

# 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  $\{X, CNOT, R_n = R_2^{(\pi/n)}\}$

### Fact

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

### Fact

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

$$(A, \vec{b}) \in GA(n, \mathbb{Z}_2) \cong GL(n, \mathbb{Z}_2) \times \mathbb{Z}_2^n$$

$$(A, \vec{b}) \vec{x} = A\vec{x} + \vec{b}$$

$$\text{E.g. } X|xy\rangle = |x\oplus y\rangle = |(I, 1)x\rangle$$

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  $|xy\rangle$   $\xrightarrow{\text{CNOT}}$   $|x\rangle|x\oplus y\rangle$   
 $\xrightarrow{T}$   $w^{x\oplus y}|x\rangle|x\oplus y\rangle$

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

## Def'n

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

# Representation of (NOT-dihedral groups)

Prop (phase polynomial representation)

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

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

where  $A \in \text{AGL}(n, \mathbb{Z}_2)$ , and  $P: \mathbb{Z}_2^n \rightarrow \mathbb{Z}_m$  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

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

$$\begin{aligned} \text{Then } [X_i \cdot C]|\vec{x}\rangle &\mapsto e^{2\pi i/m P(\vec{x})} (X_i |A\vec{x} + \vec{b}\rangle) \\ &\mapsto e^{2\pi i/m P(\vec{x})} |A\vec{x} + (\vec{b} + \vec{e}_i)\rangle \end{aligned}$$

$$\begin{aligned} [\alpha_{ij} \cdot C]|\vec{x}\rangle &\mapsto e^{2\pi i/m P(\vec{x})} ((X_{ij} |A\vec{x} + \vec{b}\rangle) \\ &\mapsto e^{2\pi i/m P(\vec{x})} |E_{ij} A\vec{x} + E_{ij} \vec{b}\rangle \end{aligned}$$

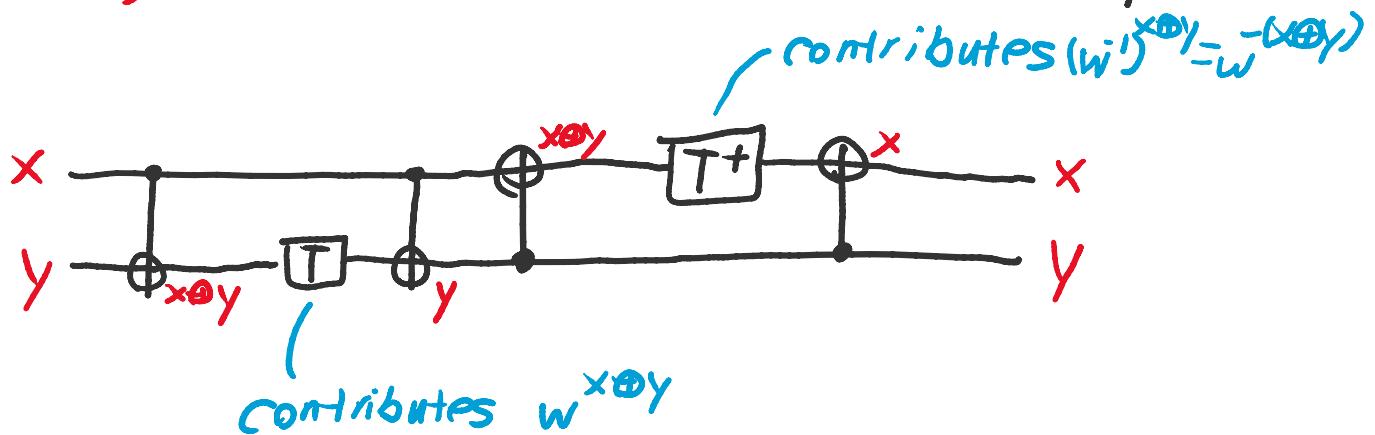
$$\begin{aligned} [(R_k)_i \cdot C]|\vec{x}\rangle &\mapsto e^{2\pi i/m P(\vec{x})} ((R_k)_i |A\vec{x} + \vec{b}\rangle) \\ &\mapsto e^{2\pi i/m P(\vec{x})} e^{2\pi i/m (A_i \vec{x} \oplus b_i)} |A\vec{x} + \vec{b}\rangle \\ &\mapsto e^{2\pi i/m (P(\vec{x}) + (-1)^{b_i} X_{A_i}(\vec{x}) + b_i)} |A\vec{x} + \vec{b}\rangle \end{aligned}$$

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

# Annotated Circuits

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

E.g.



Hence,

$$|x\rangle|y\rangle \xrightarrow{\text{CNOT}} w^{x\oplus y} w^{-(x\oplus y)} |x\rangle|y\rangle = |x\rangle|y\rangle$$

## Prop.

Given a circuit  $C$  over  $\{X, \text{CNOT}, R_n\}$ , 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  $\vec{y} \in \mathbb{Z}_2^n$  where  $y_i \neq 0$  the **Support** of a phase polynomial  $P$ , denoted **Supp( $P$ )**. While phase polynomials are **NOT unique**, P.Q.

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

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

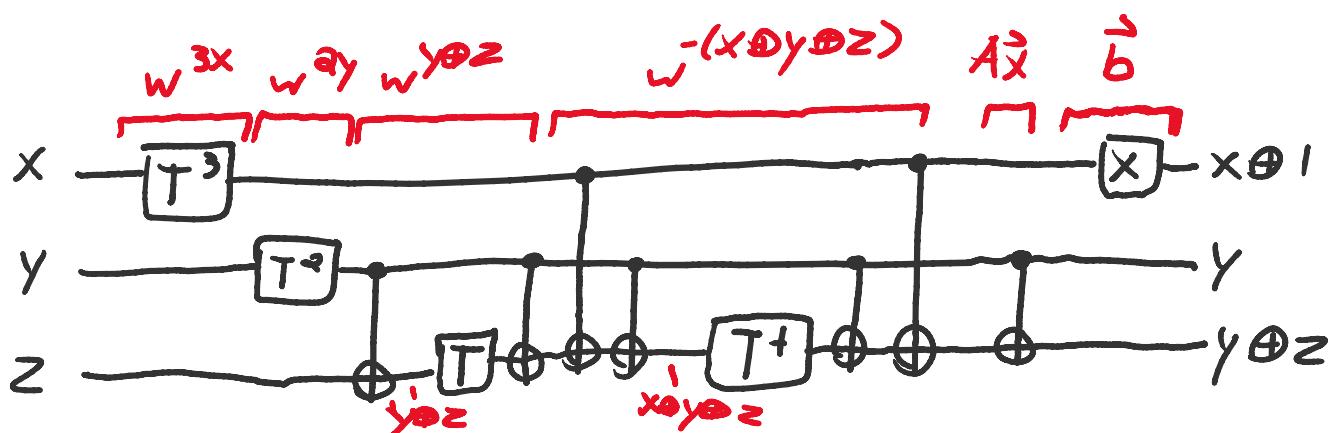
$$|\text{Supp}(P)| \leq \# R_n \text{ gates in } C$$

# Synthesis of (NOT-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  $U: |xyz\rangle \mapsto e^{i\pi/8} |w\rangle^{3x+2y+(y\oplus z)-(x\otimes y\otimes z)} |(x\oplus y)(y\oplus z)\rangle$

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



## Prop

Let  $P(\vec{x}) = \sum_y \alpha_y x_y(\vec{x})$ ,  $\alpha_y \in \mathbb{Z}_n$  and  $(A, \vec{b}) \in GA(m, \mathbb{Z}_2)$ .

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

## Note:

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

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

# Synthesizing the Toffoli gate

Recall that  $\text{Toffoli} = \overline{\text{CCZ}}$ , where  $\text{CCZ}$  is the CCZ gate defined by

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

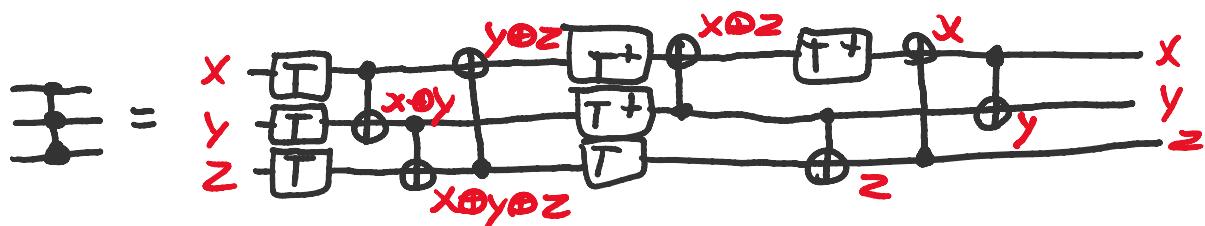
Taking the Fourier expansion\* of  $f(x,y,z) = xyz$  gives us

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

↑      ↑      ↑      ↑      ↑      ↑      ↑

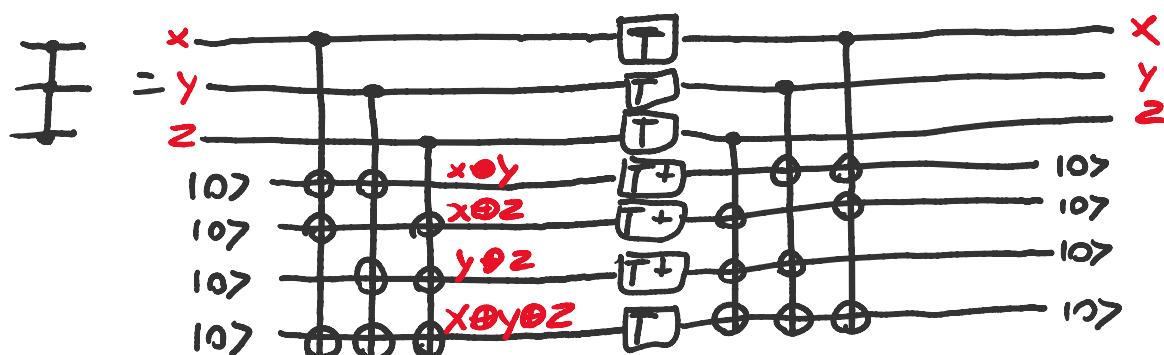
this is a phase polynomial over  $\sqrt{-1} = w = e^{\pi i/4}$

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  $f: \mathbb{Z}_2^n \rightarrow \mathbb{R}$ , an expression of  $f$  as  $f(\vec{x}) = \sum_{\vec{y}} \hat{f}(\vec{y}) X_{\vec{y}}(\vec{x})$

# The phase polynomial method

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

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

(phase gate, e.g. T gate, optimization for {CNOT, X, R<sub>m</sub>})

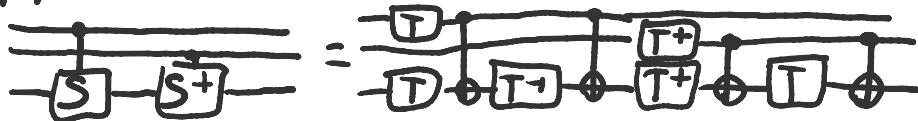
$$C \xrightarrow{\text{annotate}} (P, A, \vec{b}) \xrightarrow{\text{synthesize}} C'$$

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

profit :)

(Example)

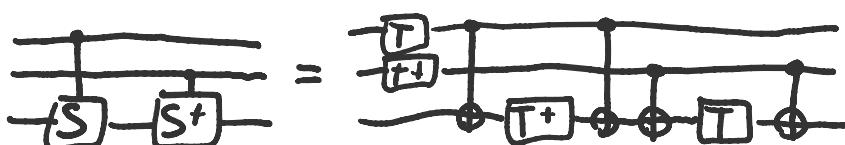
Suppose we want to optimize the circuit:



We start by computing the phase polynomial representation:

$$\begin{aligned} |xyz\rangle &\mapsto_w^{x \cdot z - (x \otimes z)} -y \cdot z + (y \otimes z) |xyz\rangle \\ &\mapsto_w^{x - (x \otimes z)} -y + (y \otimes z) |xyz\rangle \end{aligned}$$

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  $(P, A, \tilde{B})$  in some cost metric?

Synthesizing an optimal circuit for  $(P, A, \tilde{B})$  is known as a phase polynomial synthesis problem.

metric	result
$R_m$ -depth	poly-time via Matroid partitioning ( $t\text{-par}$ )
$R_m$ -count	Equivalent to min-distance decoding for $RM^{\pm}(n, n - \lfloor \log_2(m) \rfloor - 1)$ (hard!)
$CNOT$ -count	NP-hard in restricted cases asymptotically optimal heuristic (Gray-Synth)

Note:  $\lfloor \log_2(m) \rfloor$  is the 2-adic valuation of  $m$   
(highest power of 2 that divides  $m$ )

(Upper & lower bounds on Solutions)

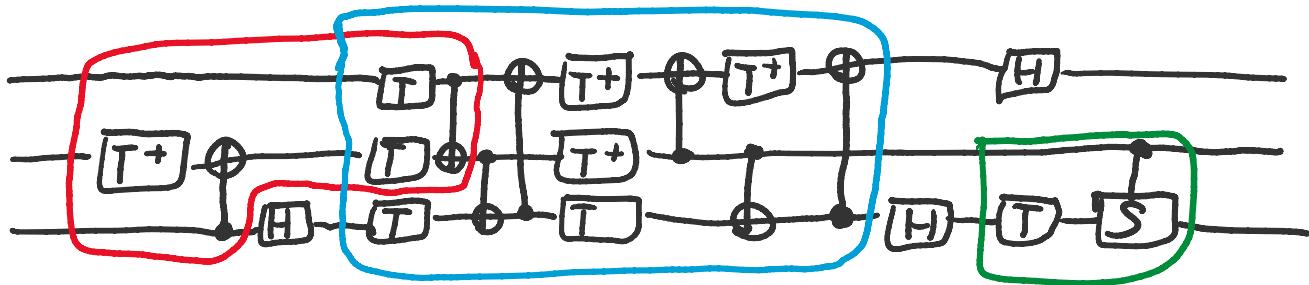
metric	lower bound	upper bound
$R_m$ -depth	$O(1)$ (with ancillas)	$O(1)$ (with ancillas)
$R_m$ -count	Covering radius of $RM^{\pm}(n, n - \lfloor \log_2(m) \rfloor - 1)$	$O\left(\frac{n^{\lfloor \log_2(m) \rfloor - 1}}{\lfloor \log_2(m) \rfloor !}\right)/O(n^2)$ for T-count
$CNOT$ -count	$O(n^2)$ (from gray code)	$O(n^2)$ (from gray code)

# Choice of subcircuits

Even with **optimal** re-synthesis, different partitionings of a circuit may result in different costs.

E.g.

Consider the below circuit. There are **3 maximal CNOT-dihedral sub-circuits**: 1, 2, & 3



Since 1 and 2 overlap, the order of re-synthesis matters — re-synthesizing 1 first may remove the gates in the intersection, but associating those gates with 2 instead could have led to better optimization.

One option is to mark gates in the intersection and try to find a **global optimum** over a set of synthesis problems. We're still just *sweeping the floors on the Titanic* though because the optimization fails to take into account the **whole circuit**. As in classical compilers, the best optimizations are **whole program optimizations** typically using static analysis to make it tractable.