$T$-count optimization and Reed-Muller codes

Matthew Amy

Joint work with Michele Mosca
Institute for Quantum Computing, University of Waterloo


BIRS Quantum Computer Science Workshop, Banff
April 22 2016
Why optimize $T$ count?
Why optimize $T$ count?
\{\text{CNOT, } T\} \text{ circuits}

Recall \((x, y \in \mathbb{F}_2)\):

\[
\text{CNOT} : |xy\rangle \mapsto |x(x \oplus y)\rangle
\]
\[
T : |x\rangle \mapsto \omega^x|x\rangle, \quad \omega = e^{i\pi/4}
\]
\{\text{CNOT, } T\} \text{ circuits}

Recall \((x, y \in \mathbb{F}_2)\):

\[
\text{CNOT} : |xy\rangle \mapsto |x(x \oplus y)\rangle
\]
\[
T : |x\rangle \mapsto \omega^x |x\rangle, \quad \omega = e^{i\pi/4}
\]

Proposition

A unitary \(U\) can be implemented over \(\text{CNOT}\) and \(T\) gates if and only if

\[
U : |x\rangle \mapsto \omega^{P(x)} |f(x)\rangle
\]

where:
\{CNOT, T\} circuits

Recall \((x, y \in \mathbb{F}_2)\):

\[
\text{CNOT} : \left| xy \right\rangle \mapsto \left| x(x \oplus y) \right\rangle
\]
\[
T : \left| x \right\rangle \mapsto \omega^x \left| x \right\rangle, \quad \omega = e^{i\pi/4}
\]

Proposition

A unitary \(U\) can be implemented over CNOT and T gates if and only if

\[
U : \left| x \right\rangle \mapsto \omega^{P(x)} \left| f(x) \right\rangle
\]

where:

1. \(P(x) = \sum_{y \in \mathbb{F}_2^n \setminus \{0\}} a_y (x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n), \quad a_y \in \mathbb{Z}\)
\{\text{CNOT, } T\} \text{ circuits}

Recall \((x, y \in \mathbb{F}_2)\):

- \(\text{CNOT} : |xy\rangle \mapsto |x(x \oplus y)\rangle\)
- \(T : |x\rangle \mapsto \omega^x|x\rangle, \quad \omega = e^{i\pi/4}\)

**Proposition**

A unitary \(U\) can be implemented over CNOT and \(T\) gates if and only if

\[
U : |x\rangle \mapsto \omega^{P(x)}|f(x)\rangle
\]

where:

1. \(P(x) = \sum_{y \in \mathbb{F}_2 \setminus \{0\}} a_y (x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n), \quad a_y \in \mathbb{Z}\)
2. \(f\) is linear (= implementable with just CNOT gates)
\{CNOT, T\} circuits

Recall \((x, y \in \mathbb{F}_2)\):

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

\[
T : |x\rangle \mapsto \omega^x|x\rangle, \quad \omega = e^{i\pi/4}
\]

Proposition

A unitary \(U\) can be implemented over CNOT and T gates if and only if

\[
U : |x\rangle \mapsto \omega^{P(x)}|f(x)\rangle
\]

where:

1. \(P(x) = \sum_{y \in \mathbb{F}_2 \setminus \{0\}} a_y (x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n), \quad a_y \in \mathbb{Z}\)

2. \(f\) is linear (= implementable with just CNOT gates)

Notation: \(P_{a}(x)\) denotes the (unique) “polynomial” with coefficients \(a \in \mathbb{Z}^{2^n-1}_8\)
Computing the phase polynomial

Consider the 1-bit full adder:
Computing the phase polynomial

Consider the 1-bit full adder:

\[ \left| x_1 x_2 x_3 x_4 \right> \]
Computing the phase polynomial

Consider the 1-bit full adder:

$$\omega^{x_1} |x_1 x_2 x_3 x_4\rangle$$
Computing the phase polynomial

Consider the 1-bit full adder:

\[ \omega^{x_1 + x_2} |x_1 x_2 x_3 x_4 \rangle \]
Computing the phase polynomial

Consider the 1-bit full adder:

\[ \omega^{x_1 + x_2 + x_4} |x_1 x_2 x_3 x_4\rangle \]
Computing the phase polynomial

Consider the 1-bit full adder:

\[
\omega^{x_1+x_2+x_4}|(x_1 \oplus x_2)x_2x_3x_4\rangle
\]
Computing the phase polynomial

Consider the 1-bit full adder:

\[ \omega^{x_1 + x_2 + x_4} \left| (x_1 \oplus x_2)(x_2 \oplus x_4)x_3x_4 \right. \]
Computing the phase polynomial

Consider the 1-bit full adder:

$$\omega^{x_1 + x_2 + x_4} \left( (x_1 \oplus x_2)(x_2 \oplus x_4)x_3(x_1 \oplus x_2 \oplus x_4) \right)$$
Computing the phase polynomial

Consider the 1-bit full adder:

\[ \omega^{x_1 + x_2 + x_4 + 7(x_1 \oplus x_2)} |(x_1 \oplus x_2)(x_2 \oplus x_4)x_3(x_1 \oplus x_2 \oplus x_4) \rangle \]
Computing the phase polynomial

Consider the 1-bit full adder:

\[ T \cdot T^\dagger \cdot \ldots \cdot T \cdot T^\dagger \cdot \ldots \cdot T \cdot T^\dagger \cdot T \]

\[ \omega^{x_1 + x_2 + x_3 + 7(x_1 \oplus x_2 \oplus x_3) + 2x_4 + 7(x_1 \oplus x_4) + 7(x_2 \oplus x_4) + 7(x_3 \oplus x_4) + (x_1 \oplus x_2 \oplus x_3 \oplus x_4)} \]

\[ |x_1(x_1 \oplus x_2)(x_1 \oplus x_2 \oplus x_3)x_4\rangle \]
Computing the phase polynomial

Consider the 1-bit full adder:

\[ \omega^{x_1 + x_2 + x_3 + 7(x_1 \oplus x_2 \oplus x_3) + 2x_4 + 7(x_1 \oplus x_4) + 7(x_2 \oplus x_4) + 7(x_3 \oplus x_4) + (x_1 \oplus x_2 \oplus x_3 \oplus x_4)} \]

\[ |x_1(x_1 \oplus x_2)(x_1 \oplus x_2 \oplus x_3)x_4\rangle \]

\[ a = (1, 1, 0, 1, 0, 0, 7, 2, 7, 0, 7, 0, 0, 0, 1) \]
Synthesis

Given $a \in \mathbb{Z}_8^{2^n-1}$, we can synthesize $|x\rangle \mapsto \omega^{Pa(x)} |x\rangle$ as follows:

For each non-zero component $a_y$ of $a$,

1. Compute $x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n$ (O(n) CNOT gates)
2. Apply $T_a y$
3. Uncompute $x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n$

Alternatively, use the $T$-par algorithm (arXiv:1303.2042)...

Recall: $T_2 := P$, $T_4 := Z$, so total $T$-count is $\sum_{y \in \mathbb{F}_2^n} \{0\} (a_y \mod 2)$

Notation

$\mathbb{Z}_8^{2^n-1}$ is the component-wise binary residue of $a \in \mathbb{Z}_8^{2^n-1}$

$\text{wt}(x)$ is the hamming weight of $x \in \mathbb{F}_2^n$

Total $T$-count is $\text{wt}(\text{Res}_2(a))$
Synthesis

Given $a \in \mathbb{Z}_8^{2^n-1}$, we can synthesize $|x\rangle \mapsto \omega^{Pa(x)}|x\rangle$ as follows:
For each non-zero component $a_y$ of $a$,

1. Compute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$ (O(n) CNOT gates)
Synthesis

Given $a \in \mathbb{Z}_8^{2^n-1}$, we can synthesize $|x\rangle \mapsto \omega^{Pa(x)}|x\rangle$ as follows:

For each non-zero component $a_y$ of $a$,

1. Compute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$ ($O(n)$ CNOT gates)
2. Apply $T^{a_y}$
Synthesis

Given $a \in \mathbb{Z}_8^{2^n-1}$, we can synthesize $|x\rangle \mapsto \omega^{Pa(x)}|x\rangle$ as follows:

For each non-zero component $a_y$ of $a$,

1. Compute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$ ($O(n)$ CNOT gates)
2. Apply $T^{a_y}$
3. Uncompute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$
Synthesis

Given $a \in \mathbb{Z}_8^{2^n-1}$, we can synthesize $|x\rangle \mapsto \omega^{Pa(x)}|x\rangle$ as follows:

For each non-zero component $a_y$ of $a$,

1. Compute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$ (O(n) CNOT gates)
2. Apply $T^{a_y}$
3. Uncompute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$

Alternatively, use the $T$-par algorithm (arXiv:1303.2042)....
Synthesis

Given $a \in \mathbb{Z}_8^{2^n-1}$, we can synthesize $|x\rangle \mapsto \omega^{Pa(x)}|x\rangle$ as follows:

For each non-zero component $a_y$ of $a$,

1. Compute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$ ($O(n)$ CNOT gates)
2. Apply $T^{ay}$
3. Uncompute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$

Alternatively, use the $T$-par algorithm (arXiv:1303.2042)...

Recall: $T^2 := P$, $T^4 := Z$, so total $T$ count is

$$\sum_{y \in \mathbb{F}_2^n \setminus \{0\}} (a_y \mod 2)$$
Synthesis

Given $a \in \mathbb{Z}_8^{2^n-1}$, we can synthesize $|x\rangle \mapsto \omega^{Pa(x)}|x\rangle$ as follows:

For each non-zero component $a_y$ of $a$,

1. Compute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$ (O(n) CNOT gates)
2. Apply $T^{a_y}$
3. Uncompute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$

Alternatively, use the $T$-par algorithm (arXiv:1303.2042)...

Recall: $T^2 := P$, $T^4 := Z$, so total $T$ count is

$$\sum_{y \in \mathbb{F}_2^n \setminus \{0\}} (a_y \mod 2)$$

Notation

- $\text{Res}_2(a)$ is the component-wise binary residue of $a \in \mathbb{Z}_8^n$
- $\text{wt}(x)$ is the hamming weight of $x \in \mathbb{F}_2^n$

Total $T$-count is $\text{wt}(\text{Res}_2(a))$
Can we do better?

Observe for $n = 2$:

\[
P(x, y) = 4x + 4y + 4(x \oplus y)
\]

\[
= 0 \quad \text{mod } 8 \quad \forall x, y \in \mathbb{F}_2
\]

\[
\implies |xy\rangle \mapsto \omega^{P(x,y)}|xy\rangle \text{ is the identity operator}
\]
Can we do better?

Observe for $n = 2$:

$$P(x, y) = 4x + 4y + 4(x \oplus y)$$

$$= 0 \mod 8 \quad \forall x, y \in \mathbb{F}_2$$

$$\implies |xy\rangle \mapsto \omega^{P(x,y)}|xy\rangle$$

is the identity operator

More generally, \( \omega^{P_a(x)} = \omega^{P_b(x)} \) for all \( x \) if and only if

$$P_a(x) - P_b(x) = P_{a-b}(x) = 0 \mod 8 \quad \forall x \in \mathbb{F}_2^n$$
Can we do better?

Observe for $n = 2$:

$$P(x, y) = 4x + 4y + 4(x \oplus y)$$

$$= 0 \mod 8 \quad \forall x, y \in \mathbb{F}_2$$

$$\implies |xy\rangle \mapsto \omega^{P(x,y)} |xy\rangle$$ is the identity operator

More generally, $\omega^{P_a(x)} = \omega^{P_b(x)}$ for all $x$ if and only if

$$P_a(x) - P_b(x) = P_{a-b}(x) = 0 \mod 8 \quad \forall x \in \mathbb{F}_2^n$$

Alternatively, the class of tuples giving phase polynomials equivalent to $P_a$ is $a + C_n$, where

$$C_n = \{ c \in \mathbb{Z}_8^{2^n-1} | P_c(x) = 0 \mod 8 \quad \forall x \in \mathbb{F}_2^n \}$$
Can we do better?

Observe for $n = 2$:

$$P(x, y) = 4x + 4y + 4(x \oplus y)$$

$$= 0 \pmod{8} \quad \forall x, y \in \mathbb{F}_2$$

$$\implies |xy\rangle \mapsto \omega^{P(x, y)}|xy\rangle$$ is the identity operator

More generally, $\omega^{P_a(x)} = \omega^{P_b(x)}$ for all $x$ if and only if

$$P_a(x) - P_b(x) = P_{a-b}(x) = 0 \pmod{8} \quad \forall x \in \mathbb{F}_2^n$$

Alternatively, the class of tuples giving phase polynomials equivalent to $P_a$ is $a + C_n$, where

$$C_n = \{ c \in \mathbb{Z}_8^{2^n-1} | P_c(x) = 0 \pmod{8} \quad \forall x \in \mathbb{F}_2^n \}$$

Proposition

There exists an implementation of $|x\rangle \mapsto \omega^{P_a(x)}|x\rangle$ over \{ CNOT, T \} with $T$-count $k$ if and only if there exists $c \in C_n$ s.t.

$$wt(Res_2(a + c)) = wt(Res_2(a) \oplus Res_2(c)) = k$$
Coding theory

Definition (Binary linear code)
A binary linear code of length $n$ is a subgroup $C < \mathbb{F}_2^n$

Example: $\text{Res}_2(C_n) < \mathbb{F}_2^{2^n-1}$ is a binary linear code
Coding theory

Definition (Binary linear code)
A binary linear code of length $n$ is a subgroup $C < \mathbb{F}_2^n$
Example: $\text{Res}_2(C_n) < \mathbb{F}_2^{2^n-1}$ is a binary linear code

Definition (Minimum distance decoding)
The minimum distance decoding problem for a binary linear code of length $n$ in $C$ is to find, given a vector $\mathbf{x} \in \mathbb{F}_2^n$, some $\mathbf{y} \in C$ such that for all $\mathbf{z} \in C$,

$$\text{wt}(\mathbf{x} \oplus \mathbf{y}) \leq \text{wt}(\mathbf{x} \oplus \mathbf{z})$$
Coding theory

**Definition (Binary linear code)**
A binary linear code of length $n$ is a subgroup $C < \mathbb{F}_2^n$

Example: $\text{Res}_2(C_n) < \mathbb{F}_2^{2^n-1}$ is a binary linear code

**Definition (Minimum distance decoding)**
The *minimum distance decoding* problem for a binary linear code of length $n$ in $C$ is to find, given a vector $x \in \mathbb{F}_2^n$, some $y \in C$ such that for all $z \in C$,

$$\text{wt}(x \oplus y) \leq \text{wt}(x \oplus z)$$

$\implies$ Optimizing the $T$-count for $P_a$ is equivalent to minimally decoding $\text{Res}_2(a)$ in $\text{Res}_2(C_n)$!
Reed-Muller codes

Given $f \in \mathbb{F}_2[x_1, x_2, \ldots, x_n]$, the evaluation vector of $f$ is

$$ f = (f(1, 0, \ldots, 0), f(0, 1, \ldots, 0), \ldots, f(1, 1, \ldots, 1)) $$

Note: the total degree of a monomial $x_{i_1}x_{i_2}\cdots x_{i_k}$ is $k$
Reed-Muller codes

Given \( f \in \mathbb{F}_2[x_1, x_2, \ldots, x_n] \), the evaluation vector of \( f \) is

\[
\mathbf{f} = (f(1, 0, \ldots, 0), f(0, 1, \ldots, 0), \ldots, f(1, 1, \ldots, 1))
\]

Note: the total degree of a monomial \( x_{i_1}x_{i_2}\cdots x_{i_k} \) is \( k \)

Definition (Punctured Reed-Muller code)

\[
\mathcal{RM}(r, n)^* = \{ \mathbf{f} \mid f \in \mathbb{F}_2[x_1, x_2, \ldots, x_n], \deg(f) \leq r \}
\]
Main theorem

Theorem

$$Res_2(C_n) = \mathcal{R}\mathcal{M}(n - 4, n)^*$$
Applications

Upper bounds

Covering radius of a code $C$:

$$
\rho(C) = \max_{x \in \mathbb{F}_2^n} \min_{y \in C} \text{wt}(x \oplus y)
$$
Applications
Upper bounds

Covering radius of a code $C$:

$$\rho(C) = \max_{x \in \mathbb{F}_2^n} \min_{y \in C} \text{wt}(x \oplus y)$$

Theorem (Cohen & Litsyn ’92)
For large $n$ and orders $r$ where $n - r \geq 3$,

$$\rho(\mathcal{RM}(r, n)) \leq \frac{n^{n-r-2}}{(n-r-2)!}.$$
Applications
Upper bounds

Covering radius of a code $C$:

$$\rho(C) = \max_{x \in \mathbb{F}_2^n} \min_{y \in C} \text{wt}(x \oplus y)$$

Theorem (Cohen & Litsyn ‘92)
For large $n$ and orders $r$ where $n - r \geq 3$,

$$\rho(\mathcal{RM}(r, n)) \leq \frac{n^{n-r-2}}{(n-r-2)!}.$$ 

Corollary
Any $n$-qubit unitary implementable over $\{\text{CNOT}, T\}$ can be synthesized with $O(n^2)$ $T$ gates.
Applications
Optimization

Algorithm: Given $n$-qubit circuit over \{CNOT, $T$\},
Applications
Optimization

**Algorithm:** Given $n$-qubit circuit over $\{\text{CNOT}, T\}$,

1. Compute phase coefficients $a \in \mathbb{Z}_8^{2^n-1}$
Applications
Optimization

Algorithm: Given $n$-qubit circuit over \{CNOT, $T$\},
1. Compute phase coefficients $a \in \mathbb{Z}_8^{2^n-1}$
2. Decode binary residue of $a$ in $RM(n-4, n)^*$ as $w$
Applications
Optimization

**Algorithm:** Given $n$-qubit circuit over \{CNOT, $T$\},

1. Compute phase coefficients $a \in \mathbb{Z}_8^{2^n-1}$
2. Decode binary residue of $a$ in $\mathcal{RM}(n-4, n)^*$ as $w$
3. Find some $c \in C_n$ with binary residue equal to $w$
Applications
Optimization

**Algorithm:** Given $n$-qubit circuit over \{CNOT, $T$\},

1. Compute phase coefficients $a \in \mathbb{Z}_8^{2^n-1}$
2. Decode binary residue of $a$ in $RM(n - 4, n)^*$ as $w$
3. Find some $c \in C_n$ with binary residue equal to $w$
4. Synthesize circuit with coefficients $a + c$
Applications

Optimization

Algorithm: Given $n$-qubit circuit over $\{\text{CNOT}, T\}$,

1. Compute phase coefficients $a \in \mathbb{Z}_8^{2^n-1}$
2. Decode binary residue of $a$ in $\mathcal{RM}(n - 4, n)^*$ as $w$
3. Find some $c \in C_n$ with binary residue equal to $w$
4. Synthesize circuit with coefficients $a + c$

Problem: how do we find $c$?
Applications
A closer look at step 3

Given $f \in \mathbb{F}_2[x_1, x_2, \ldots, x_n]$, denote by $\overline{f} \in \mathbb{Z}_8[x_1, x_2, \ldots, x_n]$ the polynomial obtained by replacing addition and multiplication mod 2 with mod 8.
Given \( f \in \mathbb{F}_2[x_1, x_2, \ldots, x_n] \), denote by \( \overline{f} \in \mathbb{Z}_8[x_1, x_2, \ldots, x_n] \) the polynomial obtained by replacing addition and multiplication mod 2 with mod 8.

**Example**

Suppose \( f = x_1x_2 \oplus x_1x_3 \oplus x_5 \)

Then \( \overline{f} = x_1x_2 + x_1x_3 + x_5 \mod 8 \)

\( \overline{f} \) denotes the tuple of (non-trivial) binary evaluations of \( \overline{f} \)
Applications
A closer look at step 3

Given $f \in \mathbb{F}_2[x_1, x_2, \ldots, x_n]$, denote by $\overline{f} \in \mathbb{Z}_8[x_1, x_2, \ldots, x_n]$ the polynomial obtained by replacing addition and multiplication mod 2 with mod 8.

Example
Suppose $f = x_1x_2 \oplus x_1x_3 \oplus x_5$
Then $\overline{f} = x_1x_2 + x_1x_3 + x_5 \mod 8$
$\overline{f}$ denotes the tuple of (non-trivial) binary evaluations of $\overline{f}$

Lemma
For all $f \in \mathbb{F}_2[x_1, x_2, \ldots, x_n]$, if $f \in \mathcal{RM}(n - 4, n)^*$ then $\overline{f} \in \mathcal{C}_n$
Applications

Optimizing the adder

Recall the full 1 bit adder:

$$|x\rangle \mapsto \omega^{P_a(x)}|x_1(x_1 \oplus x_2)(x_1 \oplus x_2 \oplus x_3)x_4\rangle$$

$$a = (1, 1, 0, 1, 0, 0, 7, 2, 7, 7, 0, 7, 0, 0, 1)$$

Want to decode $\text{Res}_2(a) = (1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1)$ in $\text{Res}_2(\mathcal{C}_n) = \mathcal{RM}(0, 4)^*$ with minimum distance
Applications
Optimizing an adder

\[ \text{Res}_2(a) = (1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1) \]

We know

\[ \mathcal{RM}(0, 4)^* = \left\{ (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) \right\} \]

So minimum distance (7) decoding of \( \text{Res}_2(a) \) is the all-one vector
Applications

Optimizing an adder

\[ \text{Res}_2(a) = (1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1) \]

We know

\[ \mathcal{RM}(0, 4)^* = \left\{ (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) \right\} \]

So minimum distance (7) decoding of \( \text{Res}_2(a) \) is the all-one vector

Now \( f = 1 \) (the constant polynomial), so

\[ c = \overline{f} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) \in C_n \]
Applications
Optimizing an adder

\[ \text{Res}_2(a) = (1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1) \]

We know

\[ \mathcal{RM}(0, 4)^* = \left\{ (0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \right\} \]

So minimum distance (7) decoding of \( \text{Res}_2(a) \) is the all-one vector

Now \( f = 1 \) (the constant polynomial), so

\[ c = \bar{f} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) \in C_n \]

Finally synthesize with phase coefficients

\[ a + c = (2, 2, 1, 2, 1, 1, 0, 3, 0, 0, 1, 0, 1, 1, 2) \]
Applications
Optimizing an adder

Resulting circuit:

Previously: $T$-count 8, $T$-depth 2
Now: $T$-count 7, $T$-depth 3
Applications

Complexity

\[
\text{MDD}(\mathcal{RM}(n - 4, n)^*) - \\
\text{Minimum distance decoding in } \mathcal{RM}(n - 4, n)^*
\]

\[
\text{T-MIN}(n, \{\text{CNOT, T}\}) - \\
\text{T-count minimization over } n\text{-qubit } \{\text{CNOT, T}\} \text{ circuits}
\]
Applications
Complexity

\[
\text{MDD}(\mathcal{RM}(n-4, n)^*) - \\
\text{Minimum distance decoding in } \mathcal{RM}(n-4, n)^* \\
T\text{-MIN}(n, \{\text{CNOT, T}\}) - \\
T\text{-count minimization over } n\text{-qubit } \{\text{CNOT, T}\} \text{ circuits}
\]

Theorem
\[
\text{MDD}(\mathcal{RM}(n, n-4)^*) \leq_P T\text{-MIN}(n, \{\text{CNOT, T}\})
\]
## Benchmarks (excerpt)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>n</th>
<th>T-count</th>
<th>Original</th>
<th>T-par</th>
<th>Majority</th>
<th>Recursive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grover&lt;sub&gt;5&lt;/sub&gt;</td>
<td>9</td>
<td>140</td>
<td>52</td>
<td>52</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>Mod&lt;sub&gt;5&lt;/sub&gt;</td>
<td>5</td>
<td>28</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>VBE-Adder&lt;sub&gt;3&lt;/sub&gt;</td>
<td>10</td>
<td>70</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>CSLA-MUX&lt;sub&gt;3&lt;/sub&gt;</td>
<td>15</td>
<td>70</td>
<td>62</td>
<td>62</td>
<td></td>
<td>58</td>
</tr>
<tr>
<td>CSUM-MUX&lt;sub&gt;9&lt;/sub&gt;</td>
<td>30</td>
<td>196</td>
<td>140</td>
<td>84</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>QCLA-Com&lt;sub&gt;7&lt;/sub&gt;</td>
<td>24</td>
<td>203</td>
<td>95</td>
<td>94</td>
<td>153</td>
<td></td>
</tr>
<tr>
<td>QCLA-Mod&lt;sub&gt;7&lt;/sub&gt;</td>
<td>26</td>
<td>413</td>
<td>249</td>
<td>238</td>
<td>299</td>
<td></td>
</tr>
<tr>
<td>Adder&lt;sub&gt;8&lt;/sub&gt;</td>
<td>24</td>
<td>399</td>
<td>215</td>
<td>213</td>
<td>249</td>
<td></td>
</tr>
<tr>
<td>RC-Adder&lt;sub&gt;6&lt;/sub&gt;</td>
<td>14</td>
<td>77</td>
<td>63</td>
<td>47</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>Mod-Red&lt;sub&gt;21&lt;/sub&gt;</td>
<td>11</td>
<td>119</td>
<td>73</td>
<td>73</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>Mod-Mult&lt;sub&gt;55&lt;/sub&gt;</td>
<td>9</td>
<td>49</td>
<td>37</td>
<td>35</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>Mod-Adder&lt;sub&gt;1024&lt;/sub&gt;</td>
<td>28</td>
<td>1995</td>
<td>1011</td>
<td>1011</td>
<td>1011</td>
<td></td>
</tr>
<tr>
<td>BCSD&lt;sub&gt;2&lt;/sub&gt;</td>
<td>9</td>
<td>14</td>
<td>14</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BCSD&lt;sub&gt;4&lt;/sub&gt;</td>
<td>14</td>
<td>20</td>
<td>20</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>BCSD&lt;sub&gt;8&lt;/sub&gt;</td>
<td>21</td>
<td>32</td>
<td>32</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Cycle 17.3</td>
<td>35</td>
<td>4739</td>
<td>1945</td>
<td>1944</td>
<td>1982</td>
<td></td>
</tr>
<tr>
<td>GF(2&lt;sup&gt;4&lt;/sup&gt;)-Mult</td>
<td>12</td>
<td>112</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>GF(2&lt;sup&gt;5&lt;/sup&gt;)-Mult</td>
<td>15</td>
<td>175</td>
<td>111</td>
<td>111</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>GF(2&lt;sup&gt;6&lt;/sup&gt;)-Mult</td>
<td>18</td>
<td>252</td>
<td>150</td>
<td>150</td>
<td>144</td>
<td></td>
</tr>
<tr>
<td>GF(2&lt;sup&gt;7&lt;/sup&gt;)-Mult</td>
<td>21</td>
<td>343</td>
<td>217</td>
<td>217</td>
<td>208</td>
<td></td>
</tr>
<tr>
<td>GF(2&lt;sup&gt;8&lt;/sup&gt;)-Mult</td>
<td>24</td>
<td>448</td>
<td>264</td>
<td>264</td>
<td>237</td>
<td></td>
</tr>
<tr>
<td>HWB&lt;sub&gt;6&lt;/sub&gt;</td>
<td>7</td>
<td>105</td>
<td>71</td>
<td>75</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>HWB&lt;sub&gt;8&lt;/sub&gt;</td>
<td>12</td>
<td>5887</td>
<td>3551</td>
<td>3531</td>
<td>3531</td>
<td></td>
</tr>
<tr>
<td>nth-prime&lt;sub&gt;6&lt;/sub&gt;</td>
<td>9</td>
<td>812</td>
<td>402</td>
<td>400</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td>nth-prime&lt;sub&gt;8&lt;/sub&gt;</td>
<td>12</td>
<td>6671</td>
<td>4047</td>
<td>4045</td>
<td>4045</td>
<td></td>
</tr>
</tbody>
</table>
Generalizations

Given a primitive rotation gate $R(2\pi/2^k)$, define the set of zero-everywhere phase functions as

$$C_n^k = \{ c \in \mathbb{Z}^{2^n-1} | P_c(x) = 0 \mod 2^k \quad \forall x \in \mathbb{F}_2^n \}$$

Theorem

$$\text{Res}_2(C_n^k) = \mathcal{RM}(n - k - 1, n)^*$$
Generalizations

Given a primitive rotation gate $R(2\pi/2^k)$, define the set of zero-everywhere phase functions as

$$C_n^k = \{ c \in \mathbb{Z}^{2^{n-1}} \mid P_c(x) = 0 \mod 2^k \quad \forall x \in \mathbb{F}_2^n \}$$

Theorem

$$Res_2(C_n^k) = \mathcal{RM}(n - k - 1, n)^*$$

Going further, we have a characterization of the zero-everywhere functions for any composite denominator $R(\pi/q)$.
Conclusion

Exact minimization of $T$-count over $\{\text{CNOT}, T\}$ – **Done!**
Conclusion

Exact minimization of $T$-count over \{CNOT, $T$\} – **Done!**

- $T$-count optimization algorithm using any $\mathcal{RM}$ decoder

Future work

- Optimization of all phase gates
  
  - Heuristic – optimize $R\left(\pi/2^k\right)$ gates in order of decreasing $k$.

  - Preferable – decode directly over $C_k^n$...

- Optimizing \{CNOT, $T$, H\} circuits

  - About 75% done

  - Requires partial decoding – decoding with some bits known
Conclusion

Exact minimization of $T$-count over \{CNOT, $T$\} – **Done!**

- $T$-count optimization algorithm using any $RM$ decoder
- Upper bound of $O(n^2)$ $T$-gates per \{CNOT, $T$\} circuit

Future work

- Optimization of all phase gates
- Heuristic – optimize $R(k \pi/2)$ gates in order of decreasing $k$
- Preferable – decode directly over $C_k n$
- Optimizing \{CNOT, $T$, $H$\} circuits
- About 75% done
- Requires partial decoding – decoding with some bits known
Conclusion

Exact minimization of $T$-count over $\{\text{CNOT}, T\}$ – **Done!**

- $T$-count optimization algorithm using any $RM$ decoder
- Upper bound of $O(n^2)$ $T$-gates per $\{\text{CNOT}, T\}$ circuit
- multi-qubit $T$-count optimization is **really hard**
Conclusion

Exact minimization of $T$-count over $\{\text{CNOT, } T\}$ – **Done!**
- $T$-count optimization algorithm using any $RM$ decoder
- Upper bound of $O(n^2)$ $T$-gates per $\{\text{CNOT, } T\}$ circuit
- multi-qubit $T$-count optimization is **really hard**

Future work
- Optimization of all phase gates
  - Heuristic – optimize $R(\pi/2^k)$ gates in order of decreasing $k$.
  - Preferable – decode directly over $C_n^k$...

Conclusion

Exact minimization of $T$-count over $\{\text{CNOT, } T\}$ – **Done!**
- $T$-count optimization algorithm using any $RM$ decoder
- Upper bound of $O(n^2)$ $T$-gates per $\{\text{CNOT, } T\}$ circuit
- multi-qubit $T$-count optimization is **really hard**

Future work
- Optimization of all phase gates
  - Heuristic – optimize $R(\pi/2^k)$ gates in order of decreasing $k$.
  - Preferable – decode directly over $C^k_n$...
- Optimizing $\{\text{CNOT, } T, H\}$ circuits
  - About 75% done
  - Requires *partial* decoding – decoding with some bits known
Thank you!