T-count optimization and Reed-Muller codes

Matthew Amy

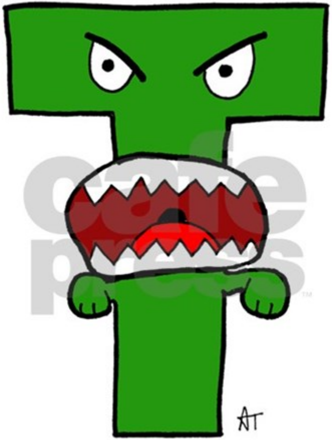
Joint work with Michele Mosca
Institute for Quantum Computing, University of Waterloo

arXiv:1601.07363 [quant-ph]

BIRS Quantum Computer Science Workshop, Banff
April 22 2016

Why optimize T count?

Why optimize T count?



$\{\text{CNOT}, T\}$ circuits

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

$$\begin{aligned} \text{CNOT} : |xy\rangle &\mapsto |x(x \oplus y)\rangle \\ T : |x\rangle &\mapsto \omega^x |x\rangle, \qquad \omega = e^{i\pi/4} \end{aligned}$$

$\{\text{CNOT}, T\}$ circuits

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

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

Proposition

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

$$U : |\mathbf{x}\rangle \mapsto \omega^{P(\mathbf{x})} |f(\mathbf{x})\rangle$$

where:

$\{\text{CNOT}, T\}$ circuits

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

$$\begin{aligned} \text{CNOT} : |xy\rangle &\mapsto |x(x \oplus y)\rangle \\ T : |x\rangle &\mapsto \omega^x |x\rangle, \qquad \omega = e^{i\pi/4} \end{aligned}$$

Proposition

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

$$U : |\mathbf{x}\rangle \mapsto \omega^{P(\mathbf{x})} |f(\mathbf{x})\rangle$$

where:

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

$\{\text{CNOT}, T\}$ circuits

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

$$\begin{aligned} \text{CNOT} : |xy\rangle &\mapsto |x(x \oplus y)\rangle \\ T : |x\rangle &\mapsto \omega^x |x\rangle, \qquad \omega = e^{i\pi/4} \end{aligned}$$

Proposition

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

$$U : |\mathbf{x}\rangle \mapsto \omega^{P(\mathbf{x})} |f(\mathbf{x})\rangle$$

where:

1. $P(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}} a_{\mathbf{y}} (x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n), \quad a_{\mathbf{y}} \in \mathbb{Z}$
2. *f is linear (= implementable with just CNOT gates)*

$\{\text{CNOT}, T\}$ circuits

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

$$\begin{aligned} \text{CNOT} : |xy\rangle &\mapsto |x(x \oplus y)\rangle \\ T : |x\rangle &\mapsto \omega^x |x\rangle, \qquad \omega = e^{i\pi/4} \end{aligned}$$

Proposition

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

$$U : |\mathbf{x}\rangle \mapsto \omega^{P(\mathbf{x})} |f(\mathbf{x})\rangle$$

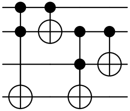
where:

1. $P(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbb{F}_2^n \setminus \{\mathbf{0}\}} a_{\mathbf{y}} (x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n), \quad a_{\mathbf{y}} \in \mathbb{Z}$
2. *f is linear (= implementable with just CNOT gates)*

Notation: $P_{\mathbf{a}}(\mathbf{x})$ denotes the (unique) "polynomial" with coefficients $\mathbf{a} \in \mathbb{Z}_8^{2^n - 1}$

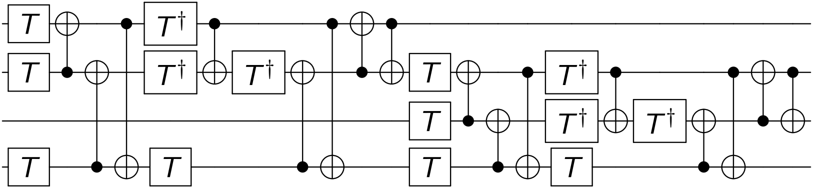
Computing the phase polynomial

Consider the 1-bit full adder:



Computing the phase polynomial

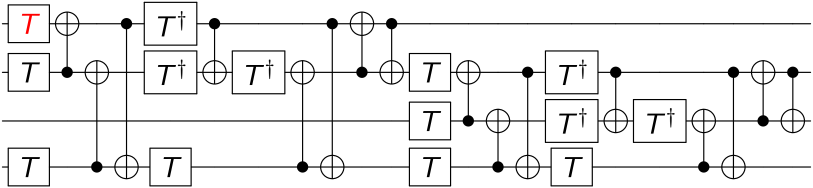
Consider the 1-bit full adder:



$$|x_1x_2x_3x_4\rangle$$

Computing the phase polynomial

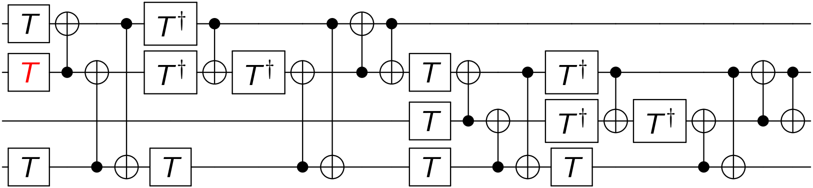
Consider the 1-bit full adder:



$$\omega^{x_1} \left| x_1 x_2 x_3 x_4 \right\rangle$$

Computing the phase polynomial

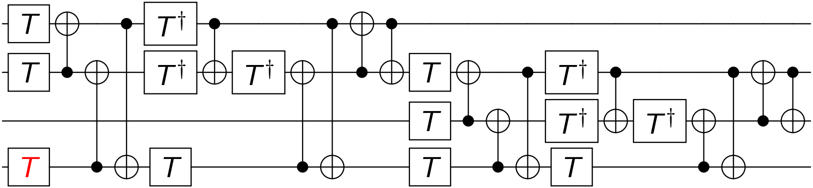
Consider the 1-bit full adder:



$$\omega^{x_1 + x_2} |x_1 x_2 x_3 x_4\rangle$$

Computing the phase polynomial

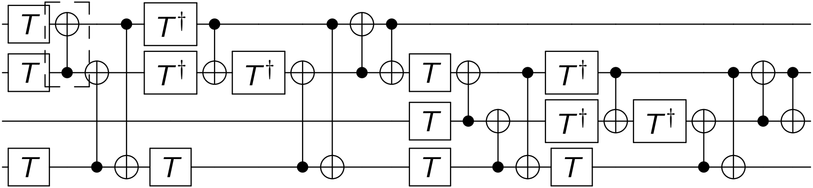
Consider the 1-bit full adder:



$$\omega^{x_1+x_2+x_4}|x_1x_2x_3x_4\rangle$$

Computing the phase polynomial

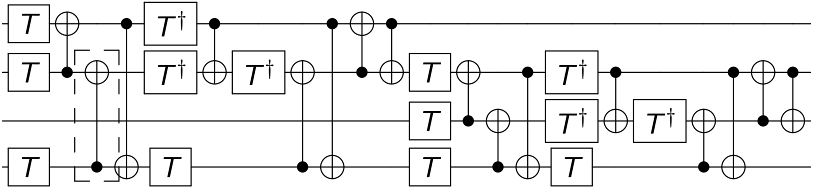
Consider the 1-bit full adder:



$$\omega^{x_1+x_2+x_4}|(x_1 \oplus x_2)x_2x_3x_4\rangle$$

Computing the phase polynomial

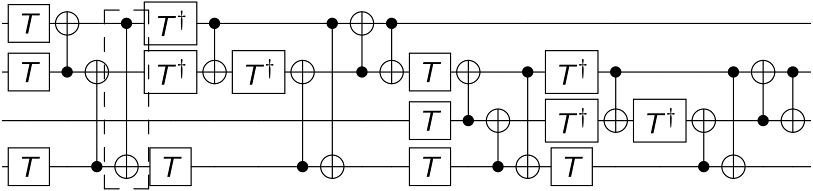
Consider the 1-bit full adder:



$$\omega^{x_1+x_2+x_4}|(x_1 \oplus x_2)(x_2 \oplus x_4)x_3x_4\rangle$$

Computing the phase polynomial

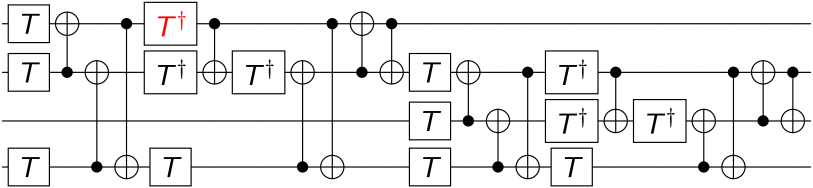
Consider the 1-bit full adder:



$$\omega^{x_1+x_2+x_4}|(x_1 \oplus x_2)(x_2 \oplus x_4)x_3(x_1 \oplus x_2 \oplus x_4)\rangle$$

Computing the phase polynomial

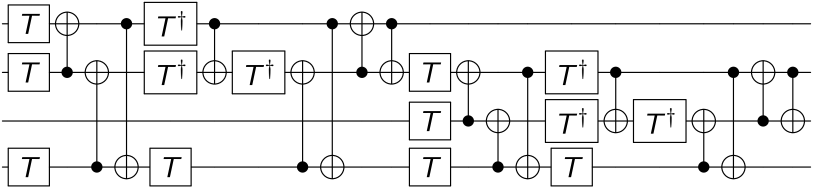
Consider the 1-bit full adder:



$$\omega^{x_1+x_2+x_4+7(x_1\oplus x_2)}|(x_1 \oplus x_2)(x_2 \oplus x_4)x_3(x_1 \oplus x_2 \oplus x_4)\rangle$$

Computing the phase polynomial

Consider the 1-bit full adder:

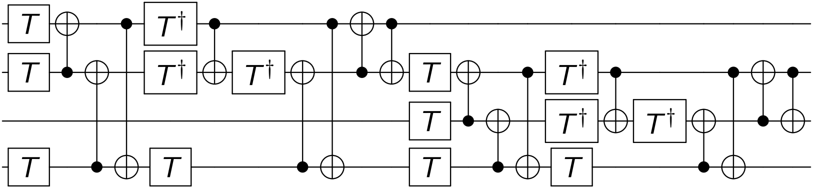


$$\omega^{x_1+x_2+x_3+7(x_1\oplus x_2\oplus x_3)+2x_4+7(x_1\oplus x_4)+7(x_2\oplus x_4)+7(x_3\oplus x_4)+(x_1\oplus x_2\oplus x_3\oplus x_4)}$$

$$|x_1(x_1 \oplus x_2)(x_1 \oplus x_2 \oplus x_3)x_4\rangle$$

Computing the phase polynomial

Consider the 1-bit full adder:



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

$$|x_1(x_1 \oplus x_2)(x_1 \oplus x_2 \oplus x_3)x_4\rangle$$

$$\mathbf{a} = (1, 1, 0, 1, 0, 0, 7, 2, 7, 7, 0, 7, 0, 0, 1)$$

Synthesis

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

Synthesis

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

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

Synthesis

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

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

Synthesis

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

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

Synthesis

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

1. Compute $x_1y_1 \oplus x_2y_2 \oplus \cdots \oplus x_ny_n$ (O(n) CNOT gates)
2. Apply $T^{\mathbf{a_y}}$
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 $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$, we can synthesize $|\mathbf{x}\rangle \mapsto \omega^{P_{\mathbf{a}}(\mathbf{x})}|\mathbf{x}\rangle$ as follows:
For each non-zero component $\mathbf{a}_{\mathbf{y}}$ of $\mathbf{a}$,

1. Compute $x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n$ (O(n) CNOT gates)
2. Apply $T^{\mathbf{a}_{\mathbf{y}}}$
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

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

Synthesis

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

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

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

Notation

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

Total T-count is $\mathrm{wt}(\mathrm{Res}_2(\mathbf{a}))$

Can we do better?

Observe for $n = 2$:

$$\begin{aligned} P(x, y) &= 4x + 4y + 4(x \oplus y) \\ &= 0 \mod 8 \quad \forall x, y \in \mathbb{F}_2 \end{aligned}$$

$\implies |xy\rangle \mapsto \omega^{P(x,y)}|xy\rangle$ 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, $\omega^{P_{\mathbf{a}}(\mathbf{x})} = \omega^{P_{\mathbf{b}}(\mathbf{x})}$ for all $\mathbf{x}$ if and only if

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

Can we do better?

Observe for $n = 2$:

$$\begin{aligned} P(x,y) &= 4x + 4y + 4(x \oplus y) \\ &= 0 \mod 8 \quad \forall x, y \in \mathbb{F}_2 \end{aligned}$$

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

More generally, $\omega^{P_{\mathbf{a}}(\mathbf{x})} = \omega^{P_{\mathbf{b}}(\mathbf{x})}$ for all $\mathbf{x}$ if and only if

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

Alternatively, the class of tuples giving phase polynomials equivalent to $P_{\mathbf{a}}$ is $\mathbf{a} + \mathcal{C}_n$, where

$$\mathcal{C}_n = \{\mathbf{c} \in \mathbb{Z}_8^{2^n-1} \mid P_{\mathbf{c}}(\mathbf{x}) = 0 \mod 8 \quad \forall \mathbf{x} \in \mathbb{F}_2^n\}$$

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, $\omega^{P_{\mathbf{a}}(\mathbf{x})} = \omega^{P_{\mathbf{b}}(\mathbf{x})}$ for all $\mathbf{x}$ if and only if

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

Alternatively, the class of tuples giving phase polynomials equivalent to $P_{\mathbf{a}}$ is $\mathbf{a} + \mathcal{C}_n$, where

$$\mathcal{C}_n = \{\mathbf{c} \in \mathbb{Z}_8^{2^n-1} \mid P_{\mathbf{c}}(\mathbf{x}) = 0 \mod 8 \quad \forall \mathbf{x} \in \mathbb{F}_2^n\}$$

Proposition

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

$$wt(Res_2(\mathbf{a} + \mathbf{c})) = wt(Res_2(\mathbf{a}) \oplus Res_2(\mathbf{c})) = k$$

Coding theory

Definition (Binary linear code)

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

Example: $\mathrm{Res}_2(\mathcal{C}_n) < \mathbb{F}_2^{2^n-1}$ 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: $\mathrm{Res}_2(\mathcal{C}_n) < \mathbb{F}_2^{2^n-1}$ 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 $\mathbf{z} \in C$,

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

Coding theory

Definition (Binary linear code)

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

Example: $\mathrm{Res}_2(\mathcal{C}_n) < \mathbb{F}_2^{2^n-1}$ 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 $\mathbf{z} \in C$,

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

$\Longrightarrow$ Optimizing the T-count for $P_{\mathbf{a}}$ is equivalent to minimally decoding $\mathrm{Res}_2(\mathbf{a})$ in $\mathrm{Res}_2(\mathcal{C}_n)$!

Reed-Muller codes

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

$$\mathbf{f} = (f(1, 0, \ldots, 0),\ f(0, 1, \ldots, 0),\ \cdots\ ,\ f(1, 1, \ldots, 1))$$

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

$$\mathbf{f} = (f(1,0,\ldots,0),\ f(0,1,\ldots,0),\ \cdots\ ,\ f(1,1,\ldots,1))$$

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

$$Res_2(\mathcal{C}_n) = \mathcal{RM}(n-4, n)^*$$

Applications

Upper bounds

Covering radius of a code C:

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

Applications

Upper bounds

Covering radius of a code C:

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

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:

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

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 $\{CNOT, T\}$ can be synthesized with $O(n^2)$ T gates.

Applications

Optimization

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

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

Applications
Optimization

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

1. Compute phase coefficients $\mathbf{a} \in \mathbb{Z}_8^{2^n-1}$
2. Decode binary residue of $\mathbf{a}$ in $\mathcal{RM}(n-4, n)^*$ as $\mathbf{w}$

Applications

Optimization

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

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

Applications
Optimization

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

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

Applications

Optimization

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

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

Problem: how do we find $\mathbf{c}$?

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.

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 $\overline{f} = x_1x_2 + x_1x_3 + x_5 \mod 8$

$\overline{\mathbf{f}}$ 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

$$\begin{array}{ll} \text{Suppose} & f = x_1x_2 \oplus x_1x_3 \oplus x_5 \\ \text{Then} & \overline{f} = x_1x_2 + x_1x_3 + x_5 \mod 8 \end{array}$$

$\overline{\mathbf{f}}$ denotes the tuple of (non-trivial) *binary* evaluations of $\overline{f}$

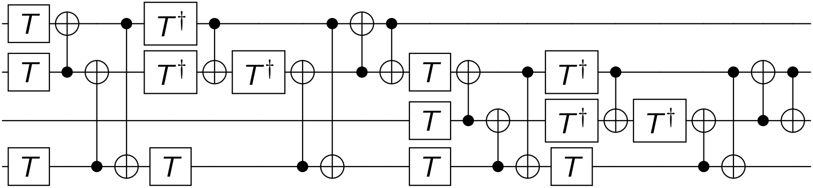
Lemma

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

Applications

Optimizing the adder

Recall the full 1 bit adder:



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

$$\mathbf{a} = (1, 1, 0, 1, 0, 0, 7, 2, 7, 7, 0, 7, 0, 0, 1)$$

Want to decode $\mathrm{Res}_2(\mathbf{a}) = (1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1)$ in $\mathrm{Res}_2(\mathcal{C}_n) = \mathcal{RM}(0, 4)^*$ with minimum distance

Applications

Optimizing an adder

$$\mathrm{Res}_2(\mathbf{a}) = (1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1)$$

We know

$$\mathcal{RM}(0, 4)^* = \begin{Bmatrix} (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \\ (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) \end{Bmatrix}$$

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

Applications

Optimizing an adder

$$\mathrm{Res}_2(\mathbf{a}) = (1,1,0,1,0,0,1,0,1,1,0,1,0,0,1)$$

We know

$$\mathcal{RM}(0,4)^* = \left\{ \begin{matrix} (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\ (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1) \end{matrix} \right\}$$

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

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

$$\mathbf{c} = \bar{\mathbf{f}} = (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1) \in \mathcal{C}_n$$

Applications

Optimizing an adder

$$\mathrm{Res}_2(\mathbf{a}) = (1,1,0,1,0,0,1,0,1,1,0,1,0,0,1)$$

We know

$$\mathcal{RM}(0,4)^* = \begin{Bmatrix} (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\ (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1) \end{Bmatrix}$$

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

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

$$\mathbf{c} = \overline{\mathbf{f}} = (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1) \in \mathcal{C}_n$$

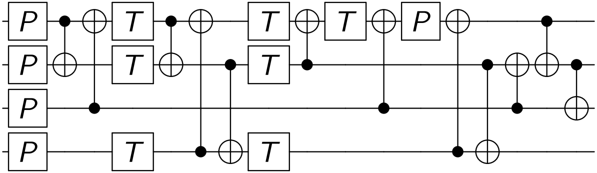
Finally synthesize with phase coefficients

$$\mathbf{a} + \mathbf{c} = (2,2,1,2,1,1,0,3,0,0,1,0,1,1,2)$$

Applications

Optimizing an adder

Resulting circuit:



Previously: T-count 8, T-depth 2

Now: T-count 7, T-depth 3

Applications

Complexity

$\mathrm{MDD}(\mathcal{RM}(n-4, n)^*)$ –

Minimum distance decoding in $\mathcal{RM}(n-4, n)^*$

T-$\textsc{min}(n, \{\mathrm{CNOT}, \mathrm{T}\})$ –

T-count minimization over n-qubit $\{\mathrm{CNOT}, T\}$ circuits

Applications
Complexity

$\mathrm{MDD}(\mathcal{RM}(n-4,n)^*)$ –
Minimum distance decoding in $\mathcal{RM}(n-4,n)^*$
$T\text{-}\textsc{min}(n, \{\mathrm{CNOT}, \mathrm{T}\})$ –
T-count minimization over n-qubit $\{\mathrm{CNOT}, T\}$ circuits

Theorem
$\mathrm{MDD}(\mathcal{RM}(n,n-4)^*) \leq_P T\text{-}\textsc{min}(n, \{\mathrm{CNOT}, \mathrm{T}\})$

Benchmarks (excerpt)

Benchmark	n	T-count			
		Original	T-par	Majority	Recursive
Grover_5	9	140	52	52	52
$\text{Mod } 5_4$	5	28	16	16	16
VBE-Adder_3	10	70	24	24	24
CSLA-MUX_3	15	70	62	62	58
CSUM-MUX_9	30	196	140	84	76
QCLA-Com_7	24	203	95	94	153
QCLA-Mod_7	26	413	249	238	299
Adder_8	24	399	215	213	249
RC-Adder_6	14	77	63	47	47
Mod-Red_{21}	11	119	73	73	73
Mod-Mult_{55}	9	49	37	35	35
Mod-Adder_{1024}	28	1995	1011	1011	1011
BCSD_2	9	14	14	2	2
BCSD_4	14	20	20	4	4
BCSD_8	21	32	32	8	8
Cycle 17_3	35	4739	1945	1944	1982
$\text{GF}(2^4)\text{-Mult}$	12	112	68	68	68
$\text{GF}(2^5)\text{-Mult}$	15	175	111	111	101
$\text{GF}(2^6)\text{-Mult}$	18	252	150	150	144
$\text{GF}(2^7)\text{-Mult}$	21	343	217	217	208
$\text{GF}(2^8)\text{-Mult}$	24	448	264	264	237
HWB_6	7	105	71	75	75
HWB_8	12	5887	3551	3531	3531
$n\text{th-prime}_6$	9	812	402	400	400
$n\text{th-prime}_8$	12	6671	4047	4045	4045

Generalizations

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

$$\mathcal{C}_n^k = \{\mathbf{c} \in \mathbb{Z}^{2^n-1} \mid P_{\mathbf{c}}(\mathbf{x}) = 0 \mod 2^k \quad \forall \mathbf{x} \in \mathbb{F}_2^n\}$$

Theorem

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

Generalizations

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

$$\mathcal{C}_n^k = \{\mathbf{c} \in \mathbb{Z}^{2^n - 1} \mid P_{\mathbf{c}}(\mathbf{x}) = 0 \mod 2^k \quad \forall \mathbf{x} \in \mathbb{F}_2^n\}$$

Theorem

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

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 $\{\text{CNOT}, T\}$ – **Done!**

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

Conclusion

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

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

Conclusion

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

- T-count optimization algorithm using any $\mathcal{RM}$ 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 $\mathcal{RM}$ 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 $\mathcal{C}_n^k$...

Conclusion

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

- T-count optimization algorithm using any $\mathcal{RM}$ 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 $\mathcal{C}_n^k$...
- Optimizing $\{\text{CNOT}, T, H\}$ circuits
  - About 75% done
  - Requires *partial* decoding – decoding with some bits known

Thank you!