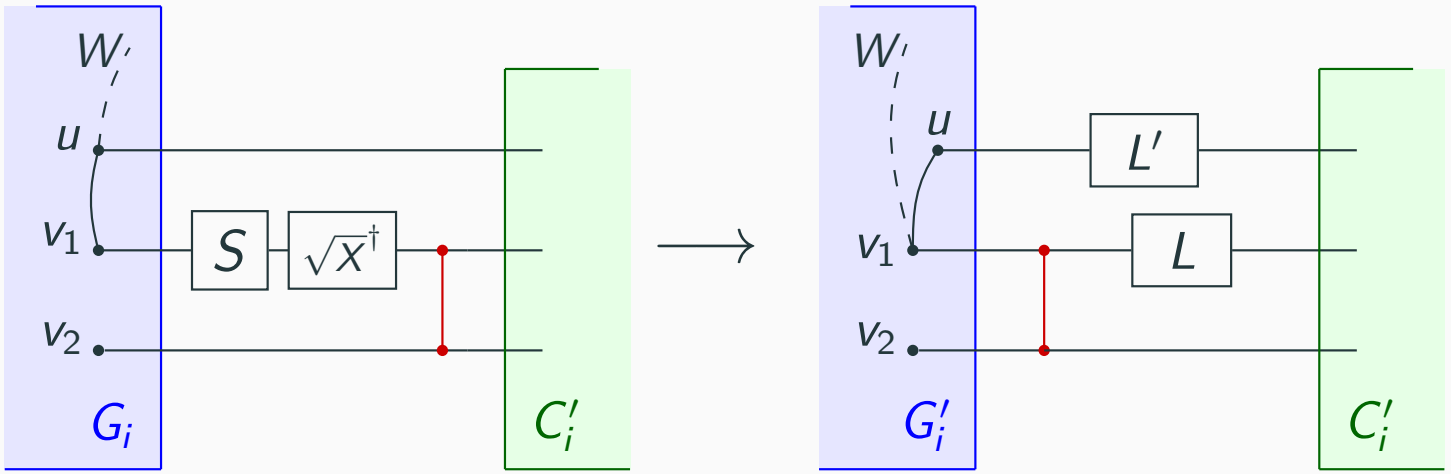


Lemma 1: These two circuits are equivalent

Proof of Lemma 1



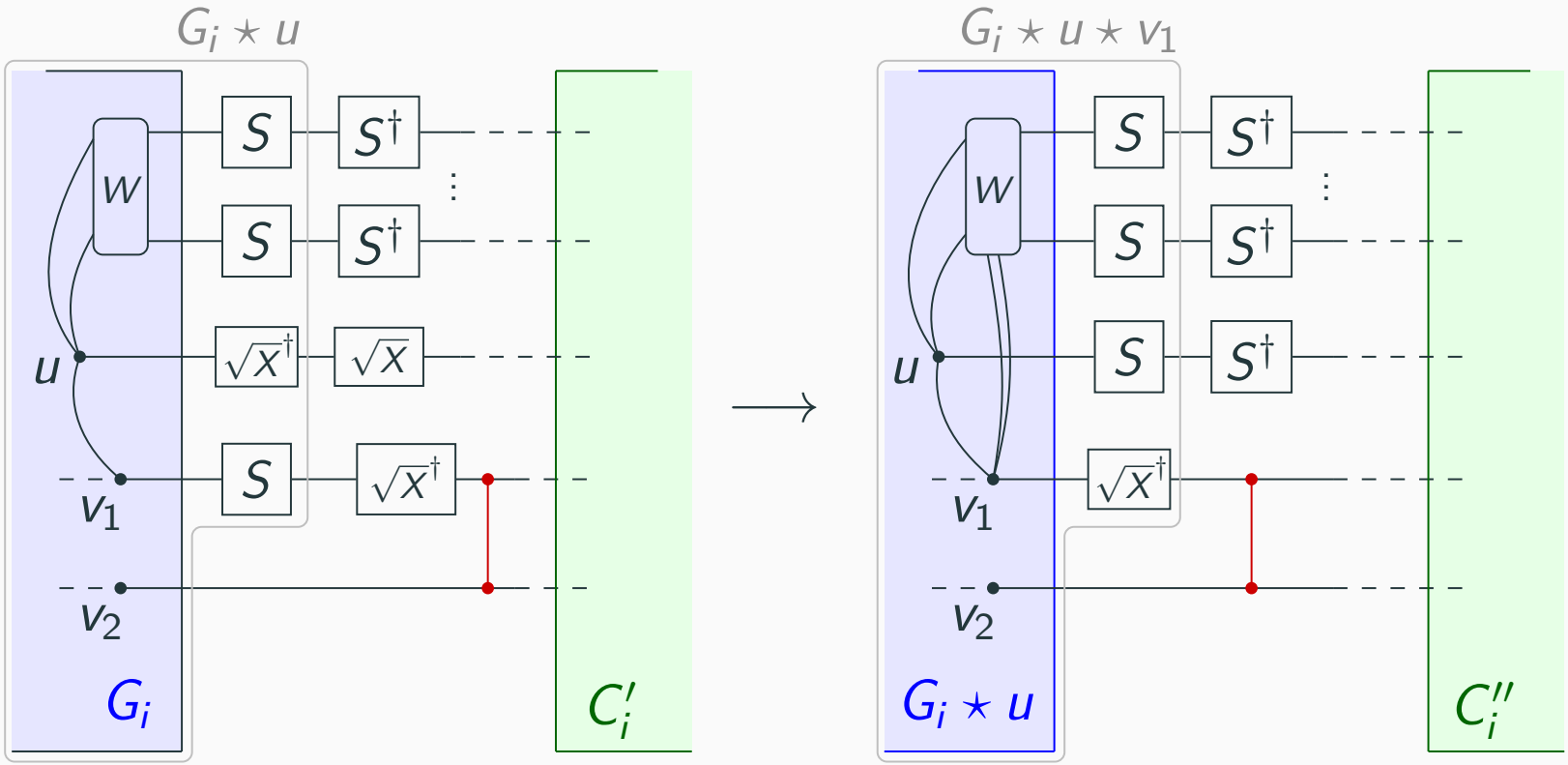
Preliminary Lemma 2: commuting and absorbing the CZ



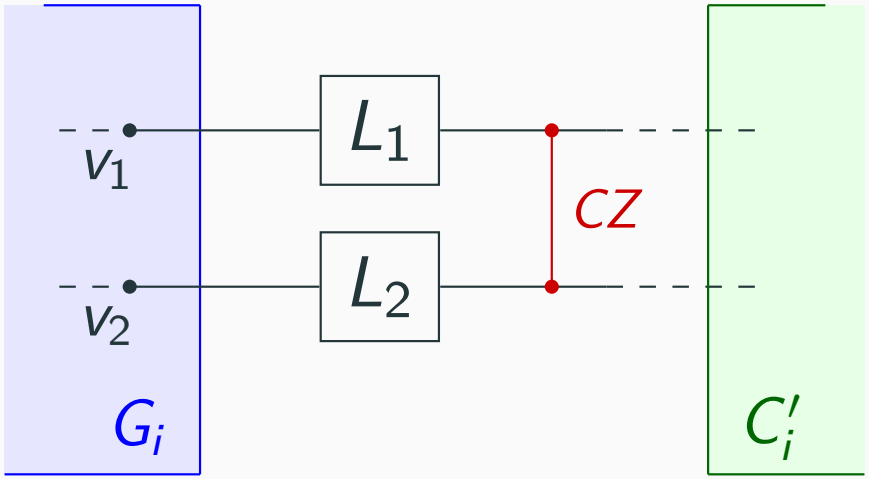
Lemma 2: These two circuits are equivalent

$W = N_u \setminus \{v_1\}$ and L, L' are generic local Clifford gates.

Proof of Lemma 2



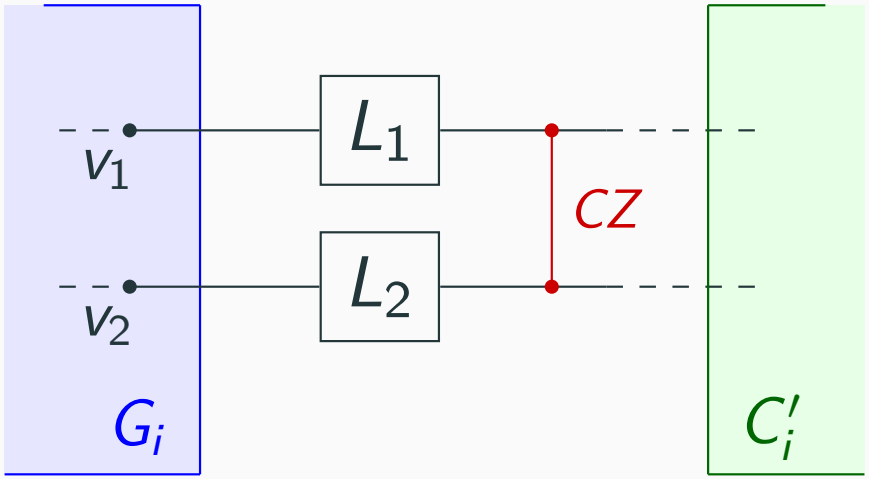
Outline of the proof



Euler decomposition:

$$\exists a_1, b_1, c_1, a_2, b_2, c_2 \in \mathbb{Z}/4\mathbb{Z} :$$
$$L_1 = S^{c_1}\sqrt{X}^{\dagger b_1}S^{a_1} \text{ and } L_2 = S^{c_2}\sqrt{X}^{\dagger b_2}S^{a_2}.$$

Outline of the proof



Euler decomposition:

$$\exists a_1, b_1, c_1, a_2, b_2, c_2 \in \mathbb{Z}/4\mathbb{Z} :$$
$$L_1 = S^{c_1}\sqrt{X}^{\dagger b_1} S^{a_1} \text{ and } L_2 = S^{c_2}\sqrt{X}^{\dagger b_2} S^{a_2}.$$

$$\text{w.l.o.g. } L_1 := \sqrt{X}^{\dagger b_1} S^{a_1},\ L_2 := \sqrt{X}^{\dagger b_2} S^{a_2}.$$

Outline of the proof: three cases

$$L_1 = \sqrt{X}^{\dagger b_1} S^{a_1},\ L_2 = \sqrt{X}^{\dagger b_2} S^{a_2}$$

Three cases:

Outline of the proof: three cases

$$L_1 = \sqrt{X}^{\dagger b_1} S^{a_1},\ L_2 = \sqrt{X}^{\dagger b_2} S^{a_2}$$

Three cases:

1. Both b_1 and b_2 are even.

Outline of the proof: three cases

$$L_1 = \sqrt{X}^{\dagger b_1} S^{a_1},\ L_2 = \sqrt{X}^{\dagger b_2} S^{a_2}$$

Three cases:

1. Both b_1 and b_2 are even.

2. Only one of b_1 and b_2 is odd.

Outline of the proof: three cases

$$L_1 = \sqrt{X}^{\dagger b_1} S^{a_1},\ L_2 = \sqrt{X}^{\dagger b_2} S^{a_2}$$

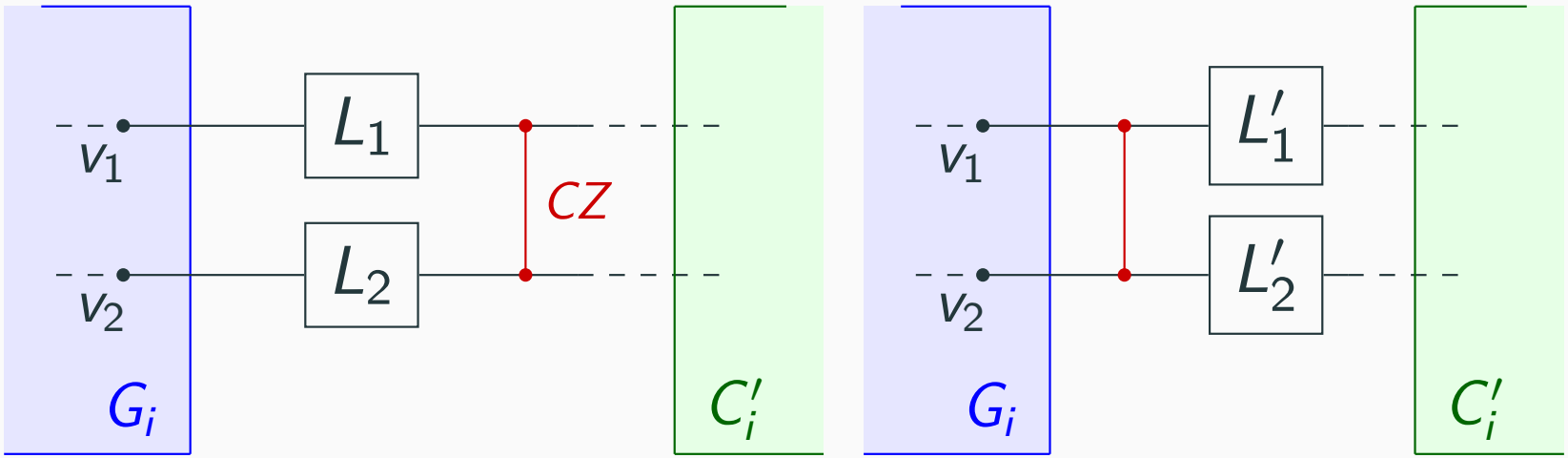
Three cases:

1. Both b_1 and b_2 are even.
2. Only one of b_1 and b_2 is odd.
3. Both b_1 and b_2 are odd.

Outline of the proof: Case 1

1. Both b_1 and b_2 are even.

$$L_1 = X^{b_1/2} S^{a_1},\ L_2 = X^{b_2/2} S^{a_2}$$

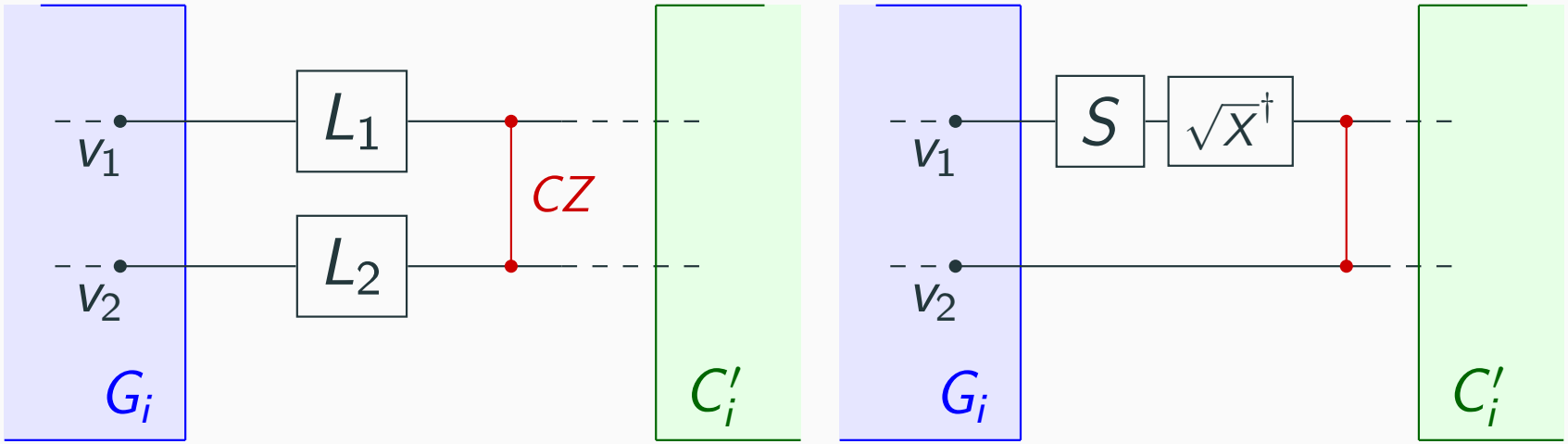


Pauli and S gates: weakly commute with CZ

Outline of the proof: Case 2

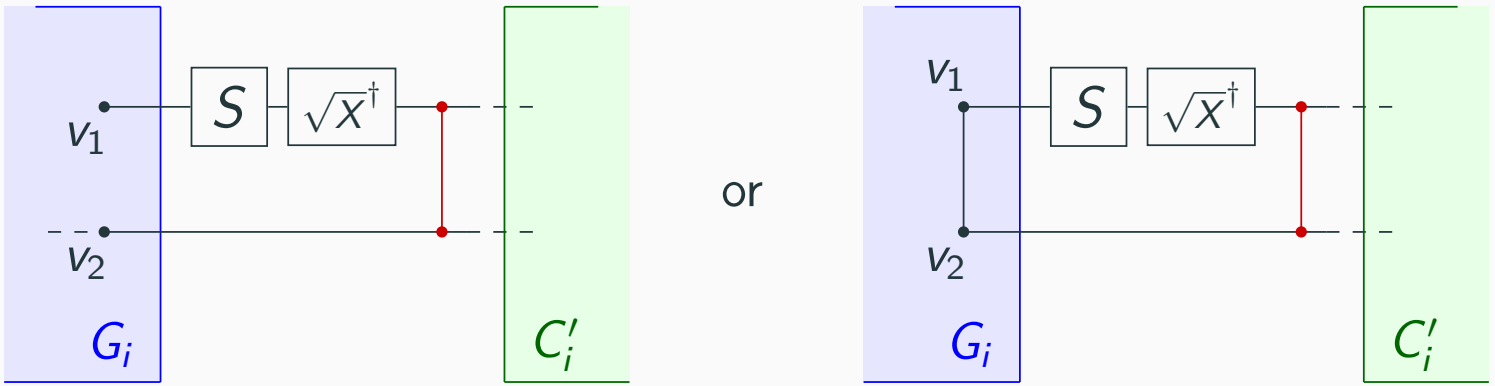
2. Only one of b_1 and b_2 is odd, w.l.o.g. take b_1 odd.

$$L_1 = X^{b_1-1}\sqrt{X}^{\dagger}S^{a_1},\ L_2 = X^{b_2/2}S^{a_2}$$



Pauli and S gates: weakly commute with CZ

Outline of the proof: Case 2



Either:

- v_1 has no neighbor in G_k, or v_1 and v_2 are isolated in G_k

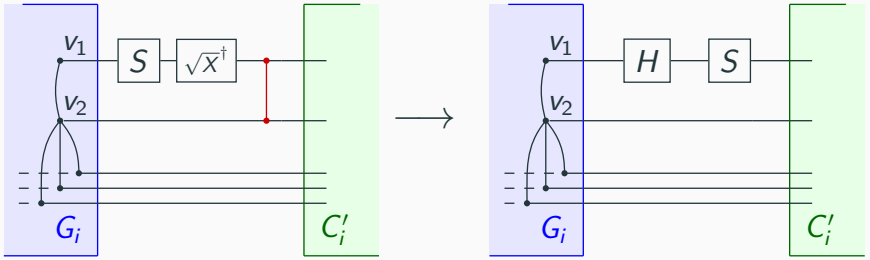
$\Rightarrow$ rewrite subcircuit as 0 or 1 CZ followed by local Cliffords;

Outline of the proof: Case 2

Either:

- v_1 has no neighbor in G_k / v_1, v_2 are each other's only neighbor: rewrite;

- the only neighbor of v_1 in G_k is v_2 and v_2 has at least one neighbor in G_k : Lemma 1;

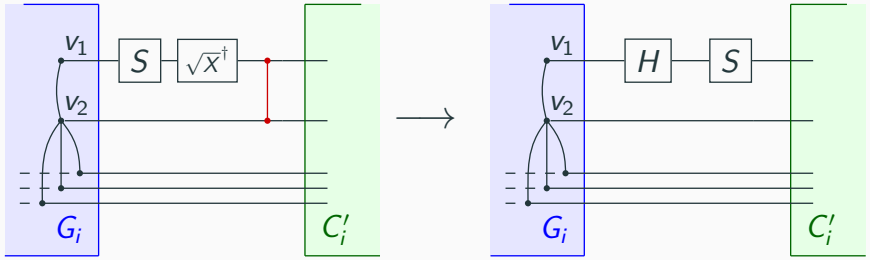


Outline of the proof: Case 2

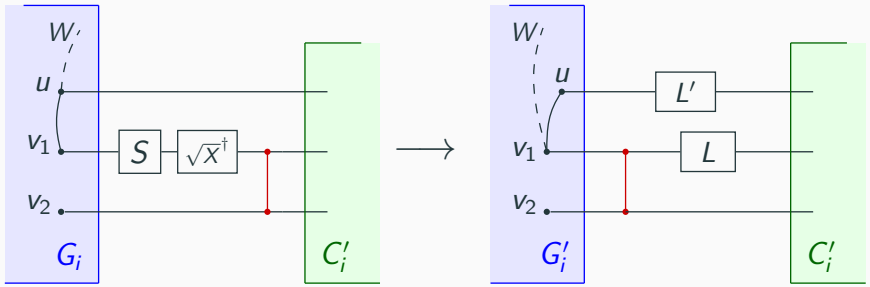
Either:

- v_1 has no neighbor in G_k / v_1, v_2 are each other's only neighbor: rewrite;

- the only neighbor of v_1 in G_k is v_2 and v_2 has at least one neighbor in G_k : Lemma 1;



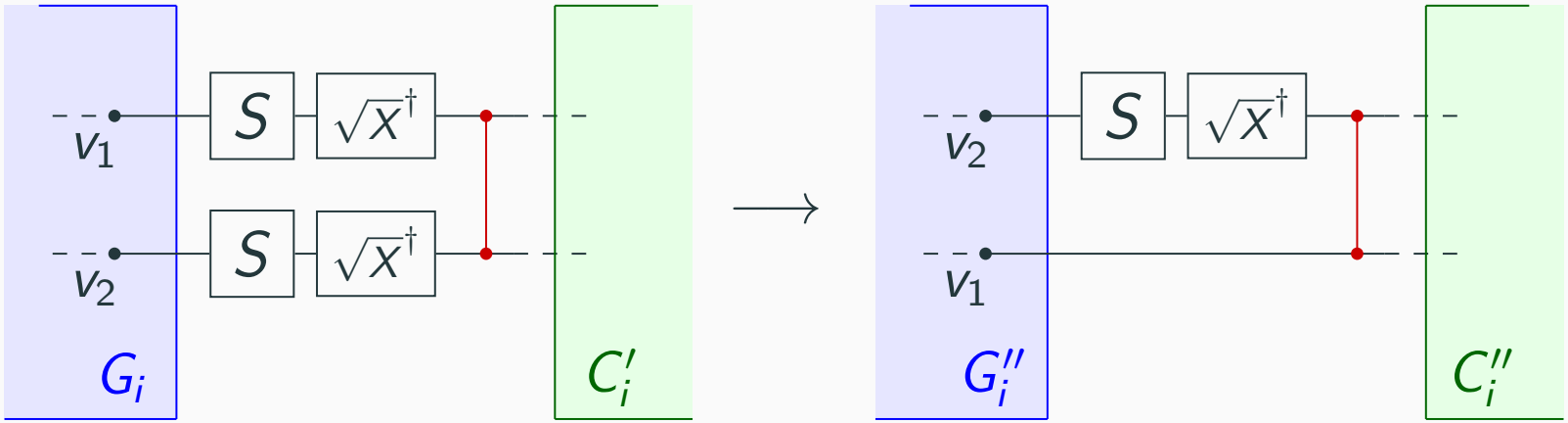
- v_1 has at least one neighbor u in G_k that is not v_2 : Lemma 2.



Outline of the proof: Case 3

3. Both b_1 and b_2 are odd.

$$L_1 = X^{b_1-1}\sqrt{X}^{\dagger}S^{a_1},\ L_2 = X^{b_2-1}\sqrt{X}^{\dagger}S^{a_2}$$

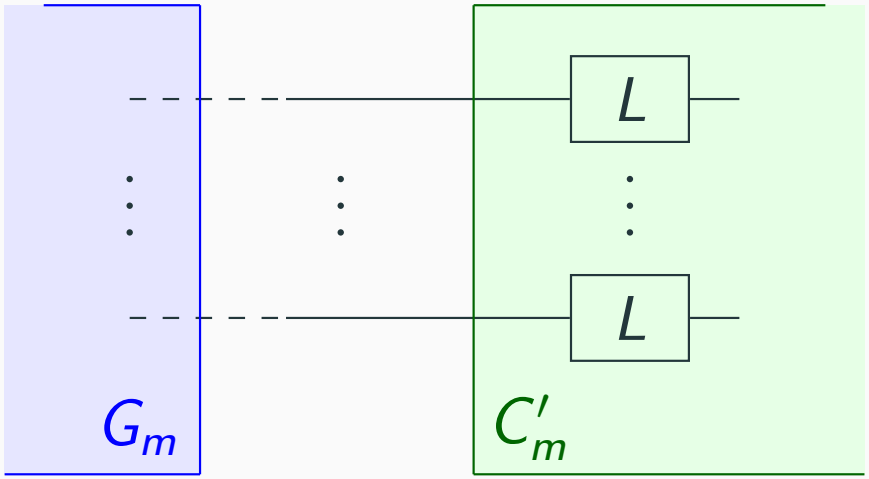


Pauli and S gates: weakly commute with CZ

$\Rightarrow$ Technicalities. Reduce **Case 3** to **Case 2**.

Outline of the proof: final touch

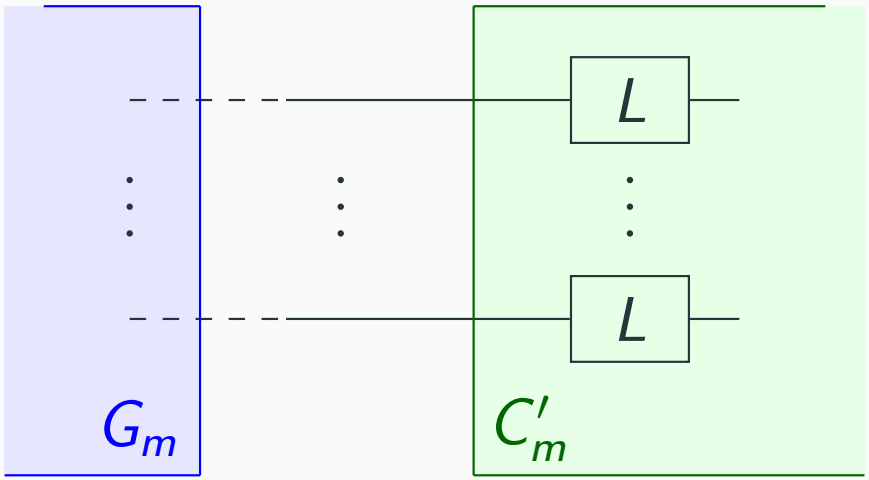
Last step: Van den Nest



Local Cliffords layer $\approx$ finite sequence of l. comp. $l_1, \ldots, l_s$

Outline of the proof: final touch

Last step: Van den Nest



Local Cliffords layer $\approx$ finite sequence of l. comp. $l_1, \ldots, l_s$

Thus $G = G_m \star l_1 \star \cdots \star l_s$ is preparable on H. $\square$

Our results II

Theorem (Universality)

Any graph state can be prepared on any hardware using only CZ gates and local complementations.

Our results II

Theorem (Universality)

Any graph state can be prepared on any hardware using only CZ gates and local complementations.

Remark

This method is implemented by a $\mathcal{O}(n^3)$ time constructive algorithm, that yields an output circuit with a CZ count of ***at most*** *the CZ count of the input circuit.*

Our results II

Theorem (Universality)

Any graph state can be prepared on any hardware using only CZ gates and local complementations.

Remark

This method is implemented by a $\mathcal{O}(n^3)$ time constructive algorithm, that yields an output circuit with a CZ count of ***at most*** *the CZ count of the input circuit.*

Theorem (Optimality)

The set of circuits on arbitrary constructed by our method always contains circuits with optimal CZ count.

Our results III

Pure graph theory restatement!

Corollary (Graph statement)

Let H be a connected graph on n vertices.
Then any graph on n vertices can be created from the empty graph on n vertices with only local complementations and edge toggling in H.

Our results III

Pure graph theory restatement!

Corollary (Graph statement)

Let H be a connected graph on n vertices.
Then any graph on n vertices can be created from the empty graph on n vertices with only local complementations and edge toggling in H.

Proof by quantum circuit.
Can we find a purely combinatorial proof?

Thank you for your attention!

Contact: tristan.cam@ibm.com