On the CNOT-complexity of CNOT-PHASE circuits

Matthew Amy$^{1,2}$  Parsiad Azimzadeh$^{2}$  Michele Mosca$^{1,2}$

$^1$Institute for Quantum Computing, Waterloo, Canada

$^2$University of Waterloo, Waterloo, Canada

Theory of Quantum Computation, Communication and Cryptography
July 18th, 2018
Sea knot???

CNOT/CZ optimization problems

<table>
<thead>
<tr>
<th>Gate set</th>
<th>Complexity</th>
<th>State-of-the-art</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNOT</td>
<td>???</td>
<td>Asymptotically optimal synthesis¹</td>
</tr>
<tr>
<td>CZ-PHASE</td>
<td>Polynomial</td>
<td>Optimal synthesis</td>
</tr>
<tr>
<td>CNOT-PHASE</td>
<td>???</td>
<td>Re-write rules</td>
</tr>
<tr>
<td>Clifford</td>
<td>???</td>
<td>Re-write rules</td>
</tr>
<tr>
<td>Clifford + T</td>
<td>???</td>
<td>Re-write rules</td>
</tr>
</tbody>
</table>

Assuming completely connected topology...

CNOT-PHASE: Circuits over CNOT and $R_Z(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i \theta} \end{pmatrix}$

¹Patel, Markov and Hayes, *Optimal synthesis of linear reversible circuits*
Why CNOT-PHASE?

Phase folding/\(T\)-par uses \(T\)-depth optimal CNOT-PHASE synthesis as a sub-routine

Amy, Maslov and Mosca, *Polynomial-time \(T\)-depth Optimization of Clifford+\(T\) circuits via Matroid Partitioning*
Why CNOT-PHASE?

Phase folding/\(T\)-par uses \(T\)-depth optimal CNOT-PHASE synthesis as a sub-routine

Idea: replace \(T\)-depth optimal with CNOT-optimal!

Amy, Maslov and Mosca, *Polynomial-time T-depth Optimization of Clifford+T circuits via Matroid Partitioning*
We...

- Show that in certain cases, minimizing the number of CNOT gates is equivalent to finding a minimal CNOT circuit cycling through a set of parities of the inputs
- Show that cycling through a set of parities is NP-hard if
  - all CNOT gates have the same target, or
  - the circuit inputs are not linearly independent
- Give a new heuristic optimization algorithm
Introduction

Parity networks

Complexity of minimal parity network synthesis

Heuristic synthesis

Experiments

Conclusion
The sum-over-paths form

Recall the basis state action of CNOT and Phase gates:

\[
\text{CNOT} : |x\rangle|y\rangle \mapsto |x\rangle|x \oplus y\rangle
\]

\[
R_Z(\theta) : |x\rangle \mapsto e^{2\pi i \theta x} |x\rangle
\]

We call this basis state action the sum-over-paths (SOP) form.
The sum-over-paths form

Recall the basis state action of CNOT and Phase gates:

\[
\text{CNOT} : |x\rangle|y\rangle \mapsto |x\rangle|x \oplus y\rangle
\]

\[
R_Z(\theta) : |x\rangle \mapsto e^{2\pi i \theta x}|x\rangle
\]

We call this basis state action the sum-over-paths (SOP) form

Definition

The SOP form of a CNOT-PHASE circuit \(C\) is a pair \((f, A)\) where

\(f : \mathbb{F}_2^n \to \mathbb{R}\) is a pseudo-Boolean function given by

\[
f(x) = \sum_{y \in \mathbb{F}_2^n} \hat{f}(y) \chi_y(x), \quad \chi_y(x) = x_1 y_1 \oplus \cdots \oplus x_n y_n
\]

\(A \in \text{GL}(n, \mathbb{F}_2)\) is a linear permutation such that \(U_C : |x\rangle \mapsto e^{2\pi i f(x)} |Ax\rangle\)
Computing the sum-over-paths

Consider an implementation of \( CCZ \):

\[
\begin{align*}
R_Z \left( \frac{1}{8} \right) & \quad \bullet & \quad R_Z \left( \frac{7}{8} \right) & \quad \bullet & \quad R_Z \left( \frac{1}{8} \right) & \quad \bullet & \quad R_Z \left( \frac{7}{8} \right) \\
R_Z \left( \frac{1}{8} \right) & \quad \bullet & \quad R_Z \left( \frac{1}{8} \right) & \quad \bullet & \quad R_Z \left( \frac{7}{8} \right) & \quad \bullet & \quad R_Z \left( \frac{7}{8} \right)
\end{align*}
\]
Computing the circuit sum-over-paths

First annotate...

\[ x_1 \equiv 100 \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ 010 \]
\[ x_2 \equiv 010 \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ x_3 \equiv 001 \]
\[ R_Z \left( \frac{7}{8} \right) \]
\[ 001 \]

Then add the phase factors

\[ |x\rangle \mapsto e^{2\pi i \left( x_1 + x_2 + 7(x_1 \oplus x_3) + 7(x_2 \oplus x_3) + (x_1 \oplus x_2 \oplus x_3) + 7(x_1 \oplus x_2) \right)} |x\rangle \]
Computing the circuit sum-over-paths

First annotate...

\[
\begin{align*}
    x_1 &\equiv 100 & R_Z \left( \frac{1}{8} \right) \quad \oplus \quad 101 &\rightarrow & R_Z \left( \frac{7}{8} \right) \quad \oplus \quad 111 &\rightarrow & R_Z \left( \frac{1}{8} \right) \quad \oplus \quad 110 &\rightarrow & R_Z \left( \frac{7}{8} \right) \quad \oplus \quad 100 \\
    x_2 &\equiv 010 & R_Z \left( \frac{1}{8} \right) \quad \oplus \quad 011 &\rightarrow & R_Z \left( \frac{7}{8} \right) \quad \oplus \quad 001 &\rightarrow & R_Z \left( \frac{1}{8} \right) \quad \oplus \quad 001 &\rightarrow & 010 \\
    x_3 &\equiv 001 & R_Z \left( \frac{1}{8} \right) \quad \oplus \quad 011 &\rightarrow & R_Z \left( \frac{7}{8} \right) \quad \oplus \quad 001 &\rightarrow & R_Z \left( \frac{1}{8} \right) \quad \oplus \quad 001 &\rightarrow & 001
\end{align*}
\]

...Then add the phase factors

\[
|\mathbf{x}\rangle \mapsto e^{\frac{2\pi i}{8}}(x_1 |\mathbf{x}\rangle)
\]
Computing the circuit sum-over-paths

First annotate...

\[ \begin{align*}
  x_1 & \equiv 100 & & R_Z \left( \frac{1}{8} \right) & & 101 & & R_Z \left( \frac{7}{8} \right) & & 111 & & R_Z \left( \frac{1}{8} \right) & & 110 & & R_Z \left( \frac{7}{8} \right) & & 100 \\
  x_2 & \equiv 010 & & R_Z \left( \frac{1}{8} \right) & & 011 & & R_Z \left( \frac{7}{8} \right) & & 001 & & R_Z \left( \frac{1}{8} \right) & & 010 \\
  x_3 & \equiv 001
\end{align*} \]

...Then add the phase factors

\[ |x\rangle \mapsto e^{\frac{2\pi i}{8}(x_1 + x_2)} |x\rangle \]
Computing the circuit sum-over-paths

First annotate...

\[ x_1 \equiv 100 \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ 101 \]
\[ R_Z \left( \frac{7}{8} \right) \]
\[ 111 \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ 110 \]
\[ R_Z \left( \frac{7}{8} \right) \]
\[ 100 \]

\[ x_2 \equiv 010 \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ 011 \]
\[ \bullet \]
\[ R_Z \left( \frac{7}{8} \right) \]
\[ 001 \]
\[ \bullet \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ 001 \]

\[ x_3 \equiv 001 \]
\[ \bullet \]
\[ \bullet \]
\[ \bullet \]
\[ \bullet \]
\[ \bullet \]

...Then add the phase factors

\[ |x\rangle \mapsto e^{\frac{2\pi i}{8}(x_1 + x_2 + 7(x_1 \oplus x_3))} |x\rangle \]
Computing the circuit sum-over-paths

First annotate...

\[ x_1 \equiv 100 \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ 101 \]
\[ R_Z \left( \frac{7}{8} \right) \]
\[ 111 \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ 110 \]
\[ R_Z \left( \frac{7}{8} \right) \]
\[ 100 \]

\[ x_2 \equiv 010 \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ 011 \]

\[ x_3 \equiv 001 \]
\[ 001 \]

...Then add the phase factors

\[ |x\rangle \mapsto e^{\frac{2\pi i}{8}(x_1+x_2+7(x_1\oplus x_3)+7(x_2\oplus x_3))} |x\rangle \]
Computing the circuit sum-over-paths

First annotate...

\[ x_1 \equiv 100 \quad R_Z \left( \frac{1}{8} \right) \quad 101 \quad R_Z \left( \frac{7}{8} \right) \quad 111 \quad R_Z \left( \frac{1}{8} \right) \quad 110 \quad R_Z \left( \frac{7}{8} \right) \quad 100 \]

\[ x_2 \equiv 010 \quad R_Z \left( \frac{1}{8} \right) \]

\[ x_3 \equiv 001 \]

...Then add the phase factors

\[ |x\rangle \mapsto e^{\frac{2\pi i}{8}(x_1 + x_2 + 7(x_1 \oplus x_3) + 7(x_2 \oplus x_3) + (x_1 \oplus x_2 \oplus x_3))} |x\rangle \]
Computing the circuit sum-over-paths

First annotate...

\[ \begin{align*}
    x_1 &\equiv 100 \\
    x_2 &\equiv 010 \\
    x_3 &\equiv 001
\end{align*} \]

...Then add the phase factors

\[ |x\rangle \mapsto e^{\frac{2\pi i}{8}(x_1+x_2+7(x_1\oplus x_3)+7(x_2\oplus x_3)+(x_1\oplus x_2\oplus x_3)+7(x_1\oplus x_2))} |x\rangle \]
Computing the circuit sum-over-paths

First annotate...

\[ x_1 \equiv 100 \quad R_Z \left( \frac{1}{8} \right) \quad 101 \quad R_Z \left( \frac{7}{8} \right) \quad 111 \quad R_Z \left( \frac{1}{8} \right) \quad 110 \quad R_Z \left( \frac{7}{8} \right) \quad 100 \]

\[ x_2 \equiv 010 \quad R_Z \left( \frac{1}{8} \right) \quad 010 \quad 011 \quad R_Z \left( \frac{7}{8} \right) \quad 001 \quad R_Z \left( \frac{1}{8} \right) \quad 010 \]

\[ x_3 \equiv 001 \quad \]...

...Then add the phase factors

\[ |x\rangle \mapsto e^{\frac{2\pi i}{8}(x_1 + x_2 + 7(x_1 \oplus x_3) + 7(x_2 \oplus x_3) + (x_1 \oplus x_2 \oplus x_3) + 7(x_1 \oplus x_2) + x_3)}|x\rangle \]
Computing the circuit sum-over-paths

First annotate...

\[ x_1 \equiv 100 \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ x_2 \equiv 010 \]
\[ R_Z \left( \frac{1}{8} \right) \]
\[ x_3 \equiv 001 \]
\[ R_Z \left( \frac{7}{8} \right) \]

...Then add the phase factors

\[ |x\rangle \mapsto e^{\frac{2\pi i}{8}(x_1 + x_2 + 7(x_1 \oplus x_3) + 7(x_2 \oplus x_3) + (x_1 \oplus x_2 \oplus x_3) + 7(x_1 \oplus x_2) + x_3)} |x\rangle \]
\[ \mapsto e^{\frac{2\pi i}{2} x_1 x_2 x_3} |x\rangle \]
An observation

Recall:

\[ CS^\dagger : |x_1 x_2\rangle \leftrightarrow e^{\frac{2\pi i}{4} x_1 x_2} |x_1 x_2\rangle \]

\[ \leftrightarrow e^{\frac{2\pi i}{8} (7x_1 + 7x_2 + x_1 \oplus x_2)} |x_1 x_2\rangle \]

Can use the same CNOT structure as CCZ to implement \( CS^\dagger \)!
An observation

Recall:

$$CS^\dagger : |x_1 x_2\rangle \mapsto e^{\frac{2\pi i}{4} 3 x_1 x_2} |x_1 x_2\rangle$$

$$\mapsto e^{\frac{2\pi i}{8} (7 x_1 + 7 x_2 + x_1 \oplus x_2)} |x_1 x_2\rangle$$

Can use the same CNOT structure as CCZ to implement $CS^\dagger$!
Definition

A **parity network** for a set $S \subseteq \mathbb{F}_2^n$ is an $n$-qubit circuit $C$ over CNOT gates where each $y \in S$ appears in the annotated circuit.

A parity network is **pointed at** $A \in \text{GL}(n, \mathbb{F}_2)$ if it implements the overall linear transformation $A$. 
Parity networks

Definition

A parity network for a set $S \subseteq \mathbb{F}_2^n$ is an $n$-qubit circuit $C$ over CNOT gates where each $y \in S$ appears in the annotated circuit.

A parity network is pointed at $A \in GL(n, \mathbb{F}_2)$ if it implements the overall linear transformation $A$.

E.g. the CNOT gates of $CCZ$,

```
100   101   111   110   100
100   101   111   110   100
010   011   001   010   010
001   011   001   001   001
```

is a parity network for $S = \{100, 010, 001, 110, 101, 011, 111\}$ pointed at $A = I$. 
A CNOT-minimal circuit with SOP form \((f, A)\) necessarily gives a minimal parity network for \(\text{supp}(\hat{f})\) pointed at \(A\)
However...
A minimal parity network for supp($\hat{f}$) may not give a CNOT-minimal circuit across equivalent SOP forms.
A minimal parity network for supp(\(\hat{f}\)) may not give a CNOT-minimal circuit across equivalent SOP forms.

E.g., \((\frac{1}{2}(x_1 \oplus x_2), I)\) and \((\frac{1}{2}x_1 + \frac{1}{2}x_2, I)\) give equivalent unitaries but have minimal parity network implementations.
Main result

**Theorem**

CNOT minimization of CNOT-PHASE circuits is at least as hard as synthesizing a minimal parity network.
Main result

**Theorem**

CNOT minimization of CNOT-PHASE circuits is at least as hard as synthesizing a minimal parity network

**Intuition:**

- If \((f, A) \sim (f', A')\), then \(A = A'\) and \(f' = f + k\) for \(k : \mathbb{F}_2^n \rightarrow \mathbb{Z}\)
Main result

Theorem

CNOT minimization of CNOT-PHASE circuits is at least as hard as synthesizing a minimal parity network

Intuition:

- If \((f, A) \sim (f', A')\), then \(A = A'\) and \(f' = f + k\) for \(k : \mathbb{F}_2^n \to \mathbb{Z}\).
- The Fourier coefficients of \(k\) have even order in \(\mathbb{R}/\mathbb{Z}\).
Main result

Theorem

CNOT minimization of CNOT-PHASE circuits is at least as hard as synthesizing a minimal parity network

Intuition:

- If \( (f, A) \sim (f', A') \), then \( A = A' \) and \( f' = f + k \) for \( k : \mathbb{F}_2^n \to \mathbb{Z} \)
- The Fourier coefficients of \( k \) have even order in \( \mathbb{R}/\mathbb{Z} \)
- If no elements of \( \hat{f} \) have even order in \( \mathbb{R}/\mathbb{Z} \), then

\[
\text{supp}(\hat{f}') \subseteq \text{supp}(\hat{f})
\]
Introduction

Parity networks

Complexity of minimal parity network synthesis

Heuristic synthesis

Experiments

Conclusion
Goal:

Prove that the minimal parity network problem (MPNP) is NP-hard

Obvious reductions don’t work due to shortcuts
Goal:

*Prove that the minimal parity network problem (MPNP) is NP-hard*

Obvious reductions don’t work due to **shortcuts**
A graphical interpretation
Conjecture

If for all \( y \in S, y_i = 1 \), then there exists a minimal parity network for \( S \) where each CNOT targets bit \( i \).
Conjecture

*If for all* $y \in S$, $y_i = 1$, *then there exists a minimal parity network for* $S$ *where each CNOT targets bit* $i$.
Theorem

The fixed-target minimal parity network problem is NP-complete

Proof:

Reduction from traveling salesman on the hypercube

\(^2\)Ernvall, Katajainen, and Penttonen, *NP-completeness of the Hamming salesman problem*
If some inputs are linearly dependent, fewer gates may be needed to implement a parity network.

E.g., $S = \{111\}$

![Diagram of minimal parity network with encoded inputs]
If some inputs are linearly dependent, fewer gates may be needed to implement a parity network

E.g., $S = \{111\}$

Direct applications to phase folding with ancillas!
Theorem

The encoded input minimal parity network problem is NP-complete

Proof:

Reduction from maximum-likelihood decoding

\(^3\)Berlekamp, McEliece, and van Tilborg, *On the inherent intractability of certain coding problems*
Introduction

Parity networks

Complexity of minimal parity network synthesis

Heuristic synthesis

Experiments

Conclusion
Given $S \subseteq \mathbb{F}_2^n$, synthesize an efficient parity network for $S$
For $S = \mathbb{F}_2^n \parallel x, x \in \mathbb{F}_2^m$, minimal parity network is the Gray code and can be computed greedily

E.g.,

$S = \mathbb{F}_2^3 \parallel 1 = \{0001, 1001, 0101, 1101, 0011, 1011, 0111, 1111\}$
Bases cases

For $S = \mathbb{F}_2^n$, case is similar

E.g.,

$S = \mathbb{F}_2^3$

\[
\begin{array}{cccc}
100 & \bullet & 110 & \bullet & 100 \\
010 & 110 & 101 & 011 & 111 \\
001 & 110 & 111 & & \\
\end{array}
\]

$S = \mathbb{F}_2^4$

\[
\begin{array}{cccccccc}
1000 & \bullet & 1100 & \bullet & 1000 & \\
0100 & 1100 & 1010 & 0110 & 1110 & \\
0010 & 1110 & 1001 & 0101 & 1101 & \\
0001 & 1101 & 0011 & 1011 & 0111 & \\
\end{array}
\]
Main idea:

Try to identify subsets $S'$ of $S$ which have the form $S' \simeq \mathbb{F}_2^n \parallel x$, and synthesize those greedily
The **Gray-synth** algorithm

1. Start with a singleton stack containing the set $S$
2. Pop a set $S'$ off the stack
3. If $x_i \oplus x_j$ appears in every parity of $S'$,
   - Apply a CNOT between bits $i$ and $j$, and
   - Adjust all subsets remaining on the stack accordingly
4. Pick some row $i$ maximizing the number of parities in $S'$
   which **either contain or do not contain** $x_i$
5. Set $S_b = \{ x \in S' \mid x_i = b \}$ and push $S_1$, $S_0$ onto the stack
6. Go to step 2

**Invariant:** remaining parities are expressed over the current basis
- Avoids “uncomputing” or backtracking
Example

Parity network for \( S = \{0110, 1000, 1001, 1110, 1101, 1100\} \)

\[
\begin{bmatrix}
0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

Columns are remaining parities
Box is current top of the stack
White rows haven't been partitioned
Grey rows have been partitioned
Example

Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

Columns are remaining parities
Box is current top of the stack
White rows haven’t been partitioned
Grey rows have been partitioned
Example

Parity network for \( S = \{0110, 1000, 1001, 1110, 1101, 1100\} \)

- Columns are remaining parities
- Box is current top of the stack
- White rows haven’t been partitioned
- Grey rows have been partitioned
Example

Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

Columns are remaining parities
Box is current top of the stack
White rows haven't been partitioned
Grey rows have been partitioned
Example
Parity network for \( S = \{0110, 1000, 1001, 1110, 1101, 1100\} \)

- Columns are remaining parities
- Box is current top of the stack
- White rows haven’t been partitioned
- Grey rows have been partitioned
Example
Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

- Columns are remaining parities
- Box is current top of the stack
- White rows haven’t been partitioned
- Grey rows have been partitioned
Example

Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

Columns are remaining parities
Box is current top of the stack
White rows haven’t been partitioned
Grey rows have been partitioned
Example
Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

- Columns are remaining parities
- Box is current top of the stack
- White rows haven’t been partitioned
- Grey rows have been partitioned
Example
Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

Columns are remaining parities
Box is current top of the stack
White rows haven’t been partitioned
Grey rows have been partitioned
Example
Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$\rightarrow$

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- Columns are remaining parities
- Box is current top of the stack
- White rows haven’t been partitioned
- Grey rows have been partitioned
Example
Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

- Columns are remaining parities
- Box is current top of the stack
- White rows haven’t been partitioned
- Grey rows have been partitioned
Example

Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

- Columns are remaining parities
- Box is current top of the stack
- White rows haven’t been partitioned
- Grey rows have been partitioned
Example

Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

- Columns are remaining parities
- Box is current top of the stack
- White rows haven’t been partitioned
- Grey rows have been partitioned
Example

Parity network for  \( S = \{0110, 1000, 1001, 1110, 1101, 1100\} \)

- Columns are remaining parities
- Box is current top of the stack
- White rows haven’t been partitioned
- Grey rows have been partitioned
Example
Parity network for $S = \{0110, 1000, 1001, 1110, 1101, 1100\}$

- Columns are remaining parities
- Box is current top of the stack
- White rows haven’t been partitioned
- Grey rows have been partitioned
Introduction

Parity networks

Complexity of minimal parity network synthesis

Heuristic synthesis

Experiments

Conclusion
Performance vs. brute force

- Data collected across all sets of parities on 4 bits
- **GRAY-SYNTH** within 15% of optimal on average for $|S| = 8$

https://github.com/meamy/feynman
## Benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>$n$</th>
<th>Base</th>
<th>Nam et al. (L)</th>
<th>$T$-par (GRAY-SYNTH)</th>
<th>% Red.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CNOT</td>
<td>Time</td>
<td>CNOT</td>
</tr>
<tr>
<td>Grover.5</td>
<td>9</td>
<td></td>
<td>336</td>
<td>0.027</td>
<td>210</td>
</tr>
<tr>
<td>Mod 5.4</td>
<td>5</td>
<td></td>
<td>32</td>
<td>&lt; 0.001</td>
<td>28</td>
</tr>
<tr>
<td>VBE-Adder.3</td>
<td>10</td>
<td></td>
<td>80</td>
<td>0.073</td>
<td>76</td>
</tr>
<tr>
<td>CSLA-MUX.3</td>
<td>15</td>
<td></td>
<td>90</td>
<td>0.097</td>
<td>136</td>
</tr>
<tr>
<td>CSUM-MUX.9</td>
<td>30</td>
<td></td>
<td>196</td>
<td>0.112</td>
<td>189</td>
</tr>
<tr>
<td>QCLA-Com.7</td>
<td>24</td>
<td></td>
<td>215</td>
<td>0.145</td>
<td>356</td>
</tr>
<tr>
<td>QCLA-Mod.7</td>
<td>26</td>
<td></td>
<td>441</td>
<td>0.165</td>
<td>352</td>
</tr>
<tr>
<td>QCLA-Adder.10</td>
<td>36</td>
<td></td>
<td>267</td>
<td>0.180</td>
<td>71</td>
</tr>
<tr>
<td>Adder.8</td>
<td>24</td>
<td></td>
<td>466</td>
<td>0.210</td>
<td>352</td>
</tr>
<tr>
<td>RC-Adder.6</td>
<td>14</td>
<td></td>
<td>104</td>
<td>0.230</td>
<td>71</td>
</tr>
<tr>
<td>Mod-Red.21</td>
<td>11</td>
<td></td>
<td>122</td>
<td>0.240</td>
<td>84</td>
</tr>
<tr>
<td>Mod-Mult.55</td>
<td>9</td>
<td></td>
<td>55</td>
<td>0.260</td>
<td>45</td>
</tr>
<tr>
<td>Mod-Adder.1024</td>
<td>28</td>
<td></td>
<td>2005</td>
<td>0.280</td>
<td>1376</td>
</tr>
<tr>
<td>Cycle 17.3</td>
<td>35</td>
<td></td>
<td>4532</td>
<td>2.618</td>
<td>2998</td>
</tr>
<tr>
<td>GF($2^{32}$)-Mult</td>
<td>96</td>
<td></td>
<td>7292</td>
<td>5.571</td>
<td>6658</td>
</tr>
<tr>
<td>GF($2^{64}$)-Mult</td>
<td>192</td>
<td></td>
<td>28861</td>
<td>114.310</td>
<td>25966</td>
</tr>
<tr>
<td>Ham.15 (low)</td>
<td>17</td>
<td></td>
<td>259</td>
<td>0.043</td>
<td>208</td>
</tr>
<tr>
<td>Ham.15 (med)</td>
<td>17</td>
<td></td>
<td>574</td>
<td>0.089</td>
<td>351</td>
</tr>
<tr>
<td>Ham.15 (high)</td>
<td>20</td>
<td></td>
<td>2489</td>
<td>0.376</td>
<td>1500</td>
</tr>
<tr>
<td>HWB.6</td>
<td>7</td>
<td></td>
<td>131</td>
<td>0.006</td>
<td>111</td>
</tr>
<tr>
<td>HWB.8</td>
<td>12</td>
<td></td>
<td>7508</td>
<td>1.706</td>
<td>6719</td>
</tr>
<tr>
<td>QFT.4</td>
<td>5</td>
<td></td>
<td>48</td>
<td>0.005</td>
<td>47</td>
</tr>
<tr>
<td>$\Lambda_5(X)$</td>
<td>9</td>
<td></td>
<td>49</td>
<td>&lt; 0.001</td>
<td>30</td>
</tr>
<tr>
<td>$\Lambda_5(X)$ (Barenco)</td>
<td>9</td>
<td></td>
<td>84</td>
<td>&lt; 0.001</td>
<td>60</td>
</tr>
<tr>
<td>$\Lambda_{10}(X)$</td>
<td>19</td>
<td></td>
<td>119</td>
<td>&lt; 0.001</td>
<td>70</td>
</tr>
<tr>
<td>$\Lambda_{10}(X)$ (Barenco)</td>
<td>19</td>
<td></td>
<td>224</td>
<td>0.001</td>
<td>144</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>23.3</td>
<td></td>
</tr>
</tbody>
</table>

https://github.com/meamy/feynman
Introduction

Parity networks

Complexity of minimal parity network synthesis

Heuristic synthesis

Experiments

Conclusion
Conclusion

In this talk...

- Parity networks characterize the CNOT complexity of CNOT-PHASE circuits for a particular phase function.
- CNOT minimization is at least as hard as synthesizing a minimal parity network.
- Synthesizing a minimal parity network is NP-hard when targets are fixed or inputs are encoded.
- A heuristic parity network synthesis algorithm & benchmarks.
Future work

- Proof of hardness for the general problem
- Synthesis algorithm that combines parity network synthesis with an output linear permutation
- Adding topology constraints
Thank you!