複数出力を考慮した量子回路設計手法

Request of the Manuscript for IPSJ Kansai-Branch Convention 2015

小野田 将人 † 山下 茂 ‡ 櫛田 耕平 †
Masato Onoda Shigeru Yamashita Kohei Kushida

1. はじめに

量子計算は量子ビットと呼ばれる、従来のビット（以下古典ビット）とは異なる特殊なビットを用いた新しい計算方式である。古典ビットが 0 か 1 どちらかの値しか保持することができない 1 ビットを扱うものであるのに対し、量子ビットは 1 量子ビットにつき 0 と 1 の値を任意の割合で同時に保持することができる。これは、量子の重ね合わせと呼ばれる性質である。この性質を利用することで、n 量子ビットは 2^n 通りの状態を同時に保持するができ、この 2^n 通りの状態を持つ n 量子ビットに対して演算を行うことで、2^n 通りの状態全てを並列的に処理することができる。この並列処理により、量子コンピュータは古典コンピュータでは解くことのできない問題を解くことが可能だとされている。このことから、量子コンピュータは古典コンピュータの処理能力を上回るとされ、注目が集まっている。

量子コンピュータとは、量子状態を用いることで、ある種の計算を高速に実行するものである。量子コンピュータを実現する量子回路は、量子ゲートの組み合わせによって構成される。量子回路は与えられる論理関数ごとに設計する必要があるため、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。ここで、効率的な設計手法とは、より回路全体の量子コストが小さくなるような設計手法のことを意味する。

量子回路を設計する効率的な手法として、論理関数を表すゲートの直前に MPMCT（Mixed-Polarity Multiple-Control Toffoli）ゲートを挿入し、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で回路のコスト削減を図る手法が櫛田らにより提案されている [1]。この手法を用いることで、MPMCT ゲートを挿入する前の回路と比較して、大きく量子コストを削減した回路を作成することが可能である。しかしながら、この手法は出力を複数持つ量子回路に適用することができず、現在主流となっている多くの量子回路設計にそのまま利用することができないという問題点がある。そこで本稿では、CNOT ゲートを用いることで論理関数を複製することができるという特性を利用し、複数の出力を持つ量子回路に対応した効率的な回路設計手法を提案する。

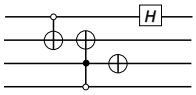

図 1: 量子回路

2. 量子コンピュータ

本章では量子コンピュータを構成する量子回路、また量子回路を構成する量子ゲートについて説明する。

2.1 量子回路

量子回路とは量子ゲートと量子ビットにより構成された、量子コンピュータの演算によって起こる量子状態の変化をモデル化したものである。量子回路は図 1 のように表される。量子回路において量子ビットは左側が入力、右側が出力であり、1 量子ビットは 1 本の直線で示される。n 量子ビットの場合は、n 本の直線を平行に並べることで示す。量子ゲートには様々な種類のゲートが存在するが、ゲートの左側が入力、右側が出力であることは、全ての量子ゲートにおいて共通である。

量子回路とそれを構成する量子ゲートは古典コンピュータにおける論理回路（以下古典回路）と論理ゲート（以下古典ゲート）と基本的な役割は同じである。しかしながら、量子回路は全体が常に可逆性を保持しなければならないという点で古典回路とは異なる。つまり、量子回路は入力値から出力値、出力値から入力値が一意に求められる必要がある。

2.2 量子ゲート

量子ゲートとは、量子ビットを入出力として、量子ビットの量子状態を変化させる役割を持つものである。量子ゲートを用いて行われる演算は、全てユニタリ行列を用いて表現できる。量子ゲートは、0 個以上のコントロールビットと 1 個以上のターゲットビットで構成される。コントロールビットはゲートが作用するかどうかを判定する量子ビットであり、ターゲットビットはゲートが作用することで状態が変化する量子ビットである。すなわち、量子ゲートはコントロールビットの入力を元に、ターゲットビットの量子状態を変化させる役割を持つ。

コントロールビットは極性と呼ばれる、正極か負極の性質を持つ。正極のコントロールビットは量子ビットが 1 の時、負極のコントロールビットはその量子ビットが

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

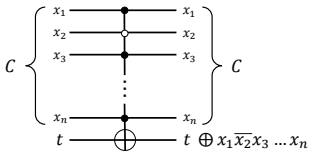

図 2: C^{n}NOT $(C;\ t)$ ゲートの一例

0 の時にゲートを作用させる。量子ゲートを表現する際、一般的に正極のコントロールビットは黒丸、負極のコントロールビットは白丸で表現される。一方、ターゲットビットは量子ゲートの種類により表現が異なる。本稿では、コントロールビットの極性に従いゲートが作用する場合、そのコントロールビットの状態を真であると言う。以下では様々な量子ゲートの中でも、本稿で重要となる MPMCT ゲートについて詳しく説明する。

MPMCT（Mixed-Polarity Multiple-Control Toffoli）ゲートとは、任意の極性の n 個のコントロールビットを持つ、$n+1$ 入力 $n+1$ 出力のゲートである。MPMCT ゲートの中でも、$n=0$、$n=1$ の MPMCT ゲートはそれぞれ NOT ゲート、CNOT ゲートと呼ばれている。MPMCT ゲートは、MPMCT ゲートの持つ全てのコントロールビットが真の場合、ターゲットビットに NOT ゲートを作用させる。コントロールビットは正極と負極の 2 つの極性を持つため、1 コントロールビットにつき 2 通りの組み合わせが存在する。よって、コントロールビット数が n の MPMCT ゲートには、2^n 通りの組み合わせが存在することがわかる。

本稿では、MPMCT ゲートを表記する際、C^{n}NOT $(C;\ t)$ と表記する。C は正極の場合は肯定、負極の場合は否定をとったコントロールビットの論理積を示し、t はターゲットビットを示す。また、コントロールビットの極性を区別するため、C が示すコントロールビットが負極である場合、否定を意味する記号である $\overline{\ }$ を対応するビットに付属させる。図 2 に、C^{n}NOT $(C;\ t)$ を表す MPMCT ゲートの一例を示す。図 2 において、x_1、$\overline{x_2}$、$x_3 ... x_n$ がこの MPMCT ゲートのコントロールビットである。よって、C はこれらのコントロールビットの論理積であるため、$C = x_1\overline{x_2}x_3 ... x_n$ となる。コントロールビットである x_1、x_2、$x_3 ... x_n$ の極性が全て真である場合のみ、この C^{n}NOT $(C;\ t)$ ゲートのターゲットビットが反転するため、ターゲットビットの出力が $t \oplus 1$ となる。

2.3 量子コスト

量子コストとは、量子回路を実装するために必要なコストのことである。本稿において、量子コストは、量子回路を構成する、ユニバーサル・ゲートセットに含まれる量子ゲートの数とする。ユニバーサル・ゲートセットとは、2 量子ビットゲートと 1 量子ビットゲートを用いることで、全てのユニタリ変換を構成できるゲートセットである。

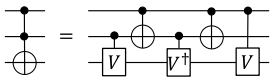

図 3: MPMCT ゲートの分解の一例

MPMCT ゲートは分解することで、ユニバーサル・ゲートセットのみで表現することができる。例として、図 3 の左に示された MPMCT ゲートは、右に示されたユニバーサル・ゲートセットに分解できる。したがって、図 3 の左に示された MPMCT ゲートのコストは、右に示されてた量子ゲート数と等しい 5 である。同一の回路内に複数の MPMCT ゲートが存在する場合、その回路全体のコストはそれぞれの MPMPC ゲートのコストの総和とする。

本稿では、MPMCT ゲートの量子コストとして、文献 [2] に示された量子コストを用いる。文献 [2] が示す通り、MPMCT ゲートのコントロールビット数が増加するほど、量子コストは大きくなる。そのため、効率的な量子回路を設計するためには、可能な限り回路全体の MPMCT ゲート数と、各 MPMCT ゲートのコントロールビット数を削減する必要がある。

3. 従来の量子回路設計手法

この章では、基礎的な量子回路の設計方法と、既存手法の説明をする。

3.1 論理関数を実現する量子回路設計手法

一般的な量子アルゴリズムは論理関数を計算する箇所を含んでいる [3]。そのため、量子回路を設計する際、論理関数を量子回路上に実現するための量子回路設計手法が必要となる。n 変数の論理関数 $f(x_1, x_2, \cdots, x_n)$ は、C^{n}NOT $(C;\ t)$ ゲートで実現できる。この C^{n}NOT $(C;\ t)$ ゲートは x_i が 1 ならば正、x_i が 0 ならば負の極性を持つコントロールビットを、i 番目のビットに持つ。この時、論理関数の論理式が真となる変数の組み合わせを最小項と呼ぶ。論理関数が複数の最小項を持つ場合、一つの最小項に対して一つの C^{n}NOT $(C;\ t)$ ゲートを用いることで実現する。すなわち、論理関数の最小項が m 個の場合、量子回路は C^{n}NOT $(C;\ t)$ ゲートを m 個用いることで実現できる（$0 \leqq\ m \leqq\ 2^n$）。例えば、表 1 に示した 4 変数 4 最小項の論理関数は、図 4 のように実現できる。

3.2 カルノー図を利用した効率的な量子回路設計手法

一般的に論理関数の論理式は、カルノー図を用いることで最適化することが可能である。この最適化を利用することで、一部の論理関数を量子回路で実現する際、より効率的な回路設計を行うことができる。

表 1: 4 変数 4 最小項を持つ
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

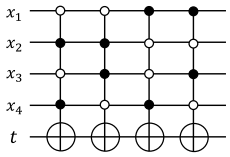

図 4: 表 1 の論理関数
を実現する量子回路

本稿では、カルノー図の 1 の記入されたセルを'1 値のセル'、0 の記入されたセルを'0 値のセル' と呼ぶ。n 入力 1 出力の論理関数のカルノー図における 1 値のセルは、それぞれその論理関数の最小項に対応している。そして、前述した通り、論理関数の最小項は C^{n}NOT ゲートを用いて表現することが可能である。つまり、サイズ $2^p (p \geqq 0)$ のカルノー図における 1 値のセルは、それぞれ C^{n}NOT ゲートを用いることで表すことができる。

古典回路におけるカルノー図による論理式の最適化は、1 値のセルを大きさ 2^p のループで囲うことで実現できる。ここでいうループとは、正方形または長方形の枠で囲まれた隣接するセルの集合のことである。サイズ 2^p のループで囲まれている 1 値のセルが表す論理式は、それぞれの 1 値のセルが示す論理積の、共通変数を用いて最適化することができる。例えば、図 5 に示すカルノー図は、サイズ 2^2 のループで 1 値のセルを囲うことができる。そのため、図 5 のカルノー図の論理式は、それぞれの 1 値のセルが示す論理積の排他的論理和ではなく、共通変数である x_2x_4 と表すことができる。

ループを用いて論理式を最適化することで、ループで囲うことのできる 2^p 個のセルは、2^p 個の C^{n}NOT ゲートではなく、1 つの C^{n-p}NOT ゲート表すことが可能になる。例として、図 5 のカルノー図の最適化する前の論理式を表す量子回路を図 6 の左に、最適化後の論理式を表す量子回路図 6 の右に示す。これは、サイズ 2^p のループにより論理式を最適化することで、C^{n}NOT ゲートの数を $\frac{1}{2^p}$ 倍、コントロールビットの数を $\frac{n-p}{n}$ 倍に削減できることを意味する。すなわち、論理関数を実現する C^{n}NOT ゲートの数とコントロールビットの数は、ループのサイズが大きいほど少なくなる。文献 [2] から、コントロールビット数が少ないほど、C^{n}NOT ゲートの量子コストが小さいことがわかる。よって、同じ論理関数のカルノー図であれば、より大きなループを用いて論理式を最適化することで、回路全体のゲート数とゲートの

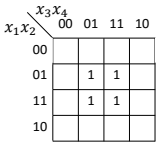

図 5: サイズ 2^2 のループで囲うことのできるカルノー図

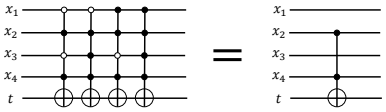

図 6: 図 5 の示す論理関数を実現する量子ゲート

持つコントロールビット数を少なくし、量子コストを小さくすることができる。

3.3 MPMCT ゲート挿入による回路コスト削減手法

ループを用いて論理式を最適化できるのは、カルノー図の 1 値のセルが隣接したセルに集合している場合のみである。しかしながら、多くの論理関数におけるカルノー図の 1 値のセルの初期位置は、隣接したセルに集合していない。そのため、このままだとカルノー図を用いた手法では、多くの論理関数を効率よく量子回路に実現することができない。そこで、カルノー図の 1 値のセルの位置を移動し、1 値を隣接したセルに集合させ論理式を最適化することで、量子コストを削減する手法が櫛田らにより提案されている [1]。櫛田らの手法 (以下、既存手法と呼ぶ) を用いることで、1 値が初期位置のカルノー図と比べて、よりサイズの大きなループを用いることができるため、回路全体の量子コストを小さくすることができる。この手法は、論理関数を実現する量子ゲートの前に C^{n}NOT ゲートを挿入し、カルノー図の 1 値を隣接したセルに移動させることにより実現される。

ある n 変数の論理関数を実現する量子ゲートを $G(C;\ t)$ とする $(C = x_1...x_n, t = x_t)$。この時、C^{m}NOT$(C';\ t')$ となる MPMCT ゲートを、ゲート G の直前に挿入する $(1 \leq m < n)$。ここで、C' と t' の示す論理式は、共に C の示す論理式に含まれる論理のみで表す必要がある。つまり、ゲート G の持つ n コントロールビットのうち m ビット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、ゲート G の直前に挿入する。挿入する C^{m}NOT ゲートは複数であってもよい。C^{m}NOT ゲートを挿入することにより、挿入した C^{m}NOT ゲートが作用する場合、C^{m}NOT ゲートのターゲットビットにあたる、ゲート G のコントロールビットの論理が反転する。

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。そのため、C^{m}NOT ゲートを挿入し、ゲート G のコントロールビットを変更することで、ゲート G が実現している論理の

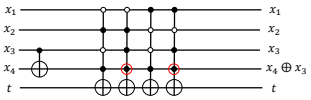

図 7: 図 4 に対する $\mathrm{C}^1\mathrm{NOT}(x_3; x_4)$ ゲートの挿入

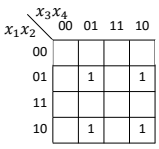

図 8: $\mathrm{C}^1\mathrm{NOT}\ (x_3;\ x_4)$ 挿入前のカルノー図

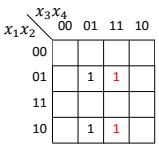

図 9: $\mathrm{C}^1\mathrm{NOT}\ (x_3;\ x_4)$ 挿入後のカルノー図

カルノー図を変更できる。複数の $\mathrm{C}^m\mathrm{NOT}$ ゲートを挿入した場合、ある $\mathrm{C}^m\mathrm{NOT}$ ゲートにより変更されたカルノー図を、さらに変更することになる。

例として、図 4 の量子回路の直前に、$\mathrm{C}^1\mathrm{NOT}(x_3; x_4)$ ゲートを挿入する。挿入後の回路を図 7 に示す。$\mathrm{C}^1\mathrm{NOT}(x_3; x_4)$ ゲートの挿入により、x_3 のビットが正極の場合、x_4 のビットの極性が反転する。よって、図 7 の場合、図 4 の左から 2 つ目と、左から 4 つ目のゲートの、x_4 のビットの極性が反転する。これは、カルノー図で表すと、図 10 のようなカルノー図の変更が行われることになる。つまり、図 8 のカルノー図が表している論理関数を実現する、図 4 のカルノー図は、$\mathrm{C}^1\mathrm{NOT}(x_3; x_4)$ ゲートの挿入により図 9 のように変更される。

図 8 のカルノー図と図 9 のカルノー図を比較すると、最小項 0110 が 0111 へ、1010 が 1011 へ移動していることが分かる。このように、$\mathrm{C}^m\mathrm{NOT}$ ゲートを挿入することで、カルノー図の特定のセルの内容を、隣接したセルの内容と入れ替えることができる。図 10 では 4 対の隣接したセルの内容を入れ替えているが、内容を入れ替えるセルの個数は、挿入する $\mathrm{C}^m\mathrm{NOT}$ ゲートにより異なる。例えば、n 変数の論理関数に $\mathrm{C}^m\mathrm{NOT}(C';\ t')$ ゲートを挿入した場合は、2^{n-m} 個のセルが入れ替わる。

$\mathrm{C}^1\mathrm{NOT}(x_3; x_4)$ ゲートの挿入後、図 11 のように更に $\mathrm{C}^1\mathrm{NOT}(x_1; x_2)$ ゲートを挿入すると、$\mathrm{C}^1\mathrm{NOT}(x_3; x_4)$ ゲート挿入後の回路の x_1 のビットが正極の場合、x_2 のビットの極性が反転する。これにより、図 9 のカルノー図が図 12 のカルノー図に変更される。図 12 から分かるように、このカルノー図の 1 値のセルは 2^2 ループで囲うことができるため、論理式を最適化することができる。図 13 に、$\mathrm{C}^m\mathrm{NOT}(C';\ t')$ ゲートの挿入により最適

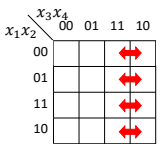

図 10: $\mathrm{C}^1\mathrm{NOT}\ (x_3;\ x_4)$ 挿入により入れ替わるセル

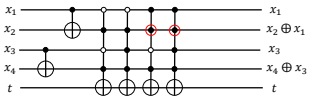

図 11: 図 7 に対する、$\mathrm{C}^1\mathrm{NOT}(x_1; x_2)$ ゲートの挿入

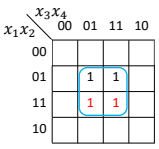

図 12: 図 11 の示す $\mathrm{C}^1\mathrm{NOT}(x_1; x_2)$ ゲート挿入後のカルノー図

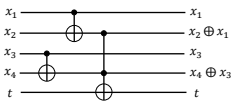

図 13: 図 12 を元に最適化した量子回路

化された論理式を実現した量子回路を示す。

しかしながら、図 13 の量子回路と、図 4 の量子回路は等価ではない。なぜなら、$\mathrm{C}^m\mathrm{NOT}(C';\ t')$ ゲートを挿入することにより、$\mathrm{C}^m\mathrm{NOT}(C';\ t')$ ゲートのターゲットビットに当たるビットの論理が変更されるからである。例えば、図 13 の回路では、x_2 にあたる論理が $x_2 \oplus x_1$ に、x_4 にあたる論理が $x_4 \oplus x_3$ に変更されている。よって、量子回路の可逆性を保つため、$\mathrm{C}^m\mathrm{NOT}(C';\ t')$ ゲートの挿入により変更された論理を元に戻す必要がある。$\mathrm{C}^m\mathrm{NOT}(C';\ t')$ ゲートの挿入により変更された論理は、挿入した $\mathrm{C}^m\mathrm{NOT}(C';\ t')$ ゲートと同じゲートを、論理関数を実現している量子ゲートの直後に挿入することで元に戻すことができる。複数の $\mathrm{C}^m\mathrm{NOT}(C';\ t')$ ゲートを挿入した場合、後に挿入した $\mathrm{C}^m\mathrm{NOT}(C';\ t')$ ゲートから順に挿入することで元に戻すことができる。図 14 に、図 4 と等価な量子回路を示す。

このように、$\mathrm{C}^m\mathrm{NOT}(C';\ t')$ ゲートを挿入することで、カルノー図の 1 値のセルを自由に移動させることができる。よって、カルノー図の 1 値のセルの初期位置がどのような状態であっても、論理式を最適化することができる。また、論理式を最適化することにより、量子回路のコストを大きく削減することができる。例えば、図 4 の回路コストは 80 であるのに対し、この手法を利用して実現した図 14 の回路コストは 9 である。

3.4 既存手法の問題点

既存手法を利用することで、既存手法を利用しない場合と比較して、大きく量子コストを削減することができる。しかしながら、既存手法はいくつか問題点を含んでいる。

まず、最小項が 2^n でない論理関数を実現する量子回

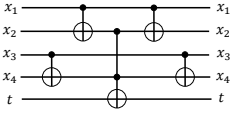

図 14: 既存手法を利用した最終的な量子回路

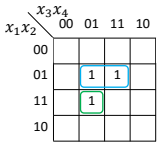

図 15: 最小項が 3 のカルノー図の一例

路を、効率的に設計することができないという問題点がある。前述した通り、ループの大きさは必ず 2^n である必要がある。よって、最小項が 2^n でない場合、1 値のセルをどのように移動させたとしても、全ての 1 値のセルを 1 つのループで囲うことができず、複数のループで囲わなければいけない。そのため、論理関数の最小項が 2^n でない場合、その論理関数を実現する量子回路の、効率的な設計ができない。

また、複数出力の量子回路を考慮していないという問題点もある。複数出力の量子回路は、1 つの出力につき 1 つの論理関数を実現する量子ゲートの集合で構成されるため、複数のカルノー図を利用する。しかし、既存手法は 1 つのカルノー図の 1 値のセルを移動させることで効率的な回路設計を行う手法である。そのため、複数出力の量子回路に対しても、単一出力の量子回路を設計する方法と同じ方法でしか量子回路設計を行うことができない。そこで本稿では、これらの問題を解決し、より効率的な量子回路を設計できる手法を提案する。

4. 複数出力を持つ量子回路のコスト削減手法

4.1 C^nNOT ゲートの挿入による回路コスト削減手法

既存手法の問題点の 1 つとして、最小項が 2^n ではない論理関数を実現する量子回路の、効率的な回路設計ができないという点があった。この問題点は、量子回路を実現する論理関数の最小項が $2^p-1(p \geqq 0)$ の場合、C^nNOT ゲートを用いることで解決できる。ある n 変数、2^p-1 最小項の論理関数を実現する量子ゲートの集合を G とした場合、G の直前、もしくは直後に $C^n\mathrm{NOT}(C;\ t)$ ゲートを 2 つを挿入する $(C = x_1...x_n, t = x_t)$。この時、挿入する $C^n\mathrm{NOT}(C;\ t)$ ゲートの論理は、G の実現する論理と異なるものである必要がある。同一の $C^n\mathrm{NOT}(C;\ t)$ ゲートを 2 つ挿入すると、1 つ目のゲートが変更したターゲットビットの論理を、2 つ目の同一のゲートが元に戻すため、最終的なターゲットビットの論理を変更することなく、$C^n\mathrm{NOT}(C;\ t)$ ゲートを挿入することができる。2 つの同一の $C^n\mathrm{NOT}(C;\ t)$ ゲートが挿入されることで、挿入された $C^n\mathrm{NOT}(C;\ t)$ ゲートの 1 つと G により、最小項が 2^n となる論理関数を実現する量子ゲートの集合を作ることができる。例として、図 15 のカルノー図を用いて説明する。このカルノー図の示す論理関数の最小項の数は 3 であり、図 15 の水色と、緑色の枠の示すループが最大のループである。よって、図 15 のカルノー図の示す論理関数は、図 16 の右の回路で表すことができる。ここで、図 16 の左の回路を構成する量子ゲート G の直前、もしくは直後に同一の $C^4\mathrm{NOT}(x_1x_2x_3x_4;\ t)$ ゲートを 2 つ挿入する。図 17 の左の回路は、図 16 の左の回路を構成する量子ゲート G の直後に、同一の $C^4\mathrm{NOT}(x_1x_2x_3x_4;\ t)$ ゲートを 2 つ挿入した回路を表している。

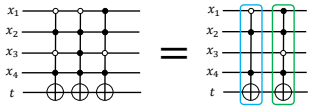

図 16: 図 15 の示す論理関数を実現する量子回路

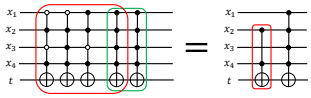

図 17: 図 16 の左の回路に対する、C^nNOT $(x_1x_2x_3x_4;\ t)$ ゲートの挿入

ここで、図 17 の赤枠で囲まれているゲートについて考える。図 17 の赤枠で囲まれているゲートの集合は、図 6 の左の回路のゲートの集合と等しいことがわかる。よって、図 6 の左の量子回路が図 6 の右の量子回路と等しいことから、図 17 の左の量子回路は、図 17 の右の量子回路と等しい。ここで、図 16 の右の回路コストは 35、図 17 の右の回路コストは 25 である。このことから、図 16 の量子回路に量子コストが 20 である $C^4\mathrm{NOT}(C;\ t)$ ゲートを 2 つ挿入したにも関わらず、最終的な量子回路のコストが削減できていることがわかる。このように、最小項の数が 2^n-1 の論理関数を実現する量子回路に対して、同一の $C^n\mathrm{NOT}(C;\ t)$ ゲートを 2 つ挿入することで、$C^n\mathrm{NOT}(C;\ t)$ ゲートを挿入していない回路と比較して、最終的な量子回路のコストを削減することができる。

4.2 CNOT ゲートを利用した論理の複製による回路コスト削減手法

既存手法の別の問題点として、複数の出力を持つ量子回路を効率的に設計することができないという問題点があった。4.1 節で提案した手法を用いても、複数出力を持つ量子回路の効率的な回路設計を行うことができない。そこで、CNOT ゲートを用いることで、量子ゲートの実現している論理を複製できるという特性を利用し、複数の出力を持つ量子回路のコストを削減する手法を提案する。

ある n 変数の論理関数を実現する、$n+1$ 入力の量子ゲートを $G(C;\ t_a)$ とする $(C = x_1...x_n,\ t_a = x_{t_a},\ a \geq 1)$。この時、$C^1\mathrm{NOT}(t_a;\ t_b)$ ゲートを、ゲート G の

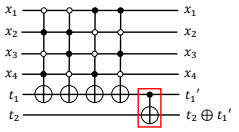

図 18: 図 4 の論理全てを複製する $\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$

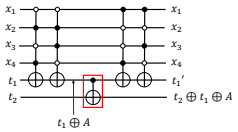

図 19: 図 4 の論理を一部複製する $\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$

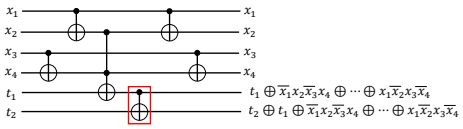

図 20: 図 14 に対する $\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートの挿入

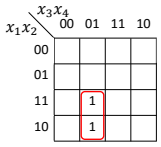

図 21: 図 23 を実現する 1 つ目のカルノー図

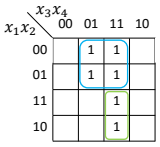

図 22: 図 23 を実現する 2 つ目のカルノー図

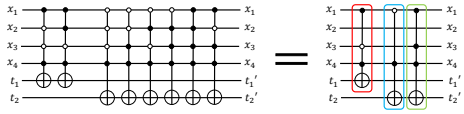

図 23: 図 21 と図 22 を同時に実現する量子回路

直後に挿入する ($a \neq b$)。つまり、ゲート G のターゲットビットにあたるビット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 CNOT ゲートを、ゲート G の直後に挿入する。すると、ゲート G の実現している論理を、CNOT ゲートのターゲットビットにあたるビットに複製することができる。また複数のゲート G の直後に $\mathrm{C}^1\mathrm{NOT}(t_a;\ t_b)$ ゲートを挿入した場合、$\mathrm{C}^1\mathrm{NOT}(t_a;\ t_b)$ より前にある全てのゲート G の実現する論理を、$\mathrm{C}^1\mathrm{NOT}(t_a;\ t_b)$ ゲートのターゲットビットにあたるビットに複製する。

例として、図 18 に図 4 の示す論理を複製する $\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ を示す。図 18 では図 4 の示す論理を全て複製しているが、$\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートを挿入する位置を変更することで、複製する論理を選択することができる。例えば、図 19 のように、図 4 の示す論理関数を実現するゲートの間に $\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートを挿入することで、$\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートより左にあるゲートによって変更された、t_1 ビットの論理を複製することができる。ただし、$\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートを図 14 のような最適化されたゲートの直後に挿入する場合、最適化されたゲートの論理のみが複製されるわけではない。$\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートにより複製されるのは、直前のゲートが示す論理ではなく、$\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートを挿入した場所の、$\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートのコントロールビットである t_1 の論理である。そのため、図 20 では、図 4 が実現する論理である $t \oplus x_2x_4\overline{x_1x_3} \oplus x_2x_3\overline{x_1x_4} \oplus x_1x_4\overline{x_2x_3} \oplus x_1x_3\overline{x_2x_4}$ が、$\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートにより複製される。

前述した CNOT ゲートを用いて論理が複製可能な性質を利用して、複数の出力を持つ量子回路のコストを削減する。例えば、図 21 と図 22 の 2 つのカルノー図の示す論理関数を実現する量子回路について説明する。図 21 と図 22 のカルノー図の示す論理関数を実現する量子回路を図 23 に示す。図 23 の左の回路は、図 21 と図 22 のカルノー図が示す論理式をそのまま実現した量子回路、右の回路は、既存手法を用いて論理式を最適化した量子回路を示している。ここで、図 21 と図 22 の 1 値のセルは、それぞれ最大であるサイズ 2^1、サイズ 2^2 のループで囲むことができる。そのため、既存手法では図 23 の右の量子回路のコストを、これ以上削減することができない。

しかしながら、CNOT ゲートを用いて論理を複製することで、図 23 の右の量子回路のコストをより削減することができる。図 24 のように、図 21 のカルノー図の示す論理関数を実現するゲートの直後に、$\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートを挿入する。すると、$\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートにより、図 21 のカルノー図の示す論理が、図 22 の論理を実現しているビットに複製される。これは、複製される論理を実現するカルノー図の 1 値のセルが、複製先のビットの論理を実現するカルノー図に複製されることを意味する。つまり、図 25 のように、複製されるカルノー図の 1 値のセルの内容が、複製先のカルノー図の同じ位置のセルに対して複製される。もし複製先のカルノー図の、複製されるセルが 1 値のセルの場合、そのセルは 0 値のセルに変更される。これは、複製先のカルノー図の 1 値のセルに当たる $\mathrm{C}^n\mathrm{NOT}$ ゲートにより変更された論理が、同じ $\mathrm{C}^n\mathrm{NOT}$ ゲートが複製されることにより、元に戻るからである。論理が複製されたことにより、図 25 のカルノー図は図 22 のカルノー図と比較して、より大きなループで囲うことができる。これにより、図 25 の示す論理関数は図 22 の示す論理関数と比較して、論理式をより最適化できるため、より量子コストの小さいゲートで量子回路を設計することができる。これにより、図 25 の示す論理関数は、$\mathrm{C}^1\mathrm{NOT}(x_4;\ t_1)$ ゲート 1 つで実現できるため、図 24 の量子回路は、図 26 の量子回路で示すことができる。図 24 の回路コストが 35 であるのに対し、図 26 の回路のコストは 17 であることから、CNOT

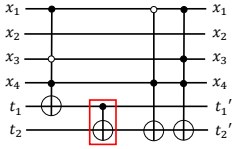

図 24: 図 23 に対する $\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートの挿入

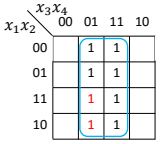

図 25: $\mathrm{C}^1\mathrm{NOT}(t_1;\ t_2)$ ゲートにより図 21 の論理が複製されたカルノー図

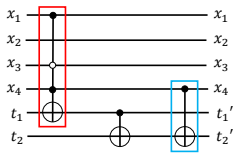

図 26: 図 25 の示す論理関数を実現する量子回路

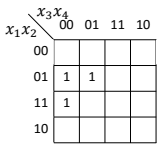

図 27: 図 29 の 1 つ目の論理関数を示すカルノー図

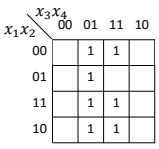

図 28: 図 29 の 2 つ目の論理関数を示すカルノー図

ゲートにより論理を複製したことで、回路コストが削減できていることがわかる。

続いて、図 27 と図 28 のカルノー図が示す論理関数を共に実現する量子回路について説明する。これらの論理関数の最小項の数はそれぞれ 3 と 7 であり、共に $2^n - 1$ である。そのため、4.1 節で提案した C^{n}NOT ゲートの挿入により、これらの論理関数を実現する量子回路はコスト削減が可能である。図 29 に C^{n}NOT ゲートの挿入により、図 27 と図 28 の示す論理関数を実現した量子回路のコストを削減したものを示す。この時、図 29 を実現する、それぞれの論理関数に対し挿入された C^{n}NOT ゲートは異なる。これは、それぞれの論理関数に対して挿入するべき、最適な C^{n}NOT ゲートが異なるためである。しかし、それぞれの論理関数を実現する量子回路に対して挿入する C^{n}NOT ゲートが等しい場合、CNOT ゲートを用いて C^{n}NOT ゲートを複製することで、回路コストを削減することができる。例として、図 27 の示す論理関数を実現する回路に対して挿入するべき最適なゲートである C^{4}NOT($x_1x_2\overline{x_3}x_4$; t) の代わりに、図 28 を実現する回路に対して最適なゲートである C^{4}NOT($\overline{x_1}x_2x_3x_4$; t) を挿入する場合を考える。それぞれの論理関数を実現する量子回路に対して、C^{4}NOT($\overline{x_1}x_2x_3x_4$; t) ゲートを挿入した量子回路を図 31 に示す。図 31 の赤枠で囲まれているゲートについて考える。赤枠で囲まれているゲートが実現している論理関数を示すカルノー図を図 30 に示す。

図 30 の 1 値のセルは、1 つのループで囲うことができない。しかし、最小項の数が 2^n である論理関数を実現

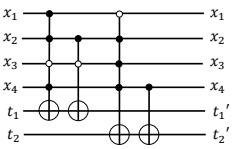

図 29: C^{n}NOT により最適化された図 27 と図 28 を実現する量子回路

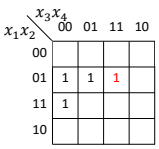

図 30: 図 31 の赤枠のゲートが実現する論理関数を示すカルノー図

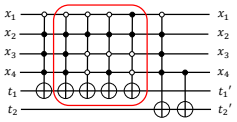

図 31: 図 28 に対して最適な C^{n}NOT ゲートをそれぞれの論理関数に対して挿入した量子回路

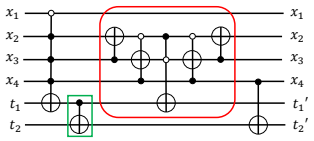

図 32: CNOT ゲートによる論理の複製と既存手法を用いて最適化された量子回路

している量子回路と考えることができるため、既存手法を適用できる。よって、図 31 の回路は、既存手法を用いて図 32 のように変形することができる。ここで、図 29 の回路のコストは 46、図 32 の回路のコストは 39 である。つまり、それぞれの論理関数にとって最適な C^{n}NOT ゲートを挿入した場合と比較して、最適ではないが互いに等しい C^{n}NOT ゲートを挿入した場合の方が、最終的な量子回路のコストが小さくなることが分かる。

このように、CNOT ゲートを用いて論理を複製することで、既存手法では考慮されていない、複数出力を持つ量子回路のコストを削減することができる。また、提案手法と既存手法を共に適用することで、複数出力を持つ最小項が 2^n でない論理関数を実現する量子回路のコストを削減することができる。つまり、既存手法のみを用いる場合と比較して、より多様な論理関数を実現する量子回路のコストを削減することができる。

5. 実験結果と考察

5.1 実行環境と評価方法

提案手法の評価を行うために、提案手法を C++で実装した。評価は、提案手法と既存手法に対して、同じ論理関数を 2 つ適用し、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。今回は、変数の数が 4、最小項の数が 2 個以上 7 個以下の 2 つの論理関数を、それぞれの組に対して 10000 通りのランダムな組み合わせを作成し、提案手法と既存手法に対してその関数を適用することで実験を行った。

前述したように、既存手法は最小項の数が 2^n である論理関数にしか適用することができない。そのため、今回の実験では 4.1 節で説明した手法と櫛田らの手法を同時に用いたものを既存手法とし、それに加えて 4.2 節で説明した手法を適用したものを提案手法とする。

表 2: 実験結果

最小項の組	既存手法	提案手法	削減率 (%)
[2,2]	32.055	32.055	0.000
[2,3]	54.346	53.494	1.568
[2,4]	35.061	35.061	0.000
[2,5]	55.148	55.148	0.000
[2,6]	51.413	49.187	4.331
[2,7]	74.809	74.809	0.000
[3,3]	76.591	58.712	23.344
[3,4]	57.269	56.943	0.570
[3,5]	77.560	58.856	24.115
[3,6]	73.688	66.796	9.353
[3,7]	96.906	76.294	21.270
[4,4]	38.035	38.018	0.044
[4,5]	58.378	58.221	0.270
[4,6]	54.361	51.663	4.962
[4,7]	77.725	58.840	24.297
[5,5]	78.437	60.707	22.604
[5,6]	74.588	72.856	2.323
[5,7]	98.084	78.180	20.293
[6,6]	70.894	57.397	19.037
[6,7]	94.180	85.092	9.649
[7,7]	117.368	84.992	27.585
合計	68.8998	60.2747	12.5184

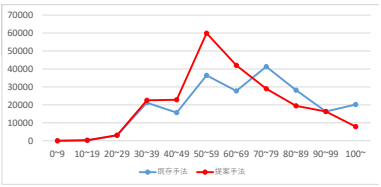

図 33: 既存手法と比較した量子コスト別の量子回路数

5.2 実験結果と考察

実験結果を表 2 に示す。表 2 の既存手法と提案手法は、最小項の組から各手法により設計された 10000 通りの量子回路の、量子コストの平均を示している。また削減率は、$(1-\frac{\text{提案手法}}{\text{既存手法}})*100$ の値を示している。すなわち、提案手法のコストが既存手法と比較して小さい場合プラスの値を示し、大きい場合マイナスの値を示す。表 2 の結果から、全ての場合で、削減率が 0 以上の値を示していることがわかる。

さらに、図 33 に既存手法と提案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場合の個数を示す。図 33 によると、量子コストが 69 以下の回路は提案手法、70 以上の回路は既存手法により多く設計されている。このことから、提案手法は既存手法と比較して、量子コストの小さい回路を多く設計できていることがわかる。これらの結果から、提案手法は既存手法と比較して、より多くの論理関数に対して効率的な量子回路設計ができることがわかる。

6. おわりに

本稿では、CNOT ゲートの挿入により論理を複製することができるという特徴を利用することで、複数の出力を持つ量子回路のコストを効率的に削減する手法を提案した。実験の結果、提案手法は最小項の数が 2 個以上 7 個以下である 4 変数の、2 つの論理関数の組み合わせに対しては、既存手法と比較して平均で約 12.5%のコストを削減することができた。また提案手法は全ての組で、既存手法と比較して量子コストの小さい量子回路を設計することができた。

今後の課題として、アルゴリズムの改善が上げられる。現在のアルゴリズムでは、特定の最小項の組を持つ論理関数しか適用することができない。そのため、不特定の論理関数の組を入力として扱うことができない。したがって、不特定の論理関数の組に対してアルゴリズムを適用させることができれば、より多くの論理関数の組の量子回路コストを削減することができると考えられる。

謝辞

本研究は JSPS 科研費 JP24106009,JP15H01677 の助成を受けたものです。

参考文献

[1] 櫛田耕平. 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. 情報処理学会関西支部支部大会講演論文集, 2015.

[2] Zahra Sasanian and D. Michael Miller. Reversible and quantum circuit optimization: A functional approach. In Reversible Computation4th International Workshop, RC 2012, Copenhagen, Denmark, July 2-3, 2012. Revised Papers, pp.112-124, 2013.

[3] 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.

[4] David Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, Vol. 400, No. 1818, pp.97-117, 1985.