MPMCT ゲートの挿入による論理関数を実現する量子回路のコスト削減手法

Cost reduction technique of quantum circuit for realizing the logic function by the insertion of MPMCT gate

櫛田 耕平 †　　山下 茂 ‡
Kohei Kushida　Shigeru Yamashita

1. はじめに

1965 年，物理学者ゴードン · ムーアは集積回路の性能向上の指標としてムーアの法則を示した [1]．以降，集積回路の性能はムーアの法則に則るように向上し続けている．しかし，近い将来この法則は破綻すると考えられている [2]．したがって，集積回路の性能に比例する古典計算機の性能も，同じく限界を迎えると考えられている．ただし，本論文で扱う古典計算機とは，現在主流となっている CMOS ベースの回路を用いた計算機のことを指す．

そこで近年，従来とは全く異なる計算方式に基づく量子計算を用いた，量子計算機の研究が注目されている．量子計算機では，量子力学における量子の振る舞いを物理的に利用している [3] [4]．また，古典計算機では現実的な時間で解くことのできない問題を，量子計算機では解くことができるという利点がある [5]．

量子計算機の研究における共通理解として，一般的な量子アルゴリズムには論理関数を計算する部分が含まれることが知られている [6]．また，扱う論理関数が問題ごとに異なるために，その論理関数を計算するための回路はその都度設計し直される必要がある．よって，論理関数を計算するための回路に対する，効率的な設計手法が求められている．この効率的な設計手法とは，回路全体の量子ゲートの合計量子コストが小さくなるような設計手法のことである．

論理関数を計算する量子回路の設計手法としては，Arabzadeh の手法が存在する [7]．Arabzadeh の手法は，量子回路における論理関数の特性を考慮したカルノー図を用いて，元の回路よりも量子コストの小さな回路を設計することができる．しかし，この手法では量子コストを削減できる論理関数が限られ，また，削減できる量子コストが小さいという問題がある．

本論文では，Arabzadeh の手法とは異なる量子回路の特性を利用した回路設計手法を提案する．その特性とは，論理関数を表すゲートの直前に MPMCT (Mixed Polarity Multiple-Control Toffoli) ゲート [8] を挿入することで，その論理関数の真理値表における入力の組み合わせを入れ替えることができるというものである．この提案手法と Arabzadeh の手法を用いた場合の量子回路の量子コストを比較する．その結果，本研究で扱った論理関数の全てにおいて，提案手法の方が量子コストの小さな回路を設計できることを示した．

† 立命館大学大学院, Graduate School of Information Science and Engineering, Ritsumeikan University
‡ 立命館大学, Ritsumeikan University

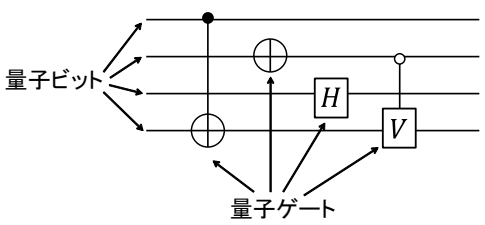

図 1: 量子回路

2. 量子回路に関する基礎知識

本章では，量子回路と各量子ゲートについて述べる．

2.1 量子回路

量子回路とは，量子ビットへの演算を量子ゲートを用いて表した図のことである．量子回路と量子ゲートは，古典計算機における古典回路と論理ゲートに似た役割を持っている．例えば，量子ゲートは論理ゲートと同様に入力と出力を持ち，基本的な演算を実装している．そして，複数の量子ゲートの組み合わせにより，あらゆる動作を表す量子回路を構成することができる．一方，古典回路との差異は，量子回路は可逆性を保持しなければならない点である．同様に，量子ゲートも可逆性を保持しなければならない．よって，量子回路と量子ゲートは，ある入力値に対して唯一の出力値を持ち，また，ある出力値に対して唯一の入力値を持つという関係性を保持している．

量子回路において，量子ビットと量子ゲートは図 1 のように表される．量子ビットは横線で表され，左側が入力，右側が出力である．量子ゲートは種類ごとに表記が異なるが，左側が入力，右側が出力であることは共通である．

2.1.1 量子ビット

量子ビットとは，量子力学の特性を利用できるビットのことである．例えば，量子ビットには光の偏光や電子のスピンなどのミクロ系が使用できると考えられている．

量子ビットの特徴として，1 量子ビットは従来のビットと同じような 0 と 1 の状態に加え，0 と 1 の重ね合わせ状態を保持することができる．0 と 1 の重ね合わせ状態とは，1 ビットで確率的に 0 と 1 の状態を保持している状態のことである．つまり，1 量子ビットは，ある確

率で 0 を示し，ある確率で 1 を示す，という状態を保持可能である．

重ね合わせ状態の n 量子ビットは，2^n 通りの組み合わせ全てを重ね合わせて表現できる．そして，重ね合わせ状態の n 量子ビットに演算を行うことで，2^n 通りの演算を 1 度に実行できる．2^n 通りもの演算を 1 度に実行できることは，古典計算機には無い利点であり，量子計算機が古典計算機の処理能力を大きく凌ぐ理由である．

2.1.2 量子ゲート

量子ゲートは，量子ビットを入出力として，その量子ビットの量子状態を変化させる役割を持つ．また，量子ゲートはユニタリ変換で表現される．ユニタリ変換は必ず逆変換を持つため，可逆性のある変換である．

量子ゲートはコントロールビットとターゲットビットで構成されている．コントロールビットとは，そのゲートが作用するかどうかの判定を行うビットである．それに対して，ターゲットビットとは，そのゲートによる演算を作用させるビットである．

コントロールビットは，正極か負極のどちらかの性質を持つ．これは極性 (polarity) と呼ばれる．本論文では，正極 (負極) の性質を持つコントロールビットをそれぞれ p (n) -コントロールビットと呼ぶ．p– コントロールビットはその量子ビットが 1 のとき，また，n– コントロールビットはその量子ビットが 0 のとき，ゲートを作用させる．この極性に従いゲートを作用できるときのコントロールビットの状態を，本論文ではコントロールビットが真であると呼ぶ．すなわち，p– コントロールビットが 1 のときと n– コントロールビットが 0 のとき，そのコントロールビットは真である．

量子ゲートを図示する際，一般に p– コントロールビットは黒丸，n– コントロールビットは白丸で表記される．例として，図 1 における 1 つ目のゲートは p– コントロールビットを持ち，4 つ目のゲートは n– コントロールビットを持つことを表している．一方，ターゲットビットの表記は量子ゲートの種類によって異なる．以下では，本論文で扱う量子ゲートを紹介する．

NOT ゲートとは，論理ゲートの NOT ゲートに相当する操作を行う量子ゲートである．すなわち，ターゲットビットが 0 ならば 1 を，1 ならば 0 を出力する．NOT ゲートは 1 量子ビットに対するゲートであり，ターゲットビットのみで構成されるため無条件で作用する．図 2 に NOT ゲートを示す．

CNOT ゲート (controlled-NOT) とは，1 コントロールビットと 1 ターゲットビットで構成された 2 入力 2 出力の量子ゲートである．CNOT ゲートは，コントロールビットが真のときのみ，ターゲットビットに NOT ゲートを作用させる．図 3 に，p– コントロールビットを持つ CNOT ゲートと，n– コントロールビットを持つ CNOT ゲートを示す．前述した通り，p– コントロールビットは黒丸，n– コントロールビットは白丸で表記される．

トフォリゲートとは，CCNOT (controlled-controlled-NOT) ゲートのことで，2 コントロールビットと 1 ターゲットビットで構成された，3 入力 3 出力の量子ゲートである．トフォリゲートは，2 つのコントロールビットが共に真であるときのみ，ターゲットビットに NOT ゲートを作用させる．また，2 つのコントロールビットはそれぞれ極性を持つため，4 種類のトフォリゲートが存在する．図 4 にその 4 種類のトフォリゲートを示す．また，表 **??**は図 4 の一番目のトフォリゲートの真理値表である．

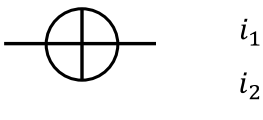

図 2: NOT ゲート

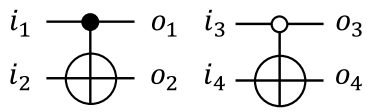

図 3: CNOT ゲート

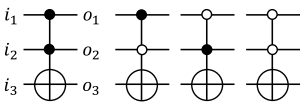

図 4: 4 種類のトフォリゲート

表 1: 図 4 の 1 番目のトフォリゲートの真理値表

i_1	i_2	i_3	o_1	o_2	o_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

MPMCT (Mixed-Polarity Multiple-Control Toffoli) ゲートとは，極性の混同する m 個のコントロールビットを持つ m+1 入力 m+1 出力の量子ゲートである ($m \geq 0$)．MPMCT ゲートは，m 個のコントロールビットの全てが真のときのみ，ターゲットビットに NOT ゲートを作用させる．つまり，上記の NOT ゲート，CNOT ゲート，トフォリゲートはそれぞれ，$m = 0$, $m = 1$, $m = 2$ とした場合の MPMCT ゲートであると考えてよい [8]．

本論文では，MPMCT ゲートを C^{m}NOT (C; t) と表記する．C はコントロールビットの集合を表し，t は 1 つのターゲットビットを表す．ただし，$C \cap t = \emptyset$ である．もし，C のあるコントロールビットが n– コントロールビットならば，そのビットの変数名に $\overline{\ }$ を付けることで p– コントロールビットと区別する．また，各 p– コントロールビットの値の積を P_x，各 n– コントロールビットの値の否定の積を N_x とする．これらの表記を用いると，図 5 に示すように，MPMCT ゲートの入力が $\{C,\ t\}$ のとき，出力は $\{C,\ t \oplus P_x \cdot N_x\}$ と表記できる．

2.2 量子コスト

量子コストとは，量子回路を設計するために必要となるコストのことである．本論文では，量子コストは，量

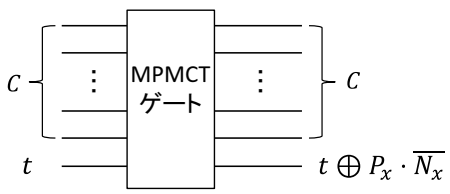

図 5: MPMCT ゲートの入出力

子回路内の基本量子ゲートの合計数に等しいと考える．基本量子ゲートとは，量子ゲートを構成する最小単位となるゲートのことで，量子コストは 1 である．

基本量子ゲートには，NOT ゲートと正極のコントロールビットを持つ CNOT ゲートに加え，正極のコントロールビットを持つ CV (Contorolled-V) ゲートと $\mathrm{CV}^\dagger$ (Contorolled-$\mathrm{V}^\dagger$) ゲートなどが該当する．そして，全ての MPMCT ゲートは基本量子ゲートのみの形に分解することができる．例えば，図 1 における 1 つ目のトフォリゲートを分解すると，図 6 のように 5 つの基本量子ゲートで表すことができる．したがって，このトフォリゲートの量子コストは 5 である．[7]

本論文では，MPMCT ゲートの量子コストとして，文献 [8] に記述された量子コストを用いている．文献 [8] によると，$n-$ コントロールビットが増えるほど量子コストは大きくなる．また，本論文では，回路内に複数の MPMCT ゲートが存在する場合，各 MPMCT ゲートの量子コストの和を，回路全体の量子コストとして計算する．

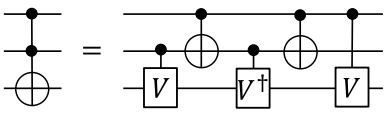

図 6: トフォリゲートの分解

3. 論理関数を実現する量子回路の設計手法

本章では，論理関数を実現する量子回路の設計手法として，カルノー図を用いた Arabzadeh の手法を紹介する [7].

3.1 論理関数を実現する量子回路

量子計算機の能力を発揮するためには，量子計算を上手く活用した量子アルゴリズムが必要となる．また，一般的な量子アルゴリズムは，論理関数を計算する部分を含んでいる．よって，量子計算機で論理関数を計算するために，論理関数を量子回路上に実現する設計手法が求められる．

論理関数の論理式が真となるときの各変数の論理は，その論理関数の各最小項の論理と等しくなる．よって，k 個の最小項を持つ n 変数の論理関数 $f(x_1, \ldots, x_n)$ を実現する量子回路は，コントロールビット数 n の MPMCT ゲートを k 個用いて設計できる $(0 \leq k \leq 2^n)$．このとき，各 MPMCT ゲートのコントロールビットの極性は，各最小項の変数の論理に対応する．つまり，1 つの最小項に対して，変数 x_i が 1 ならば $p-$ コントロールビットを，変数 x_i が 0 ならば $n-$ コントロールビットを i 番目のビットに持つ MPMCT ゲートを用いることになる $(1 \leq i \leq n)$．また，各 MPMCT ゲートのターゲットビットは共通する 1 ビットのみである．例えば，表 2 に示した 4 個の最小項を持つ 4 変数の論理関数は，図 7 のように実現できる．そして，表 2 の 1 つ目の最小項 0101 と図 7 の 1 つ目のゲートのコントロールビットの極性は一致している．

図 7 において，最終的なターゲットビットの出力は，

$$t \oplus x_2x_4\bar{x_1}\bar{x_3} \oplus x_2x_3\bar{x_1}\bar{x_4} \oplus x_1x_4\bar{x_2}\bar{x_3} \oplus x_1x_3\bar{x_2}\bar{x_4}$$

となる．この出力の $x_2x_4\bar{x_1}\bar{x_3} \oplus x_2x_3\bar{x_1}\bar{x_4} \oplus x_1x_4\bar{x_2}\bar{x_3} \oplus x_1x_3\bar{x_2}\bar{x_4}$ の部分は，ゲートのコントロールビットへの入力が最小項と等しいときのみ 1 となることから，$f(x)$ を表していることが分かる．よって，最終的なターゲットビットの出力は $t \oplus f(x)$ となる．そしてこの出力は，

$$\begin{aligned}
f(x)=0 \text{ のとき} \quad & t=0 \quad \text{ならば} \quad t \oplus f(x) = 0 \\
& t=1 \quad \text{ならば} \quad t \oplus f(x) = 1 \\
f(x)=1 \text{ のとき} \quad & t=0 \quad \text{ならば} \quad t \oplus f(x) = 1 \\
& t=1 \quad \text{ならば} \quad t \oplus f(x) = 0
\end{aligned}$$

となっている．したがって，$f(x) = 1$ のときのみ，ゲートの入力 t に対して出力 $t \oplus f(x)$ の真偽が反転することを利用して，論理関数の計算を行うことができる．ただし，古典回路を設計する場合とは異なり，実現する論理関数 $f(x)$ の論理式は，各最小項の論理の排他的論理和となっている．

表 2: 4 個の最小項を持つ 4 変数の論理関数の真理値表の例

x_1	x_2	x_3	x_4	$f(x)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

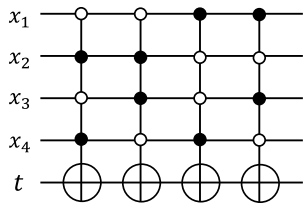

図 7: 表 2 の論理関数を実現する量子回路

3.2 カルノー図を利用した Arabzadeh の手法

論理関数を実現する量子回路の設計の問題点は，解く問題が異なれば扱う論理関数も異なることから，その都度設計し直さなければならない点である．よって，論理関数を計算するための量子回路に対する，効率的な設計手法が望まれている．効率的な設計手法とは，回路の量子コストが小さくなるような設計手法のことである．そのような設計手法として，カルノー図を利用した Arabzadeh の手法がある [7].

カルノー図は古典計算において，論理関数の論理式を最適化する目的で利用されている．つまり，Arabzadeh の手法は，所望の論理関数を量子回路で実現する際に，カルノー図を使って最適化された論理式を用いることで量子コストを削減する手法である．

本論文では，あるカルノー図の全セル数をカルノー図のサイズと呼び，カルノー図で用いられるループに含

まれるセル数をそのループのサイズと呼ぶ．また，カルノー図における空白のセルと 1 の書かれたセルのことを，0 値のセルと 1 値のセルと呼ぶ．n 入力 1 出力の論理関数の 1 つの最小項は，C^{n}NOT ゲートとして表現できる．そして，ある論理関数のカルノー図における 1 つの 1 値のセルは，その論理関数の 1 つの最小項に対応している．つまり，サイズ 2^n のカルノー図における 1 つの 1 値のセルは，C^{n}NOT ゲートを 1 つ用いて表現できる．

古典回路において，カルノー図による論理式の最適化は，いくつかのルールに従い行われる．しかし， 3.1 節で述べたように，量子回路と古典回路では，論理関数の論理式の表現が異なる．よって，古典回路で適用される最適化のルールは，量子回路においては適用することができない．そこで Arabzadeh の手法では，カルノー図から量子回路へ変換するために CTR (Common-Target Rule) と呼ばれる以下のルールを適用する．

(a) 全ての 1 値のセルは，少なくとも 1 つのループに属する．

(b) 各ループのサイズは必ず 2^p $(p \geq 0)$ とする．

(c) 各ループのサイズは最大にする．

(d) 各 1 値のセルは奇数個のループに属する．

(e) 各 0 値のセルは偶数個のループに属する．

(f) ループ数を最少にする．

CTR を適用した Arabzadeh の手法を用いるならば，表 2 の論理関数は，図 8 に示すようにループを選択することで，最終的に図 9 に示す量子回路によって実現できる．図 7 に示した量子回路の量子コストと比較すると 80 から 20 へと削減できていることが分かる．よって，この手法を用いれば，図 7 のように真理値表から直接量子回路へと変換する場合に比べると，大幅なコスト削減が可能である．

Arabzadeh の手法は，カルノー図における 1 値のセルの初期位置からループのサイズや数を求めることで，コスト削減を行っている．つまり，1 値のセルの位置を初期位置から移動させることがないため，最終的に選択できるループの形やサイズなどが大きく制限されてしまう場合がある．例として図 10 と図 11 に，Arabzadeh の手法において，4 個の最小項を持つ 4 変数の論理関数の中で，削減後の量子コストが最も大きくなる論理関数のカルノー図と量子回路を示す．図 11 の量子コストは 80 から 62 へと削減できているが，図 9 と比較すると削減後の量子コストは非常に大きくなっている．これは，初期位置と選び得るループの形やサイズの関係から，図 10 よりも良いループの選択が不可能なためである．次節では，Arabzadeh の手法とは異なる量子回路の特性を利用した回路設計手法を提案する．提案手法ならば，図 10 に示したカルノー図であっても，量子コスト 30 の量子回路を設計できる．

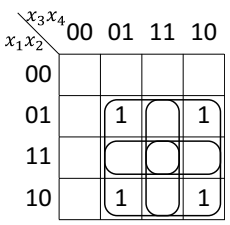

図 8: 表 2 のカルノー図に対するループの選択パターン

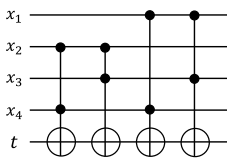

図 9: 図 8 を実現する量子回路

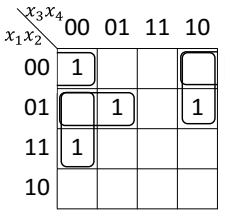

図 10: Arabzadeh の手法における，最悪な 1 値のセルの初期位置の例

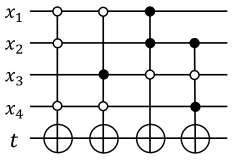

図 11: 図 10 を実現する量子回路

4. MPMCT ゲートの挿入によるコスト削減手法

本章では，Arabzadeh の手法よりも効率的な，MPMCT ゲートの挿入による設計手法を提案する．

4.1 MPMCT ゲートの挿入

Arabzadeh の手法では，カルノー図の 1 値のセルの位置を，どのような場合でも初期位置のまま扱う．そのため，初期位置の状態によっては，効率よく小さな回路を設計できない．したがって，初期位置を移動させることで，選択できるループの形やサイズを変更する手法を考案した．この提案手法は，MPMCT ゲートの挿入により実現している．

n 変数の論理関数を実現する $n+1$ 入力の量子ゲート G を $G(C;\ t)$ ，$C = \{x_1,\ ...,\ x_n\}$，$t = x_t$ とする．このとき，C^{m}NOT $(C';\ t')$ ，$C' \subset C$，$t' \in C$，$C' \cap t' = \emptyset$ $(1 \leq m < n)$ となる MPMCT ゲートをゲート G の直前に挿入する．つまり，n 変数の論理関数を実現する量子ゲートの n コントロールビットの内，m ビット $(1 \leq m < n)$ をコントロールビットとして持ち，1 ビットをターゲットビットとして持つ C^{m}NOT ゲートを挿入する．すると，量子ゲート G のコントロールビットの論理が，MPMCT ゲートが作用する論理となる場合のみ，MPMCT ゲートが作用する G のコントロールビットの論理を反転させることができる．また，挿入する MPMCT ゲートは複数であってもよい．

量子ゲート G のコントロールビットは，ゲート G が実現している論理関数の入力変数を表している．つまり，MPMCT ゲートの挿入によって，ゲート G のコントロールビットの論理を変更することで，その論理関数のカルノー図も変更することになる．複数の MPMCT ゲートを挿入した場合は，ある MPMCT ゲートによって変更

されたカルノー図に対して，再度変更を行うことになる．例として，表 2 の論理関数を実現する量子ゲート G_1 の直前に，C^{1}NOT $(x_3;\ x_4)$ を挿入する場合について説明する．挿入後の量子回路を図 12 に示す．また，挿入前のカルノー図を図 13 に，挿入後のカルノー図を図 14 に示す．図 13 と図 14 を比較すると，最小項 0110 が 0111 へ，1010 が 1011 へと移動していることが分かる．このように，MPMCT ゲートの挿入により，カルノー図における 1 値のセルの初期位置を移動させることが可能になる．

正確には，MPMCT ゲートの挿入は，1 値のセルを移動させるのではなく，図 15 のように 4 対のセルの内容を入れ替えることに相当している．つまり，図 14 は，図 15 のようにセルの内容が入れ替わっている．このとき，本論文では，0 値のセルを図には示さずに，1 値のセルの移動のみを考える．図 15 に示した 4 つの矢印は，C^{1}NOT $(x_3;\ x_4)$ によって論理が変更されるセルの対を表している．ただし，論理関数の変数の数や，挿入する MPMCT ゲートのコントロールビット数が異なるとき，この矢印の数や位置は異なる．例えば，n 変数の論理関数に C^{m}NOT ゲートを挿入するとき，矢印の数は 2^{n-m-1} 個となり，2^{n-m} 個のセルが入れ替わることになる．1 つの MPMCT ゲートにより変更されるビットは 1 ビットのみであるため，入れ替わるセルは 1 ビットの論理のみ異なる，隣接したセル同士である．

図 16 のように，図 12 の量子回路に更に C^{1}NOT $(x_1;\ x_2)$ を挿入する．図 16 の量子回路を表すカルノー図は図 17 のようになる．図 17 を見ると分かるように，このカルノー図の全ての 1 値のセルは同一のループに属している．

ところが，図 12 や図 16 の量子回路は，図 7 と等価ではない．なぜなら，MPMCT ゲートの挿入のためにゲート G_1 のコントロールビットに当たるビットの論理が変更されているからである．よって，MPMCT ゲートの挿入によって変更したビットの論理を元に戻さなくてはならない．MPMCT ゲートの挿入によって変更されたビットは，同じ MPMCT ゲートを量子ゲート G の直後に挿入することで元に戻すことができる．複数の MPMCT ゲートを挿入した場合は，後に挿入した MPMCT ゲートから順に挿入する．つまり，図 16 の量子回路に対しては，図 18 のように C^{1}NOT $(x_1;\ x_2)$ と C^{1}NOT $(x_3;\ x_4)$ をゲート G_1 の直後に挿入することになる．図 18 と図 7 の量子回路は論理的に等価である．

以上より，Arabzadeh の手法では初期位置のままだった 1 値のセルの位置を，提案手法では移動することができる．これにより，提案手法では，選択できるループの形やサイズが増えることになり，Arabzadeh の手法よりも量子コストの小さな量子回路設計を行うことができると考えられる．例えば，図 17 と図 18 から設計できる量子回路は図 19 のようになり，量子コストは 9 である．Arabzadeh の手法による図 9 の量子回路の量子コストと比較すると 20 から 9 まで削減できている．

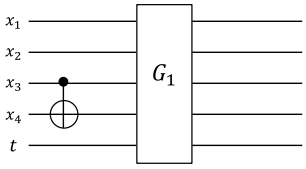

図 12: 表 2 を実現するゲート G_1 に対する，C^{1}NOT $(x_3;\ x_4)$ の挿入

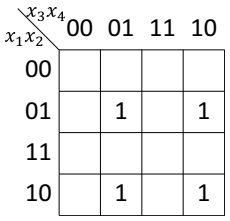

図 13: 図 12 の C^{1}NOT $(x_3;\ x_4)$ 挿入前のカルノー図

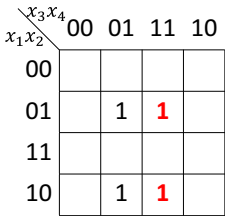

図 14: 図 12 の C^{1}NOT $(x_3;\ x_4)$ 挿入後のカルノー図

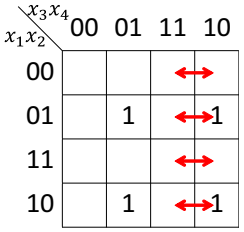

図 15: C^{1}NOT $(x_3;\ x_4)$ の挿入により入れ替わるセル

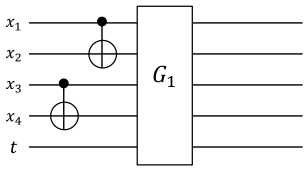

図 16: 図 14 の回路に対する，C^{1}NOT $(x_1;\ x_2)$ の挿入

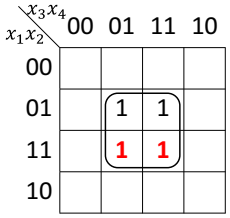

図 17: 図 16 の C^{1}NOT $(x_1;\ x_2)$ 挿入後のカルノー図

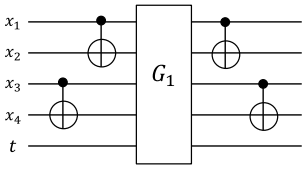

図 18: 図 16 の論理を元に戻すための MPMCT ゲートの挿入

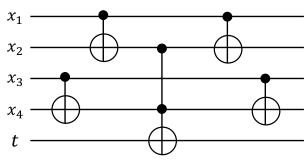

図 19: 図 17 と図 18 による，最終的な量子回路

Algorithm 1 提案アルゴリズム

Require: M : 各最小項の変数の論理を持つ文字列配列
1: $n \Leftarrow$ 変数の数
2: $k \Leftarrow$ 最小項の数
3: $H \Leftarrow minHamming(M)$
4: $R \Leftarrow MakeRoop(H, k)$
5: $P \Leftarrow SelectRoop(M, R, k)$
6: $i \Leftarrow 0$
7: $d \Leftarrow 1$
8: **while** $d \neq 0$ **do**
9: $i \Leftarrow i + 1$
10: $D \Leftarrow GetDiffBit(M, P)$
11: $d \Leftarrow maxD(D)$
12: **for** $j = d$ *to* 1 **do**
13: **if** $SearchGate(M, D, i, j)$ が新しくゲートを発見 **then**
14: G へ新しいゲート情報を格納
15: $M \Leftarrow RewriteMinterm(M, G)$
16: $i \Leftarrow 0$
17: $j \Leftarrow 0$
18: **end if**
19: **end for**
20: **end while**

4.2 提案アルゴリズム

提案手法では，1 値のセルの位置をほとんど自由に動かせるため，様々な移動のパターンが考えられる．つまり，最終的な回路の量子コストを最小にする移動のパターンを探さなくてはならない．そこで，最終的な回路の量子コストが最小となるときの，回路内の量子ゲートを求めるアルゴリズムを提案する．ただし，対象とする論理関数は，最小項の数が 2^p 個 ($p \geq 0$ かつ p は整数) である論理関数に限る．

提案アルゴリズムを説明するために，提案手法の擬似コードを Algorithm 1 に示す．入力は，論理関数の各最小項の論理であり，文字列配列 M に格納する．ただし，最小項の数は 2^p 個 ($p \geq 0$) に制限する．また，各入力のサイズと M のサイズより，論理関数の変数の数 n と最小項の数 k を取得する．

3 行目では，関数 $minHamming()$ の返り値を文字列 H に格納している．引数は，文字列配列 M である．この関数では，1 つの最小項 M_t と残りの最小項 M_d とのハミング距離の合計を計算している．そして，k 個の計算結果から最小となる最小項 M_t を返り値とする．この関数によって，その後のセルの移動において中心となる最小項を求めることができる．その最小項は最終的に移動しないことになる．それは，最小移動数を求めた時，必ず 1 つ以上のセルは元の位置から移動しないと推測されるためである．ただし，最小移動数のとき，必ず最終的な回路の量子コストが最小になるとは限らないことが分かっている．もしも，この後の全ての処理を，M の全ての最小項に対して行うならば，最適解を得ることが可能かもしれないが，当然，計算量は膨大になると予想される．したがって，提案アルゴリズムではヒューリスティックな手法を用いている．

4 行目では，関数 $MakeRoop()$ の返り値を 2 次元文字列配列 R に格納している．引数は，文字列 H と最小項数 k である．この関数では，カルノー図上で文字列 H を要素に含み，かつ大きさが k となるループを全て列挙している．そして，その各ループに含まれる各セルの論理を返り値とする．この関数によって，$minHamming()$ において求めた移動させないセル，を含んだ全ループを生成することができる．その全ループ数は ${}_nC_{log_2 k}$ 個存在する．各ループのサイズは，初めに与えられた最小項数 k である．ここで，ある 1 つのループは，各最小項の最終的な移動先になる可能性があるセルの集合であると言える．つまり，図 17 におけるループのような，全ての移動後のセルを囲むことのできるループを表している．また，この例でも，0101 のセルは元の位置から移動していないことが分かる．このように，全最小項の移動先は，1 つのループによって表現できる．したがって，この関数では，条件を満たす全てのループを生成している．

5 行目では，関数 $SelectRoop()$ の返り値を文字列配列 P に格納している．引数は，文字列配列 M と 2 次元文字列配列 R，最小項数 k である．この関数では，M と R 内の 1 つのループとの合計ハミング距離を計算している．ここで，1 つのループに対して，M の 1 つの要素の移動先の候補は $k-1$ 個存在する．つまり，1 つのループと M の各要素との組み合わせは $(k-1)!$ 通り存在する．その全ての組み合わせについて合計ハミング距離を計算する．そして，この計算を R 内の全ループに対して行う．その計算結果の中で，合計ハミング距離が最小となるときの，M に対するループの最小項を返り値とする．この関数によって，M の各要素の移動距離が最も短くなるループの選択と，ループのどのセルに移動させるかを判定することができる．

6 行目では，変数 i を用意し，初期値として 0 を格納している．i は，9 行目で i を 1 ずつインクリメントすることで，8 行目の while によるループ回数を保持する．i は 10 行目の関数 $GetDiffBit()$ の引数として使用される．また，後述する 14 行目の if 文が実行されたとき，i は 0 に更新される．

7 行目では，変数 d を用意し，初期値として 1 を格納している．d は 8 行目の while ループの終了条件として使用される．また，12 行目の for ループのカウンタの初期値としても使用される．d の値は，11 行目の関数 $maxD()$ の返り値を格納する度に更新される．

8 行目から 20 行目までは，while ループになっている．終了条件は $d = 0$ となったときである．

10 行目では，関数 $GetDiffBit()$ の返り値を 2 次元整数配列 D に格納している．引数は，文字列配列 M と P である．関数 $SelectRoop()$ によって，文字列配列 P には，文字列配列 M と 1 対 1 対応の最小項が格納されている．よって，文字列配列 P の 1 つの最小項は，文字列配列 M の 1 つの最小項の移動先のセルの最小項を表している．この関数では，文字列配列 M の 1 つの最小項と，対応する文字列配列 P の 1 つの最小項において，論理の異なっているビットが何ビット目であるかを求めている．つまり，文字列配列 M の 1 つの最小項の何ビット目を反転させれば，移動先のセルの最小項と等しくなるか，ということを求めている．これを，文字列配列 M の全ての最小項について求め，返り値としている．この関数によって，1 つの MPMCT ゲートの挿入によって，複数の最小項を同時に動かすことが可能かどうかの判定材料を得ることができる．実際に同時に動かす

ことができるかの判定は，13 行目の関数 $SearchGate()$ において行っている．

11 行目では，関数 $maxD()$ の返り値を d に格納している．引数は D である．関数 $maxD()$ では，D を参照して，ある 1 ビットに対して反転する必要がある最小項の数を求めている．その内で，最大の最小項数を返り値としている．

12 行目から 19 行目までは，for ループ文になっている．ループカウンタ j は d で初期化され，1 ずつデクリメントされる．ループ終了条件は $j < 1$ となったときである．

13 行目では，if 文によって，関数 SearchGate() が新しく挿入すべき MPMCT ゲートを発見できたかどうかを判定している．発見できたと判定されたときは，14 行目から 17 行目の処理が実行される．関数 SearchGate() の引数は，文字列配列 M，文字列配列 P，整数 i，j である．この関数では，j 個の最小項に共通な反転させたいビットを反転させることのできる，コントロールビット数 i の MPMCT ゲートが存在するかどうかを探索している．j は 1 つの MPMCT ゲートによって同時に移動させることのできる最小項の数を表している．j は 12 行目の for ループのループカウンタであるため，8 行目で求めた d から 1 までの間の数である．ここで，挿入する MPMCT ゲートの総数を少なくするために，1 つの MPMCT ゲートによって移動可能な最小項の数が多くなるように探索する．したがって，MPMCT ゲートの探索は，同時に移動できる可能性のある最小項の数の最大値 d から順に行う．最大値が d の理由は，反転させたい共通ビットを持つ最小項数の最大値が d であることから分かる．最小値は 1 である．i は 1 つの挿入する MPMCT ゲートのコントロールビット数を表している．i は 1 から n-1 までの間の数である．i が少ない順に探索を行っている理由は，MPMCT ゲートのコントロールビット数が少ないほどその MPMCT ゲートの量子コストは小さくなるためである．コントロールビット数の最小値は 1 である．最大値は，1 つのターゲットビット以外の全てのビットをコントロールビットに使用するときのため，n-1 である．関数 SearchGate() の返り値は，新しく発見した MPMCT ゲートのゲート情報である．ゲート情報とは，MPMCT ゲートの $p-$ コントロールビットと $n-$ コントロールビット，ターゲットビットが，それぞれ何ビット目であるか，という情報のことである．この返り値は，14 行目でゲート情報配列 G へと格納される．

15 行目では，RewriteMinterm() 関数によって，文字列配列 M を更新している．引数は文字列配列 M とゲート情報配列 G である．13 行目の関数 SearchGate() によって発見された MPMCT ゲートによって，文字列配列 M のいくつかのビットは反転される．よって，MPMCT ゲートが発見される度に，文字列配列 M を更新していく必要がある．更新の度に，文字列配列 M は文字列配列 P の論理に近づいていき，最終的に文字列配列 P の論理と等しくなる．関数 RewriteMinterm() では，この文字列配列 M の更新を行っている．

16，17 行目では，i，j を更新している．その結果，12 行目の for ループを抜け，8 行目の while ループを初めからやり直す処理が行われる．

8 行目の while ループを抜けたとき，最適な MPMCT ゲートを全て発見することができる．このアルゴリズムによって，提案手法によって設計される量子回路の量子ゲートは，ゲート情報を保持する G に格納される．ただし，前述した通り，このアルゴリズムはヒューリスティックである．よって，提案手法で求めることができる最適解とは異なる解や，既存手法よりも量子コストの高い解が得られてしまう場合が存在する．これは，関数 $minHamming()$ や関数 $SelectRoop()$ などで，解の探索範囲を限定しているためであると推測される．したがって，解の探索範囲を広げることで，アルゴリズムの精度を高めることが可能であると考えられる．しかし，そのようなアルゴリズムは，計算量が膨大になることが予測される．今回の実験では，Algorithm 1 に示したヒューリスティックアルゴリズムを利用したプログラムによって提案手法の検証を行っている．

5. 実験結果と考察

本章では，Arabzadeh の手法と提案手法により設計された量子回路の量子コストを比較した実験結果とその考察を示す．

5.1 評価方法

提案手法の評価を行うために，Algorithm 1 に示したアルゴリズムを用いて，提案手法を C++言語で実装した．

Arabzadeh の手法の評価を行うために，Alan Mishchenko によって開発された EXORCISM-4 と呼ばれるプログラムを使用した [9]．EXORCISM-4 は，本論文で扱った EXOR と最小項で構成された論理式に対して，ヒューリスティックな最適化を行うプログラムである．EXORCISM-4 による最適化は CTR に準拠しているため，Arabzadeh の手法の評価を行えると考えた．

評価は，提案手法と Arabzadeh の手法に対して，同じ論理関数を適用し，設計された量子回路の量子コストを比較することにより行った．また，今回扱った論理関数は，4 個の最小項を持つ 4 変数の論理関数に制限している．そのような論理関数は ${}_{16}C_4$=1820 通り存在する．

5.2 実験結果と考察

実験結果を表 3 に示す．表 3 の既存手法と提案手法は，各手法により設計された量子回路を量子コスト別に分けたときの個数を表している．また，平均コストは，各手法により設計された量子回路の量子コストの平均を表している．表 3 より，提案手法では，より多くの論理関数に対して，既存手法と比べて低コストでの設計が可能であることが分かった．また，平均では，既存手法による量子コストの約 38.0%のコスト削減が可能であることが分かった．

次に，各論理関数を実現する量子回路に対する量子コストの比較の結果，対象とした 1820 通りの論理関数の約 98.9%に当たる 1800 通りにおいて，提案手法により設計された量子回路の量子コストの方が小さくなることを確認できた．つまり，本実験で用いた提案アルゴリズ

表 3: 実験結果

量子コスト	既存手法	提案手法
$0-20$	193	900
$21-40$	1050	920
$41-60$	572	0
$61-80$	5	0
平均コスト	34.1	21.1

ムでは，最適な量子コストを求めることができない論理関数が 20 通り存在することがわかった．

ただし，提案手法による量子コストの方が大きくなってしまう 20 通りの論理関数においても，挿入する MPMCT ゲートを正しく選択することで，提案手法の方が量子コストが小さくなることが確認できた．よって，更に精度の高いアルゴリズムを用いたプログラムならば，この 20 通りの論理関数に対しても，提案手法の方が量子コストが小さくなると考えられる．したがって，提案手法を用いることで，4 個の最小項を持つ 4 変数の論理関数の全てに対して，Arabzadeh の手法よりも効率的な回路設計を行うことができる．

また，一部ではあるが，実験に用いた論理関数以外でも，提案手法を用いることで量子コストが小さくなることを確認している．特に，8 個の最小項を持つ 10 変数の論理関数といった，変数の数が多く，最小項の数が少ない関数では，大幅なコストダウンが確認された．よって，提案手法は，本実験で扱った論理関数だけでなく，様々な論理関数においても有用であると考えられる．

6. おわりに

本論文では，MPMCT ゲートの挿入によって，カルノー図のセルの位置を初期位置から移動させる手法を提案した．この手法は，セルの位置は初期位置のまま考えるという，カルノー図における既成概念を打ち破る全く新しい手法であると考える．本研究では，この新しい手法を量子回路設計における量子コスト削減のために用いた．その結果，4 個の最小項を持つ 4 変数の論理関数に対しては，既存手法である Arabzadeh の手法に比べて，必ず小さな量子コストとなる量子回路を設計可能であった．

この提案手法は，変数の数と最小項の数を変更した論理関数においても，非常に有用であると考えられる．よって，将来の研究としては，様々な論理関数に対して提案手法を適用した結果を示したい．また，提案アルゴリズムとして，更に精度が高いアルゴリズムを考案していきたい．

参考文献

[1] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, Vol. 38, No. 8, April 1965.

[2] International technology roadmap for semiconductors. `http://www.itrs.net/Links/2013ITRS/2013Chapters/2013ExecutiveSummary.pdf`.

[3] Richard P. Feynman. Cramming more components onto integrated circuits. *Foundations of Physics*, Vol. 16, pp. 507–531, 1986.

[4] David Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Mathematical and Physical Sci-ences*, Vol. 400, pp. 97–117, 1985.

[5] 西森秀稔. 量子アニーリングと d-wave. 情報処理学会論文誌, Vol. 55, pp. 1–6, 2014.

[6] Yamashita Shigeru, Minato Shin-ichi, and Miller D.Michael. DDMF: An Efficient Decision Diagram Structure for Design Verification of Quantum Circuits under a Practical Restriction. *IEICE Transactions on Fundamentals of Electronics,Communications and Computer Sciences*, pp. 3793–3802, 2008.

[7] Mona Arabzadeh, Mehdi Saeedi, and Morteza Saheb Zamani. Rule-based optimization of reversible circuits. In *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, pp. 849–854. IEEE, 2010.

[8] Zahra Sasanian and D. Michael Miller. Reversible and quantum circuit optimization: A functional approach. In *Reversible Computation 4th International Workshop, RC 2012, Copenhagen, Denmark, July 2-3, 2012. Revised Papers,*, pp. 112–124, 2013.

[9] Two-level exclusive sum-of-product minimization. `http://web.cecs.pdx.edu/~alanmi/research/min/minEsop.htm`.