CNOT-dihedral circuits

Another class of circuits which can be **efficiently represented** and **optimally** (in some cases) synthesized are **CNOT-dihedral circuits**

Def'n (CNOT-dihedral)

CNOT-dihedral circuits of order n are those over $\{X_n, CNOT_n, R_n = R_z(\pi/n)\}$

Fact

$\{X_n, R_n\}$ generate the **dihedral group** of order n, D_n

Fact

CNOT and X together generate the **affine** permutations

$$(A, \vec{b}) \in GA(n, \mathbb{Z}_2) \simeq GL(n, \mathbb{Z}_2) \ltimes \mathbb{Z}_2^n$$
$$(A, \vec{b})\vec{x} = A\vec{x} \oplus \vec{b}$$

E.g. $X|x\rangle = |x \oplus 1\rangle = |(I, 1)x\rangle$

Intuitively then, CNOT-dihedral circuits compute an affine on basis states, together with a **phase** which is a sum of phases on affine combinations of basis states

e.g.  sends $|x\rangle|y\rangle \xrightarrow{CNOT} |x\rangle|x \oplus y\rangle$
$\xrightarrow{T} \omega^{x \oplus y}|x\rangle|x \oplus y\rangle$

 sends $|x\rangle \xrightarrow{X} |x \oplus 1\rangle$
$\xrightarrow{T} \omega^{x \oplus 1}|x \oplus 1\rangle$

Def^n

Given $\vec{x}, \vec{y} \in \mathbb{Z}_2^n$, $\chi_{\vec{y}}(\vec{x}) = \vec{x} \cdot \vec{y} = x_1 y_1 \oplus \cdots \oplus x_n y_n$

Representation of CNOT-dihedral groups

Prop (phase polynomial representation)

Let C be a circuit over $\{X, CNOT, R_m\}$. Then

$$[\![C]\!] : |\vec{x}\rangle \longmapsto e^{2\pi i/m\, P(\vec{x})} |A\vec{x} + \vec{b}\rangle$$

where $A \in GL(n, \mathbb{Z}_2)$, and $P : \mathbb{Z}_2^n \to \mathbb{Z}_m$ can be written as

$$P(\vec{x}) = \sum_{\vec{y}} a_{\vec{y}}\, x_{\vec{y}}(\vec{x}), \quad a_{\vec{y}} \in \mathbb{Z}_m$$

We call P a **phase polynomial** and $(P, A, \vec{b})$ a **phase polynomial representation** of C

Pf.

By induction on the gates of g. Let

$$[\![C]\!] : |\vec{x}\rangle \longmapsto e^{2\pi i/m\, P(\vec{x})} |A\vec{x} + \vec{b}\rangle$$

Then $[\![X_i \cdot C]\!]|\vec{x}\rangle \longmapsto e^{2\pi i/m\, P(\vec{x})}(X_i |A\vec{x} + \vec{b}\rangle)$

$$\longmapsto e^{2\pi i/m\, P(\vec{x})} |A\vec{x} + (\vec{b} + \vec{e}_i)\rangle$$

$$[\![CX_{ij} \cdot C]\!]|\vec{x}\rangle \longmapsto e^{2\pi i/m\, P(\vec{x})}(CX_{ij} |A\vec{x} + \vec{b}\rangle)$$

$$\longmapsto e^{2\pi i/m\, P(\vec{x})} |E_{ij} A\vec{x} + E_{ij}\vec{b}\rangle$$

$$[\![(R_k)_i \cdot C]\!]|\vec{x}\rangle \longmapsto e^{2\pi i/m\, P(\vec{x})}((R_k)_i |A\vec{x} + \vec{b}\rangle)$$

$$\longmapsto e^{2\pi i/m\, P(\vec{x})}\, e^{2\pi i/m (A_i \vec{x} \oplus b_i)} |A\vec{x} + \vec{b}\rangle$$

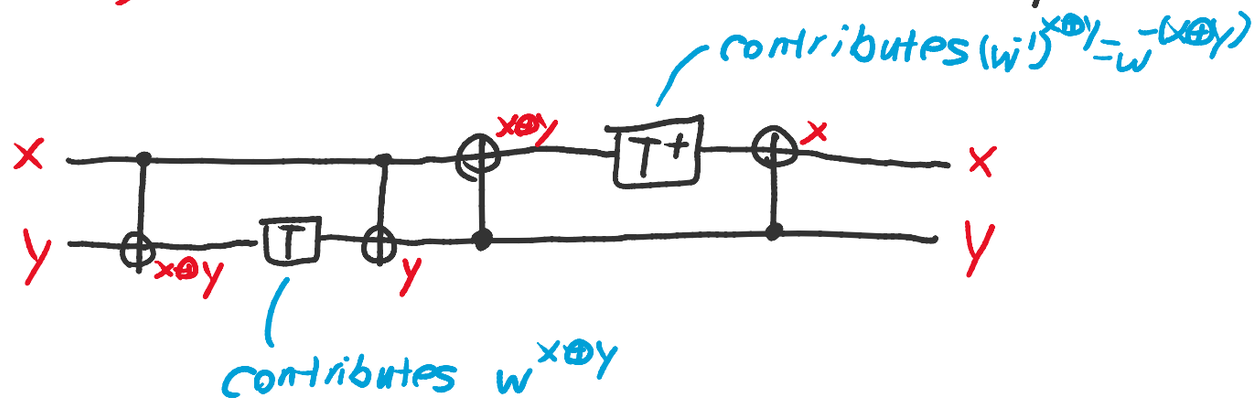
$$\longmapsto e^{2\pi i/m (P(\vec{x}) + (-1)^{b_i} x_{A_i}(\vec{x}) + b_i)} |A\vec{x} + \vec{b}\rangle$$

Note that $(-1)^{b_i} x_{A_i}(\vec{x}) + b_i$ is itself a prose polynomial

Annotated circuits

We can find a Phase polynomial representation by annotating the states of qubits (in the comp. basis)

E.g.



Hence,

$$|x\rangle|y\rangle \longmapsto w^{x\oplus y} w^{-(x\oplus y)} |x\rangle|y\rangle = |x\rangle|y\rangle$$

Prop.

Given a circuit C over $\{X, CNOT, R_m\}$, a phase polynomial representation of C can be computed in time and space linear in the volume of C,

$$|C| = (\text{num qubits of } C) \cdot (\text{num gates of } C)$$

We call the bitstrings $\vec{y} \in \mathbb{Z}_q^n$ where $q_{\vec{y}} \neq 0$ the Support of a phase polynomial P, denoted $\text{supp}(P)$. While phase polynomials are NOT unique, e.g.

$$\sum_{\vec{y}} x_{\vec{y}}(\vec{x}) \equiv 0 \bmod 8 \quad \forall \vec{x}$$

The canonical phase polynomial of C, obtained by annotating the circuit and collecting the phases satisfies

$$|\text{supp}(P)| \leq \#\, R_n \text{ gates in } C$$

Synthesis of CNOT-dihedral ops

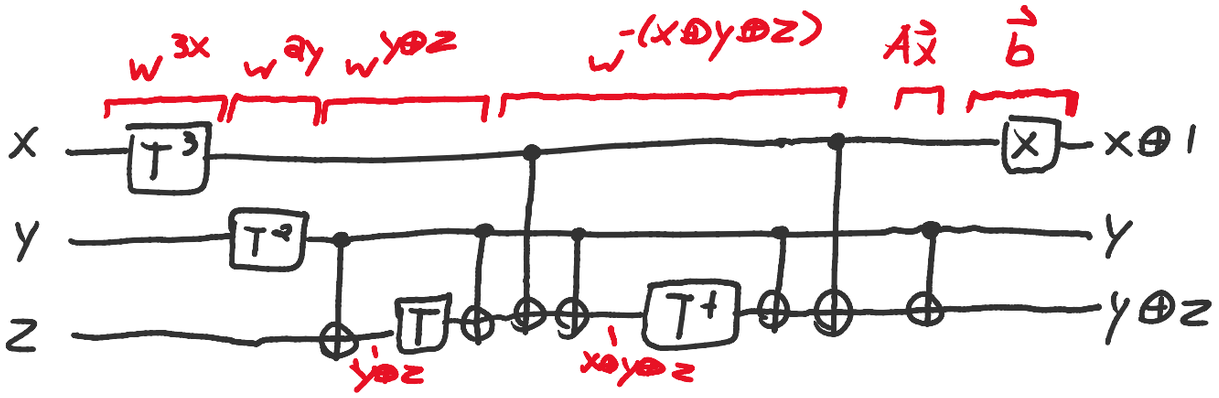
Intuitively, a phase polynomial representation $(P, A, \vec{b})$ corresponds to a sequence of phase rotations on parities of $\vec{x}$, followed by a final Affine permutation, so we can easily synthesize such a circuit.

E.g.

Let $U:|xyz\rangle \longmapsto \omega^{3x+2y+(y\oplus z)-(x\oplus y\oplus z)}|(x\oplus 1)y(y\oplus z)\rangle$

($e^{\pi i/4}$)

We can implement U over $\{CNOT, X, T = R_8\}$ as so:



Prop

Let $P(\vec{x}) = \sum_{\vec{y}} a_{\vec{y}} x_{\vec{y}}(\vec{x})$, $a_{\vec{y}} \in \mathbb{Z}_m$ and $(A, \vec{b}) \in GA(n, \mathbb{Z}_2)$.

Then $U:|\vec{y}\rangle \longmapsto e^{2\pi i/m\, P(\vec{y})}|A\vec{x}+\vec{b}\rangle$ can be synthesized over $\{CNOT, X, R_m\}$ in time polynomial in n and with $|\mathrm{supp}(P)|$ R_n^k gates, $k \in \mathbb{Z}_m$

Note:

If $m = 2^8$, i.e. $R_m = T$, then any even power of T is a Clifford gate, and any odd power needs only 1 T gate to implement, since $T^{2k+1} = T \cdot T^{2k}$

Hence, T-count $= |\mathrm{supp}(P \bmod 2)|$

Synthesizing the Toffoli gate

Recall that  , where is the CCZ gate defined by

$$CCZ : |xyz\rangle \longmapsto (-1)^{xyz} |xyz\rangle$$

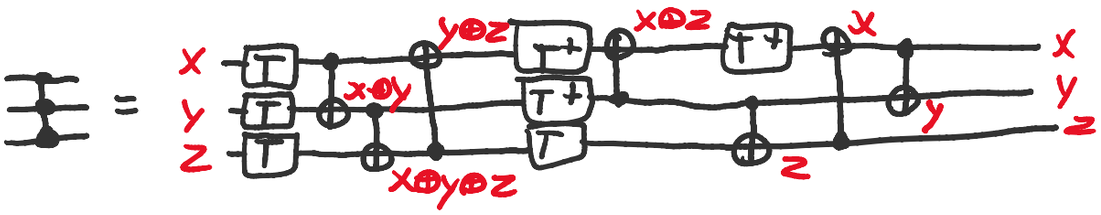
Taking the Fourier expansion* of $f(x,y,z) = xyz$ gives us

$$f(x,y,z) = \frac{1}{4}\left[x + y + z - (x \oplus y) - (x \oplus z) - (y \oplus z) + (x \oplus y \oplus z)\right]$$

↑ ↑ ↑ ↑ ↑ ↑ ↑

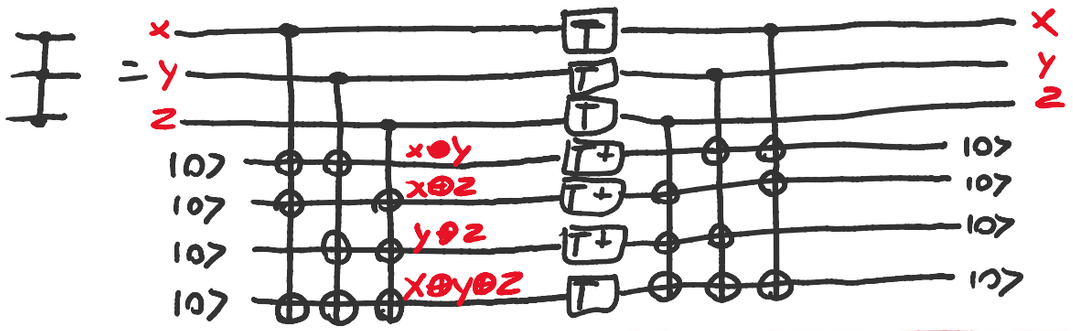
this is a phase polynomial over $\sqrt[4]{-1} = w = e^{\pi i/4}$

We can synthesize an explicit circuit for CCZ by preparing each parity and applying either a $w = e^{\pi i/4}$ or $w^{-1} = e^{\pi i/4}$ rotation (T or T^+).



(T-depth 1 CCZ)

Alternatively, we can synthesize the CCZ (and hence Toffoli gate) in T-depth 1 by using an ancilla to "hold" each parity at once



---

* Fourier expansion: given $f : \mathbb{Z}_2^n \to \mathbb{R}$, an expression of f as $f(\vec{x}) = \sum_{\vec{y}} \hat{f}(\vec{y}) \chi_{\vec{y}}(\vec{x})$

The phase polynomial method

We now have the ingredients for a **phase gate reducing algorithm**, by re-synthesizing CNOT-dihedral circuits. In particular, note

1. The phase polynomial representation $(P, A, \vec{b})$ of C can be computed efficiently
2. $|\mathrm{supp}(P)| \leq$ # phase gates in C
3. The phase polynomial representation $(P, A, \vec{b})$ can be synthesized efficiently with $|\mathrm{supp}(P)|$ phase gates

(phase gate, e.g. T gate, optimization for $\{CNOT, X, R_m\}$)

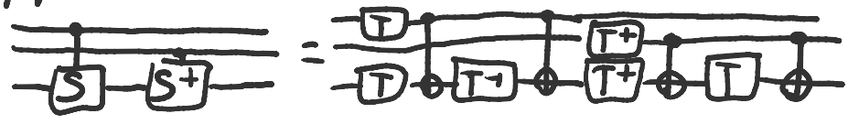
$$C \xrightarrow{\text{annotate}} (P, A, \vec{b}) \xrightarrow{\text{synthesize}} C'$$

$$R_m\text{-count}(C') \leq R_m\text{-count}(C)$$

profit ☺

(Example)

Suppose we want to optimize the circuit:

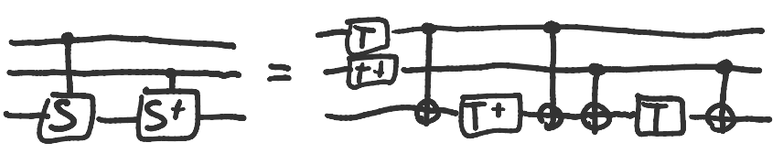


We start by computing the phase polynomial representation:

$$|xyz\rangle \longmapsto w^{x+2(z)-(x\oplus z)-y-2(z)+(y\oplus z)}|xyz\rangle$$

$$\longmapsto w^{x-(x\oplus z)-y+(y\oplus z)}|xyz\rangle$$

Re-synthesizing we get a circuit with $4 < 6$ T gates



Phase polynomial synthesis problems

More generally, we can ask what is the best circuit for $(P, A, \vec{b})$ in some cost metric? Synthesizing an optimal circuit for $(P, A, \vec{b})$ is known as a phase polynomial synthesis problem.

metric	result
R_m-depth	poly-time via Matroid partitioning (t-par)
R_m-count	equivalent to min-distance decoding for $RM^*(n, n-U_2(m)-1)$ (hard!)
CNOT-count	NP-hard in restricted cases asymptotically optimal heuristic (Gray-Synth)

Note: $U_2(m)$ is the 2-adic valuation of m (highest power of 2 that divides m)

(Upper & lower bounds on solutions)

metric	lower bound	upper bound
R_m-depth	$O(1)$ (with ancillas)	$O(1)$ (with ancillas)
R_m-count	Covering radius of $RM^*(n, n-U_2(m)-1)$	$O\left(\frac{n^{U_2(m)-1}}{U_2(m)!}\right)$ / $O(n^2)$ for T-count
CNOT-count	$O(n^2)$ (from gray code)	$O(n^2)$ (from gray code)

CNOT-minimal synthesis

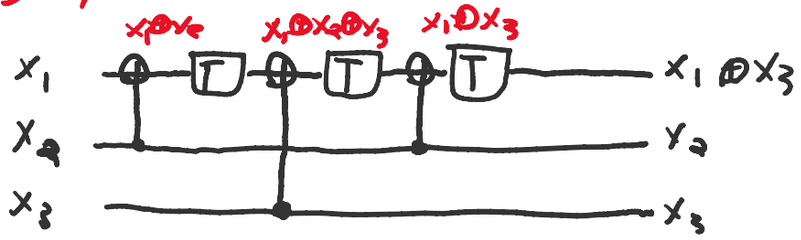
Problem:
Given a phase polynomial representation $(P, A, \vec{b})$, synthesize with minimal CNOT-count

Key idea:
we need to construct a tour of $\propto_{\vec{y}}(\vec{x})$ for each $\vec{y} \in \text{supp}\{P\}$ with the minimal CNOT-count

Ex.

Let $P(x_1, x_2, x_3) = x_1 \oplus x_2 + x_1 \oplus x_3 + x_1 \oplus x_2 \oplus x_3$

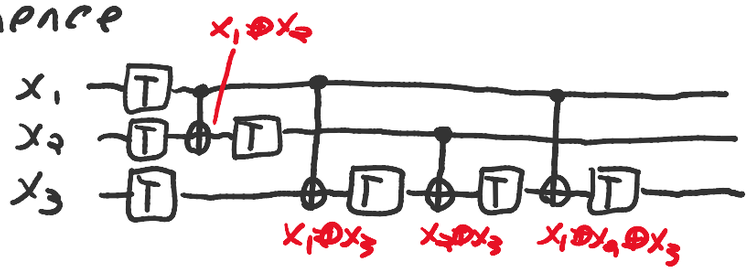
Since $\text{supp}\{P\} \simeq \mathbb{Z}_2^2$, we can construct a minimal tour using the gray code.



Ex.

Let $P(x_1, x_2, x_3) = x_1 + x_2 + x_3 + x_1 \oplus x_2 + x_1 \oplus x_3 + x_2 \oplus x_3 + x_1 \oplus x_2 \oplus x_3$

Now $\text{supp}\{P\} = \mathbb{Z}_2^3 \setminus \{0\}$. Any tour of $\text{supp}\{P\}$ hence requires at least $2^3 - 1$ CNOT gates. An optimal circuit is hence



Prop

Any set $X \simeq \mathbb{Z}_2^k$ (toured) can be enumerated optimally with $|X \setminus Y|$ CNOT gates where $Y \subseteq \mathbb{Z}_2^k$ is the set of inputs

Gray-synth

Basic idea

Attempt to decompose

$$S = \mathrm{supp}\{P\} = X_1 \cup X_2 \cup \cdots \cup X_m$$

where each $X_i \approx \mathbb{Z}_2^k$ for some k.

we do this by recursively selecting some qubit i which maximizes either the 0 or 1 co-factor,

$$i = \operatorname*{argmax}_i \max_{x \in \{0,1\}} |\{\vec{y} \in S \mid y_i = x\}|$$

Algorithm

queue

1. $C := []$, $Q := [(S, \{1, \ldots, n\}, e)]$
2. while $Q \neq []$
3. $\quad (S, I, j) = Q.\mathrm{pop}$
4. $\quad$ if $S = \emptyset$, then we're done
5. $\quad$ if $S = \{\vec{y}\}$, then
6. $\quad\quad$ for each j s.t. $\vec{y}_j = 1$
7. $\quad\quad\quad C := [CNOT_{j,i}] \mathbin{++} C$
8. $\quad\quad\quad$ for all remaining $\vec{y}'$, set $\vec{y}' := E_{i,j}\vec{y}'$
9. $\quad$ Else
10. $\quad\quad j := \operatorname{argmax}_{j \in I} \max_{x \in \{0,1\}} |\{\vec{y} \in S \mid y_j = x\}|$
11. $\quad\quad S_0 := \{\vec{y} \in S \mid y_j = 0\}$, $S_1 := \{\vec{y} \in S \mid y_j = 1\}$
12. $\quad\quad Q := [(S_0, I \setminus \{j\}, i \vee j), (S_1, I \setminus \{j\}, i \vee j)] \mathbin{++} Q$

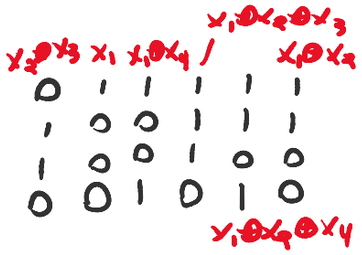
$(i \vee j = j$ if $i = e$, i otherwise$)$

Example

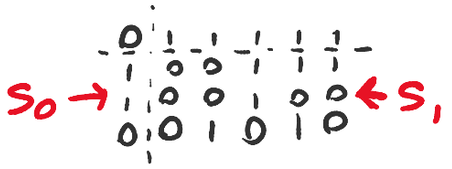
Suppose we wish to synthesize the phase polynomial

$$f(\vec{x}) = \frac{1}{8}\left[x_2 \oplus x_3 + x_1 + x_1 \oplus x_4 + x_1 \oplus x_2 \oplus x_3 + x_1 \oplus x_2 \oplus x_4 + x_1 \oplus x_2\right]$$

We can visualize Gray-synth as operating on a matrix with columns corresponding to each parity:



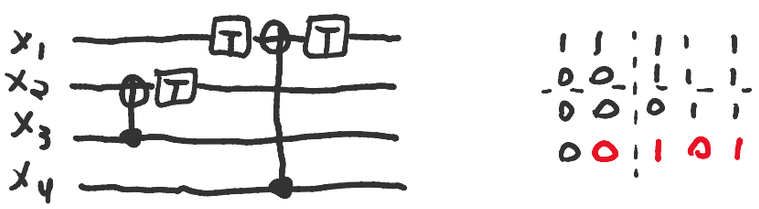
We first partition into S_0 & S_1 by the first row



To compute the only parity in S_0, $x_2 \oplus x_3$, we perform a CNOT between 3 & 2, then update the matrix

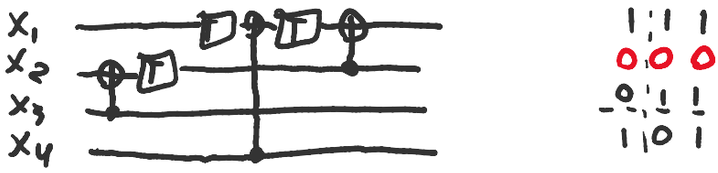
We're done with S_0 now, so we pop it off the queue and continue with the remaining, selecting row 2 to construct the next set of co-factors

The first parity of S_0 is already "computed", so we only need to perform a CNOT between row 4 & row 1:



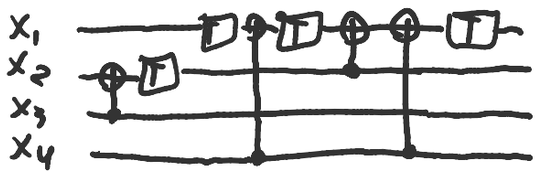
Example cont.

Continuing on with S_1, we zero out the second row with $CNOT_{2,1}$ then partition on row 3



1 1 1
0 0 0
0 1 1
1 0 1

Next we $CNOT_{4,1}$ and move on to S_1



1 1
0 0
1 1
1 0

At this point we perform $CNOT_{3,1}$, and then divide into co-factors S_0 & S_1, which require 0 & 1 $CNOT_{n,1}$ resp