Reducing the CNOT Count for Clifford+T Circuits on NISQ Architectures

Vlad Gheorghiu[2], Sarah Meng Li[1], Michele Mosca[2], and Priyanka Mukhopadhyay[2].

[1]Department of Mathematics and Statistics, Dalhousie University, Halifax NS, Canada
[2]Institute for Quantum Computing, University of Waterloo, Waterloo ON, Canada

April 6th, 2021

Background

Compilation: A set of instructions are realized by some universal gate set.

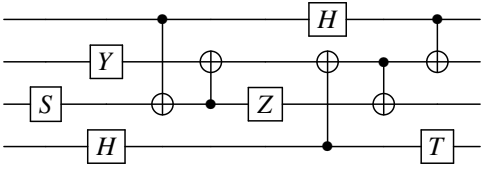
Implementation: Unitary operations are mapped to physical architectures.

Connectivity constraint: We cannot arbitrarily apply a multi-qubit gate on any set of qubits.

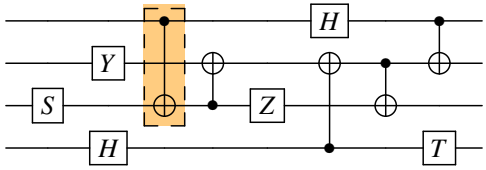
Clifford+T Circuits

Clifford+T circuits are quantum circuits over the gate set

$$\{CNOT, H, T, S, X, Y, Z\}.$$



Basic Gates



- CNOT acts on two qubits, *control* c and *target* t.

$$CNOT\,|c,t\rangle = |c, c \oplus t\rangle\,.$$

- X, Y, Z, T, S act on a single qubit.

$$X\,|t\rangle = |t \oplus 1\rangle\,,\ Y\,|t\rangle = \omega^{4t}\,|t \oplus 1\rangle\,,\ Z\,|t\rangle = \omega^{4t}\,|t\rangle\,,\ S\,|t\rangle = \omega^{2t}\,|t\rangle\,,\ T\,|t\rangle = \omega^{t}\,|t\rangle\,.$$

$c, t \in \mathbb{F}_2, \omega = e^{\frac{i\pi}{4}}, \oplus$ corresponds to Boolean exclusive-OR.

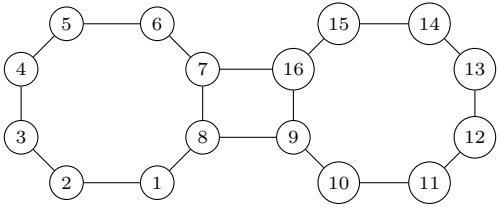
Connectivity Graph

Definition

A *graph* is a pair $G = (V_G, E_G)$ where V_G is a set of *vertices* and E_G is a set of pairs $e = (u, v)$ such that $u, v \in V_G$. Each such pair is called an *edge*.

Remark: We are interested in the *simple undirected connected graphs*.

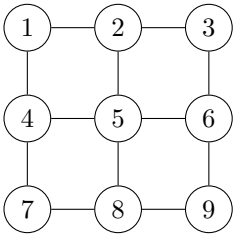
- Simple: there is at most one edge between two distinct vertices and no self-loops, i.e., $(u, u) \notin E_G$.
- Undirected: edges have no direction i.e., $(u, v) \equiv (v, u)$.



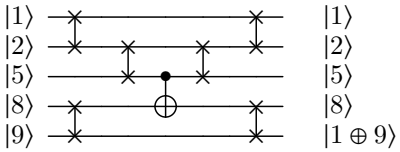
Rigetti 16Q-Aspen

Naive Solution

Naively we can insert SWAP operators to move a pair of logical qubits to physical positions admissible for two-qubit operations.



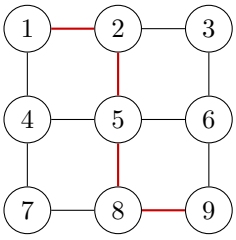
9-qubit square grid



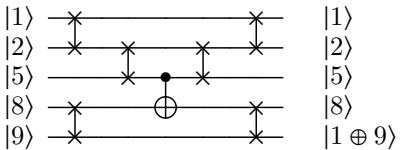
$CNOT_{1,9}$ with SWAPs

Naive Solution

Naively we can insert SWAP operators to move a pair of logical qubits to physical positions admissible for two-qubit operations.

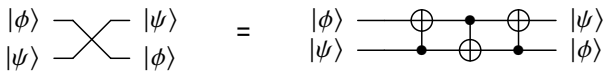


9-qubit square grid



$CNOT_{1,9}$ with SWAPs

Motivation



- If the shortest path length between vertices corresponding to c and t in G is ℓ, the naive solution requires about $6(\ell - 1)$ CNOT gates.
- This entails a significant increase in CNOT-count.
- **Can we reduce the CNOT-count while respecting the connectivity constraint?**

Related Work

We were inspired to use the following techniques.

- Steiner tree problem reduction[1][2].
- Parity network synthesis algorithm[3].
- Linear reversible circuits synthesis[4].

---

[1]Beatrice Nash, Vlad Gheorghiu, and Michele Mosca. “Quantum circuit optimizations for NISQ architectures”. In: *Quantum Science and Technology* 5.2 (2020), p. 025010.

[2]Aleks Kissinger and Arianne Meijer-van de Griend. “CNOT circuit extraction for topologically-constrained quantum memories”. In: *arXiv preprint arXiv:1904.00633* (2019).

[3]Matthew Amy, Parsiad Azimzadeh, and Michele Mosca. “On the controlled-NOT complexity of controlled-NOT–phase circuits”. In: *Quantum Science and Technology* 4.1 (2018), p. 015002.

[4]Ketan N Patel, Igor L Markov, and John P Hayes. “Optimal synthesis of linear reversible circuits”. In: *Quantum Information & Computation* 8.3 (2008), pp. 282–294.

Steiner Tree

Definition

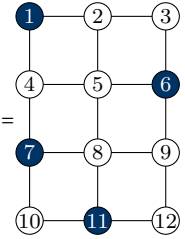
Given a graph $G = (V_G, E_G)$ with a weight function w_E and a set of vertices $\mathcal{S} \subseteq V_G$, a Steiner tree $T = (V_T, E_T)$ is a minimum weight tree that is a subgraph of G such that $\mathcal{S} \subseteq V_T$.

Terminals: Vertices in $\mathcal{S}$;

Steiner nodes: Vertices in $V_T \setminus \mathcal{S}$.

Example: $\mathcal{S} = \{1, 6, 7, 11\}, V_T \setminus \mathcal{S} = \{2, 3, 4, 5, 8, 9, 10, 12\}$

$G =$



G is a simple, undirected, and unweighted graph

Steiner Tree

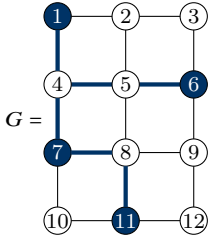
Definition

Given a graph $G = (V_G, E_G)$ with a weight function w_E and a set of vertices $\mathcal{S} \subseteq V_G$, a Steiner tree $T = (V_T, E_T)$ is a minimum weight tree that is a subgraph of G such that $\mathcal{S} \subseteq V_T$.

Terminals: Vertices in $\mathcal{S}$;

Steiner nodes: Vertices in $V_T \setminus \mathcal{S}$.

Example: $\mathcal{S} = \{1, 6, 7, 11\}, V_T \setminus \mathcal{S} = \{2, 3, 4, 5, 8, 9, 10, 12\}$



A solution to the Steiner tree problem on G.

Slice-and-Build Technique[5]

Slice: Slice the circuit at the position of H gate, by either

(A) partitioning the gates of the circuit based on the locality of H gates, or

(B) partitioning the phase polynomial of the input circuit.

Build: Re-synthesize the intermediate sliced portions so that connectivity is respected and the CNOT count is reduced.

[5] Vlad Gheorghiu et al. "Reducing the CNOT count for Clifford+ T circuits on NISQ architectures". In: *arXiv preprint arXiv:2011.12191* (2020).

Slice-and-Build Technique

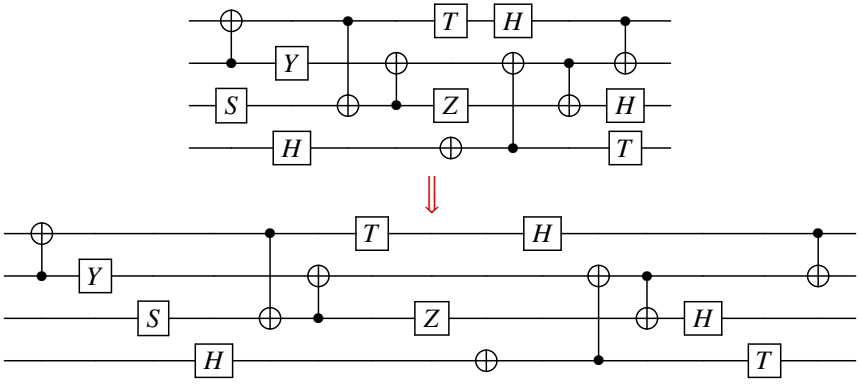
Slice: Slice the circuit at the position of H gate, by either

(A) partitioning the gates of the circuit based on the locality of H gates, or

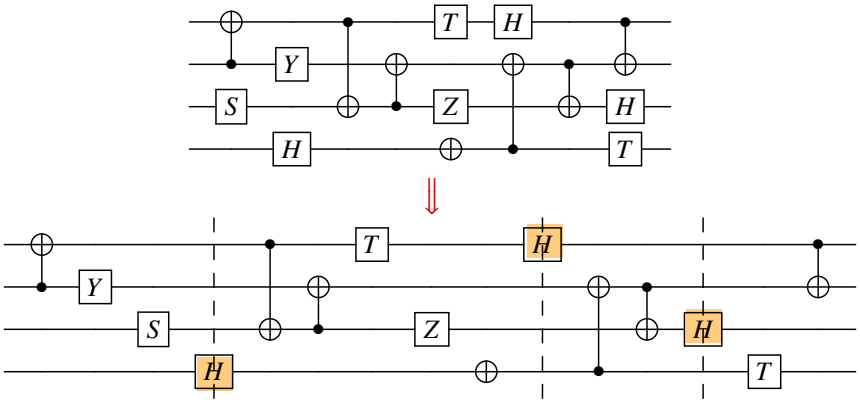
(B) partitioning the phase polynomial of the input circuit.

Build: Re-synthesize the intermediate sliced portions so that connectivity is respected and the CNOT count is reduced.

Slice



Slice



Build

LINEAR-TF-SYNTH algorithm synthesize circuits over $\{CNOT, X\}$.

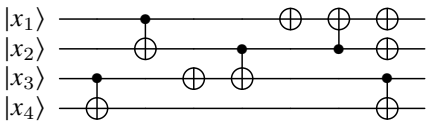
PHASE-NW-SYNTH algorithm synthesize circuits over $\{CNOT, X, T\}$.

Synthesize Circuits over $\{CNOT, X\}$

Overall Linear Transformation

Consider an n–qubit circuit over $\{CNOT, X\}$, we represent the overall linear transformation using an $n \times (n+1)$ binary matrix.

Example



$|x_2\rangle$
$|x_1 \oplus x_2 \oplus 1\rangle$
$|x_1 \oplus x_2 \oplus x_3 \oplus 1\rangle$
$|x_1 \oplus x_2 \oplus x_4 \oplus 1\rangle$

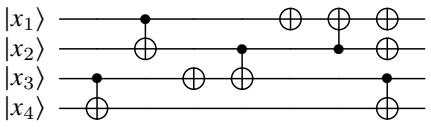
$$A = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & b \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Synthesize Circuits over $\{CNOT, X\}$

Overall Linear Transformation

Consider an n–qubit circuit over $\{CNOT, X\}$, we represent the overall linear transformation using an $n \times (n+1)$ binary matrix.

Example



$|x_2\rangle$
$|x_1 \oplus x_2 \oplus 1\rangle$
$|x_1 \oplus x_2 \oplus x_3 \oplus 1\rangle$
$|x_1 \oplus x_2 \oplus x_4 \oplus 1\rangle$

$$A = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & b \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

LINEAR-TF-SYNTH Algorithm

Reverse Engineering

(a) Make $\mathbf{b} = \mathbf{0}$ by applying X to corresponding qubits.

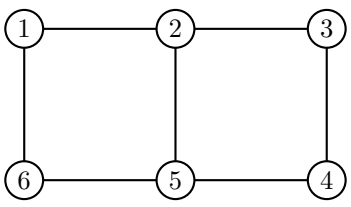
(b) Carry out an analogue of Gaussian elimination.

(c) Use Steiner tree to incorporate connectivity constraints.

Example: Let A be a linear transformation and G be the connectivity graph.

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$G =$



Step 1: Reducing to Upper Triangular Form

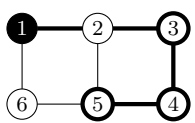
Row Operations

(a) Starting from the left most column, fix one column at a time.

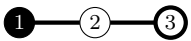
(b) Fixing the ith column means applying row operations such that $A_{ii} = 1$ and $A_{ji} = 0$ for every $j > i$.

$$A = \begin{bmatrix} \mathbf{1} & 1 & 0 & 1 & 1 & 0 \\ \mathbf{0} & 0 & 1 & 1 & 0 & 1 \\ \mathbf{1} & 0 & 1 & 0 & 1 & 0 \\ \mathbf{1} & 1 & 0 & 1 & 0 & 0 \\ \mathbf{1} & 1 & 1 & 1 & 0 & 0 \\ \mathbf{0} & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$T_{1,\{1,3,4,5\}} =$

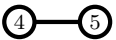


$T_1 =$



$T_2 =$

$T_3 =$



The Steiner tree $T_{1,\mathcal{S}}$ with pivot at 1 and terminals $\mathcal{S} = \{1, 3, 4, 5\}$. T_1, T_2 and T_3 are the sub-trees built from it.

Step 1: Reducing to Upper Triangular Form

Remark: By traversing subtrees, $CNOT$s are concatenated.

$$\begin{aligned}\mathcal{Y}_1^1 &= \text{CNOT}_{45}, \text{CNOT}_{34}, \text{CNOT}_{12}, \text{CNOT}_{23}, \text{CNOT}_{12} \\ \mathcal{A}_1^1 &= A[5,.] \leftarrow A[5,.] \oplus A[4,.], A[4,.] \leftarrow A[4,.] \oplus A[3,.], A[2,.] \leftarrow A[2,.] \oplus A[1,.], \\ &\quad A[3,.] \leftarrow A[3,.] \oplus A[2,.], A[2,.] \leftarrow A[2,.] \oplus A[1,.]\end{aligned}$$

After a series of row operations, the matrix A is reduced to an upper triangular form.

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \longrightarrow \ldots \longrightarrow A = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 1: Reducing to Upper Triangular Form

Remark: By traversing subtrees, $CNOT$s are concatenated.

$$\begin{aligned}\mathcal{Y}_1^1 &= \text{CNOT}_{45}, \text{CNOT}_{34}, \text{CNOT}_{12}, \text{CNOT}_{23}, \text{CNOT}_{12} \\ \mathcal{A}_1^1 &= A[5,.] \leftarrow A[5,.] \oplus A[4,.], A[4,.] \leftarrow A[4,.] \oplus A[3,.], A[2,.] \leftarrow A[2,.] \oplus A[1,.], \\ &\quad A[3,.] \leftarrow A[3,.] \oplus A[2,.], A[2,.] \leftarrow A[2,.] \oplus A[1,.]\end{aligned}$$

After a series of row operations, the matrix A is reduced to an upper triangular form.

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \longrightarrow \ldots \longrightarrow A = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 2: Transposing A and Reducing to Identity

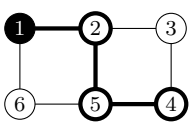
Row Operations

(a) Starting from the left most column, fix one column at a time.

(b) Fixing the ith column means applying row operations such that $A_{ii} = 1$ and $A_{ji} = 0$ for every $j > i$.

$$A = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 1 & 0 & 0 & 0 & 0 \\ \mathbf{0} & 1 & 1 & 0 & 0 & 0 \\ \mathbf{1} & 1 & 1 & 1 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 1 & 0 \\ \mathbf{0} & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$T_{1,\{1,2,4,5\}}$ =



T_1 =

T_2 =

T_3 =

The Steiner tree $T_{1,\mathcal{S}}$ with pivot at 1 and terminals $\mathcal{S} = \{1, 2, 4, 5\}$. T_1, T_2 and T_3 are the sub-trees built from it.

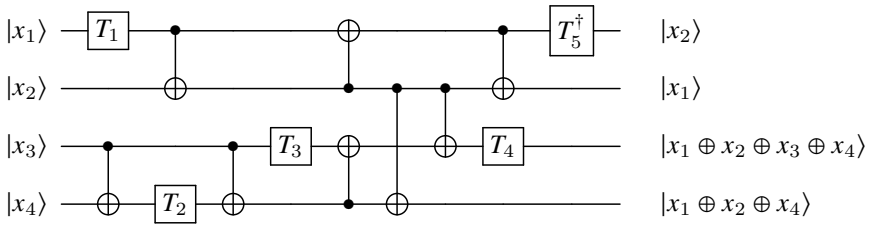
Synthesize Circuits over $\{CNOT, X, T\}$

- Consider circuits over the gate set

$$\{CNOT, X, T, S := T^2, Z := T^4, T^\dagger := T^7, S^\dagger := T^6\}.$$

- $CNOT\,|x, y\rangle = |x, x \oplus y\rangle$, $T\,|x\rangle = \omega\,|x\rangle$, where $\omega = e^{\frac{i\pi}{4}}$ and $x, y \in \mathbb{F}_2$.

Example:



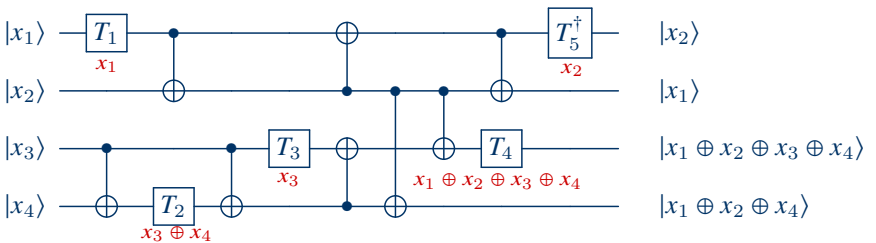
Synthesize Circuits over $\{CNOT, X, T\}$

- Consider circuits over the gate set

$$\{CNOT, X, T, S := T^2, Z := T^4, T^\dagger := T^7, S^\dagger := T^6\}.$$

- $CNOT\,|x, y\rangle = |x, x \oplus y\rangle$, $T\,|x\rangle = \omega\,|x\rangle$, where $\omega = e^{\frac{i\pi}{4}}$ and $x, y \in \mathbb{F}_2$.

Example:



$T_1 : \omega^{x_1}$

$T_2 : \omega^{x_3 \oplus x_4}$

$T_3 : \omega^{x_3}$

$T_4 : \omega^{x_1 \oplus x_2 \oplus x_3 \oplus x_4}$

$T_5 : \omega^{7x_2}$

Circuit-polynomial Correspondence[6]

Lemma

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

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

where $\omega = e^{\frac{i\pi}{4}}$, $x_1, x_2, \ldots, x_n \in \mathbb{F}_2$ and

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

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

[6] Matthew Amy, Dmitri Maslov, and Michele Mosca. "Polynomial-time T-depth optimization of Clifford+ T circuits via matroid partitioning". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.10 (2014), pp. 1476–1489.

Definition

$\omega^{p(x_1,x_2,\ldots,x_n)}|g(x_1,x_2,\ldots,x_n)\rangle$**:** The sum-over-paths form of a circuit.

$x_1,x_2,\ldots,x_n$**:** Path variables.

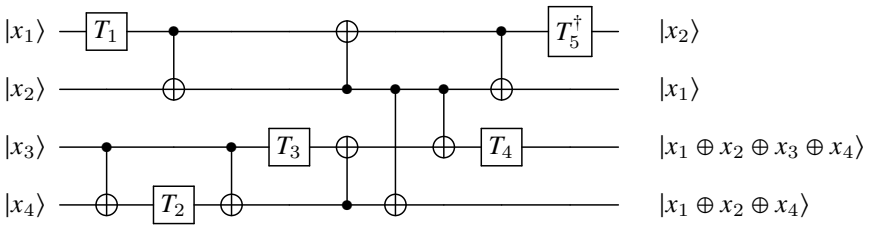
$p(x_1,x_2,\ldots,x_n)$**:** A phase polynomial.

$f_i(x_1,x_2,\ldots,x_n)$**:** A parity term.

$\mathcal{P}$**:** A phase polynomial set consists of linear Boolean functions together with coefficients in $\mathbb{Z}_8$.

Intuition: A unitary implemented over $\{\text{CNOT},\text{T}\}$ can be characterized by a set $\mathcal{P}=\{(c,f):c\in\mathbb{Z}_8 \text{ and } f\in\left(\mathbb{F}_2^n\right)^*\}$ and linear reversible output functions $g:\mathbb{F}_2^n\to\mathbb{F}_2^n$.

Example Continued



$T_1 : \omega^{x_1}$

$T_2 : \omega^{x_3 \oplus x_4}$

$T_3 : \omega^{x_3}$

$T_4 : \omega^{x_1 \oplus x_2 \oplus x_3 \oplus x_4}$

$T_5 : \omega^{7x_2}$

- $g(x_1, x_2, \ldots, x_n) = (x_2)(x_1)(x_1 \oplus x_2 \oplus x_3 \oplus x_4)(x_1 \oplus x_2 \oplus x_4)$.

  LINEAR-TF-SYNTH algorithm returns a circuit over {CNOT, X} that realizes g.

- $\mathcal{P} = \{(1, x_1), (1, x_3 \oplus x_4), (1, x_3), (1, x_1 \oplus x_2 \oplus x_3 \oplus x_4), (7, x_2)\}$.

- $\forall (c_1, f_1), (c_2, f_2) \in \mathcal{P}$, if $f_1 = f_2$, they can be merged into a single pair $(c_1 +_8 c_2, f_1)$.

PHASE-NW-SYNTH Algorithm

Let $\mathcal{P}$ be a phase polynomial set and **A** be the matrix corresponding to the linear reversible output function g.

Synthesizing a Phase Polynomial Network

(a) Synthesize a circuit over $\{CNOT, X\}$ that realizes the parity terms in $\mathcal{P}$.

(b) Apply $\{T, T^\dagger, S, S^\dagger, Z, Y\}$ depending on the coefficients c in $\mathcal{P}$.

(c) synthesize a circuit so that the overall linear transformation is **A**.

Example: Consider a 6–qubit quantum system, let

$$\mathcal{P} = \{(1, 1 \oplus x_1 \oplus x_4 \oplus x_5), (2, x_2 \oplus x_3 \oplus x_5 \oplus x_6), (4, 1 \oplus x_4 \oplus x_5 \oplus x_6), (4, 1 \oplus x_1 \oplus x_2 \oplus x_6), (6, 1 \oplus x_1 \oplus x_2 \oplus x_3), (7, 1 \oplus x_1 \oplus x_2 \oplus x_4 \oplus x_6), (1, x_2 \oplus x_4 \oplus x_5)\}$$

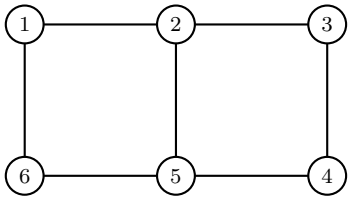
Columns Represent Parity Term

$$\mathcal{P} = \{(1, 1 \oplus x_1 \oplus x_4 \oplus x_5), (2, x_2 \oplus x_3 \oplus x_5 \oplus x_6), (4, 1 \oplus x_4 \oplus x_5 \oplus x_6), (4, 1 \oplus x_1 \oplus x_2 \oplus x_6), (6, 1 \oplus x_1 \oplus x_2 \oplus x_3), (7, 1 \oplus x_1 \oplus x_2 \oplus x_4 \oplus x_6), (1, x_2 \oplus x_4 \oplus x_5)\}$$

$P =$

p_1	p_2	p_3	p_4	p_5	p_6	p_7
1	0	0	1	1	1	0
0	1	0	1	1	1	1
0	1	0	0	1	0	0
1	0	1	0	0	1	1
1	1	1	0	0	0	1
0	1	1	1	0	1	0
1	0	1	1	1	1	0
1	2	4	4	6	7	1

$G =$



The parity matrix $P_{8\times7}$ and connectivity graph G.

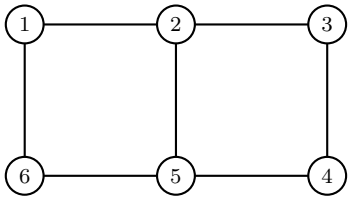
Top Six Rows Encode Parity

$$\mathcal{P} = \{(1, 1 \oplus x_1 \oplus x_4 \oplus x_5), (2, x_2 \oplus x_3 \oplus x_5 \oplus x_6), (4, 1 \oplus x_4 \oplus x_5 \oplus x_6), (4, 1 \oplus x_1 \oplus x_2 \oplus x_6), (6, 1 \oplus x_1 \oplus x_2 \oplus x_3), (7, 1 \oplus x_1 \oplus x_2 \oplus x_4 \oplus x_6), (1, x_2 \oplus x_4 \oplus x_5)\}$$

$P =$

p_1	p_2	p_3	p_4	p_5	p_6	p_7
1	0	0	1	1	1	0
0	1	0	1	1	1	1
0	1	0	0	1	0	0
1	0	1	0	0	1	1
1	1	1	0	0	0	1
0	1	1	1	0	1	0
1	0	1	1	1	1	0
1	2	4	4	6	7	1

$G =$



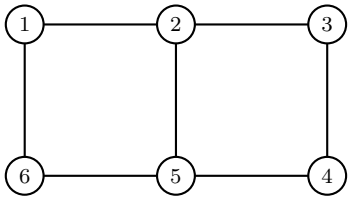
The parity matrix $P_{8\times7}$ and connectivity graph G.

The 7th Row Encodes Bit Flip

$$\mathcal{P} = \{(1, 1 \oplus x_1 \oplus x_4 \oplus x_5), (2, x_2 \oplus x_3 \oplus x_5 \oplus x_6), (4, 1 \oplus x_4 \oplus x_5 \oplus x_6), (4, 1 \oplus x_1 \oplus x_2 \oplus x_6), (6, 1 \oplus x_1 \oplus x_2 \oplus x_3), (7, 1 \oplus x_1 \oplus x_2 \oplus x_4 \oplus x_6), (1, x_2 \oplus x_4 \oplus x_5)\}$$

$$P = \begin{bmatrix} \underline{p_1} & \underline{p_2} & \underline{p_3} & \underline{p_4} & \underline{p_5} & \underline{p_6} & \underline{p_7} \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 4 & 4 & 6 & 7 & 1 \end{bmatrix}$$

$G =$



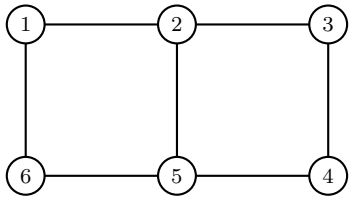
The parity matrix $P_{8\times7}$ and connectivity graph G.

The Last Row Stores Coefficients

$$\mathcal{P} = \{(1, 1 \oplus x_1 \oplus x_4 \oplus x_5), (2, x_2 \oplus x_3 \oplus x_5 \oplus x_6), (4, 1 \oplus x_4 \oplus x_5 \oplus x_6), (4, 1 \oplus x_1 \oplus x_2 \oplus x_6), (6, 1 \oplus x_1 \oplus x_2 \oplus x_3), (7, 1 \oplus x_1 \oplus x_2 \oplus x_4 \oplus x_6), (1, x_2 \oplus x_4 \oplus x_5)\}$$

$$P = \begin{bmatrix} \underline{p_1} & \underline{p_2} & \underline{p_3} & \underline{p_4} & \underline{p_5} & \underline{p_6} & \underline{p_7} \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 4 & 4 & 6 & 7 & 1 \end{bmatrix}$$

$G =$



The parity matrix $P_{8\times7}$ and connectivity graph G.

PHASE-NW-SYNTH Algorithm Snapshot

- Ignore the last two rows of P, let $B = \{p'_1, p'_2, p'_3, p'_4, p'_5, p'_6, p'_7\}$, $\mathcal{K}$ be an empty stack, and $I = [6]$.
- Cycle through the set of n-bit strings and apply corresponding $CNOT$ gates at each iteration.
- Whenever a column has a single 1, it implies that the corresponding parity has been realized.

Example: After the 4th iteration, we have

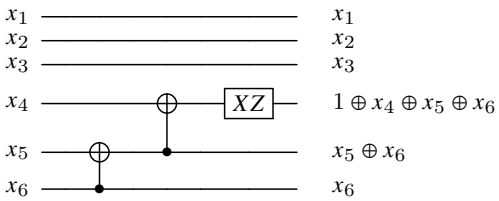
$$B^{(4)} = \begin{bmatrix} \underline{p_1} & \underline{p_2} & \underline{p_3} & \underline{p_4} & \underline{p_5} & \underline{p_6} & \underline{p_7} \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

PHASE-NW-SYNTH Algorithm Snapshot

- Whenever a column has a single 1, it implies that the corresponding parity has been realized.
- Remove these columns from the remaining parities.
- Place the gate X if parity realized on circuit is $1 \oplus f$ for some $(c, f) \in \mathcal{P}$. We can also place a gate in $\{\mathsf{T}, \mathsf{T}^\dagger, \mathsf{S}, \mathsf{S}^\dagger, \mathbb{Z}, \mathsf{Y}\}$ corresponding to the value of the coefficient c.

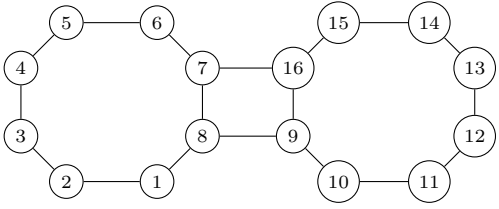
Example: The partial circuit obtained after applying a sequence of gates from iteration 4.

$$B^{(4)} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

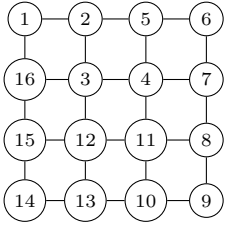


Implementation

We simulated benchmarks as well as random circuits on popular architectures such as 9-qubit square grid, 16-qubit square grid, Rigetti 16-qubit Aspen, 16-qubit IBM QX5 and 20-qubit IBM Tokyo.



Rigetti 16Q-Aspen



16q-Square Grid

Results

Architecture	#Qubits	Initial count	SWAP-template Count	CNOT-OPT-A Count	CNOT-OPT-A Time
9q-square	9	3	560%	0.00%	0.184s
		5	612%	146%	0.146s
		10	594%	105%	0.167s
		20	546%	176%	0.2s
		30	596%	184.67%	0.233s
16q-square	16	4	1050%	238%	0.23s
		8	840%	146.25%	0.27s
		16	817.50%	158.13%	0.43s
		32	853%	340.63%	0.41s
		64	892.50%	220.78%	0.49s
		128	858.75%	210.63%	0.57s
		256	897.42%	237.5%	0.72s
rigetti-16q-aspen	16	4	1680%	355%	0.23s
		8	1740%	253%	0.396s
		16	1619.90%	351%	0.47s
		32	1794%	469.48%	0.48s
		64	1755%	399%	0.66s
		128	1760.63%	368.13%	0.58s
		256	1757.11%	410.9%	0.61s

Architecture	#Qubits	Initial count	SWAP-template Count	CNOT-OPT-A	
				Count	Time
ibm-qx5	16	4	1260%	173%	0.38s
		8	1035%	295%	0.36s
		16	1042.50%	283%	0.41s
		32	1179.38%	398.44%	0.42s
		64	1130.63%	339.06%	0.45s
		128	1110.94%	344.69%	0.575s
		256	1141.17%	379.88%	0.73s
ibm-q20-tokyo	20	4	525%	128%	0.186s
		8	555%	275%	0.295s
		16	570%	88%	0.37s
		32	500.63%	154.38%	0.55s
		64	542.81%	136.88%	0.54s
		128	539.53%	141.02%	0.645s
		256	534.61%	125.27%	0.72s

Table: The overhead or increase in CNOT-count has been compared to the overhead obtained by using SWAP-template.

Conclusion

- We provided a heuristic algorithm that work with the univeral Clifford+T gate set.
- For both benchmark and random circuits, our algorithm results in much less overhead in terms of the increase in CNOT-count, compared to the overhead obtained by using SWAP template.
- The results will likely be improved if coupled with procedures that optimize the initial mapping of qubits.


Thank you!