推薦論文

量子回路における定数段加算器の設計

小林健了[†] 小野廣隆[††] 山下雅史[††]

本論文では量子回路上における加算の定数時間での実現について考察する．古典計算では，冗長2進表現による定数段の加算回路がすでに知られている．一方，量子計算での演算は可逆であるユニタリ変換により行われるため，古典アルゴリズムを量子計算で実行すると，古典アルゴリズムでの動作時間の多項式に比例する速度低下をともなう場合がある．本論文においては，量子回路においても，冗長2進表現を用いた加算器が定数段で設計できることを示す．

Design of an Adder Circuit with a Constant Depth on a Quantum Computer

Takenori Kobayashi,[†] Hirotaka Ono[††] and Masafumi Yamashita[††]

In this paper, we consider an adder with constant depth on a quantum circuit. By utilizing the redundant binary representation, the additional operation can be achieved in $O(1)$ on a classical computer. On the other hand, straightforward implementation of classical algorithms may cause to degrade their performance; i.e., the running time may become polynomially large, since all operations on a quantum computer must be based on (reversible) unitary transformations though irreversible operations are possible on a classical computer. We show that the addition on a quantum computer can also be done in $O(1)$ by the redundant binary representation.

1. はじめに

今日，実用的に用いられている計算機のほとんどは電子計算機である．電子計算機の高速化の中で重要な役割を担っているのは回路素子の小型化技術である．しかし，このまま小型化が進むと素子に量子力学的な問題が発生し，小型化には限界があることが指摘されている．

電子計算機とは異なる概念で動作する計算機の1つとして量子計算機がある[1),4),5)]．現在，量子計算機によって動作する量子アルゴリズムが研究されており，古典アルゴリズムより効率的に問題を解くものがある．その代表例として，1994年にShorによって発見された因数分解の多項式時間量子アルゴリズム[7)]があげられる．因数分解が多項式時間で計算できれば公開鍵暗号のシステムは重大な危機に瀕する．このことは世の中に大きな衝撃を与えた．

しかし一方，一般に古典アルゴリズムを量子回路上で行うと古典回路での実行時間の多項式に比例する速度低下を生じる可能性があるとの指摘もある[3)]．たとえば，入力 $x_1, \ldots, x_n$ に対する $x_1 \oplus \cdots \oplus x_n$ の論理式は，古典回路上では有限個の入力に対する排他的論理和であるために，1ステップで計算可能であるのに対し，量子回路では，後述のC-NOTゲートを用いると，入力の数に比例する計算時間が必要となる．このように論理式の計算において，量子回路上での計算の方が古典回路上での計算より遅くなることがある．

本論文では，定数段加算器の量子回路上での実現について考察する．内部表現に冗長2進表現を用いることで，加算は古典計算機でも入力の桁数の定数時間で計算可能である[8)]．冗長2進表現では各桁の表現に3

† 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†† 九州大学大学院システム情報科学研究院
Graduate School of Information Science and Electrical Engineering, Kyushu University

本論文の内容は2003年3月の火の国情報シンポジウム2003にて報告され，火の国情報シンポジウム2003プログラム委員会により情報処理学会論文誌への掲載が推薦された論文である．

種類の数（すなわち 1，0，−1）を用いることにより桁上げの伝搬をたかだか 1 桁におさえられる．本論文では，量子回路においても，冗長 2 進表現を用いた加算器が定数段で構成できることを示す．これは指田らの，量子回路による $O(\log n)$ 段桁上げ先見加算器[6]の改良となっている．

本論文は次のような構成となっている．2 章では量子計算と量子回路に関する準備を述べる．3 章では本手法で用いる冗長 2 進表現について述べる．4 章で実際に加算回路を設計，その計算量について考察し，5 章でまとめと今後の課題について述べる．

2. 量子計算と量子回路

まず量子計算の基本的な事項を定義し，次に加算器の設計で用いるユニタリゲート（特に Toffoli ゲート）について述べる[1),4),5)]．

1 量子ビットの 0 と 1 の状態ベクトルは $|0\rangle$ および $|1\rangle$ と表記され，それぞれ列ベクトルにより

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

と表される．このとき，1 量子ビットでの任意の状態の重ね合せ $|\psi\rangle$ は次のように表される．

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

ただし $\alpha, \beta \in \mathbf{C}$，$|\alpha|^2 + |\beta|^2 = 1$ である．$|0\rangle$ および $|1\rangle$ の係数 α，β を確率振幅と呼ぶ．状態 $|n\rangle$ の確率振幅を α_n とすると，状態 $|n\rangle$ が観測される確率は $|\alpha_n|^2$ となる．ここで，観測とは，重ね合せの状態を見ることにより，その状態がある 1 つの状態に定まる現象である．

複数の量子ビットを同時に考える場合，テンソル積☆により状態ベクトルを与える．n 量子ビットの状態空間は 2^n 次元になる（テンソル積を $|a\rangle\otimes|b\rangle$，$|a\rangle|b\rangle$ あるいは $|ab\rangle$ と表記する）．

ユニタリ変換ゲート U は，量子計算上の計算素子である．これは，逆変換が転置共役行列であるような変換である．代表的なユニタリ変換ゲートとして，NOT ゲート，Walsh-Hadamard 変換，量子 Fourier 変換（QFT）などがあげられる．ユニタリ変換ゲートを組み合わせたものを量子回路という．重ね合せの線型性より，重ね合せの状態にユニタリ変換を適用すると，重ね合わされた個別の状態に変換を適用し，再び重ね合わせた状態と結果が等しくなる．よって以下では，本論文で用いるユニタリゲートを $|0\rangle$，$|1\rangle$ に対する演算を用いて説明する．

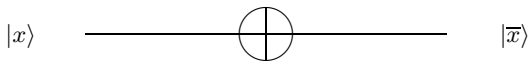

図 1 NOT ゲート
Fig. 1 NOT gate.

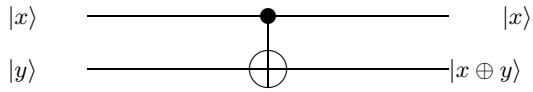

図 2 C-NOT ゲート
Fig. 2 C-NOT gate.

NOT ゲートは，量子計算で NOT，つまり

$$|x\rangle \longrightarrow |\overline{x}\rangle \qquad (x \in \{0,1\})$$

を計算するゲートである．行列表現では

$$N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

となる．また，量子回路では**図 1** のダイアグラムによって表現する．

C-NOT（Controlled-NOT）**ゲート**は，2 量子ビットの入力と出力を持つユニタリ変換ゲートである．この 2 つの量子ビットは**制御ビット** $|x\rangle$ と**目標ビット** $|y\rangle$ からなる．入力が与えられたとき，制御ビットの値が 0 のときは目標ビットには何も演算をせずに出力する．制御ビットの値が 1 のときは目標ビットの NOT を計算して出力する．いずれの場合も，制御ビットの値は不変である．つまり，入力 $|x\rangle|y\rangle$（$x, y \in \{0,1\}$）に対し，

$$|x\rangle|y\rangle \longrightarrow |x\rangle|x \oplus y\rangle$$

を計算する．量子回路では**図 2** のダイアグラムによって表現する．

C-NOT ゲートは，量子もつれ（エンタングルド）状態を作る作用をする．また，C-NOT ゲートと 1 量子ビットのユニタリ変換を組み合わせることで任意のユニタリ変換を行うことができる．

Toffoli ゲートは C-NOT ゲートの拡張として定義される．入力と出力を n 量子ビットの状態 $|x_1x_2\cdots x_{n-1}y\rangle$ とする．このとき $|x_1x_2\cdots x_{n-1}\rangle$ は $(n-1)$ 量子ビットからなる制御ビット，$|y\rangle$ は 1 量子

☆
$$|a\rangle \otimes |b\rangle = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \otimes \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 \times b_1 \\ a_1 \times b_2 \\ a_2 \times b_1 \\ a_2 \times b_2 \end{pmatrix}$$
となるような演算.

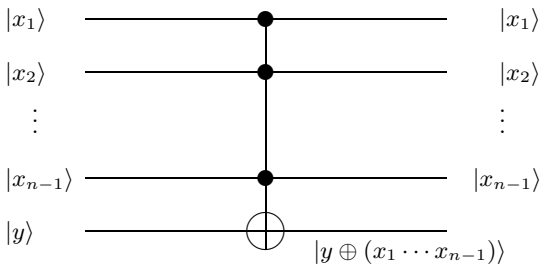

図 3 Toffoli ゲート
Fig. 3 Toffoli gate.

ビットからなる目標ビットとする．出力は，制御ビットがすべて 1 であるときのみ目標ビットに 2×2 のユニタリ変換 U を行い，そうでないときは目標ビットをそのまま出力する．制御ビットの値は不変である．つまり，$x_1, \cdots, x_{n-1}, y \in \{0,1\}$ のとき，$|x_1 x_2 \cdots x_{n-1}\rangle|y\rangle$ を次のように変換する．

$$\begin{cases} |x_1 x_2 \cdots x_{n-1}\rangle \otimes U|y\rangle & : \quad x = 11\ldots1 \\ |x_1 x_2 \cdots x_{n-1} y\rangle & : \quad \text{それ以外} \end{cases}$$

このゲートを Toffoli ゲート T_y^x と呼び，量子回路では**図 3** のダイアグラムにより表現する．ただし以下では Toffoli ゲート中の U は NOT（つまり N）として考える．$n-1$ 量子ビットの制御ビット，1 量子ビットの目標ビットを持つ Toffoli ゲートの変換行列は一般に

$$\left(\begin{array}{c|cc} I_{2^n-2} & & \\ \hline & 0 & 1 \\ & 1 & 0 \end{array}\right)$$

と表され，$T_y^x T_y^{x\dagger} = I$ であることから容易にユニタリであることが確認できる．

3. 冗長 2 進表現

本章では，高木ら[8]によって提案された，**冗長 2 進表現**およびそれを用いた加算について説明する．ただし，被加数 $x_n \cdots x_1$ および加数 $y_n \cdots y_1$ は整数とする．本論文で利用する冗長 2 進表現は，SD（Signed Digit）表現の一種である．通常の 2 進表現と同様，下位から i 番目の桁の重みは 2^{i-1} だが，それぞれの桁は $\{-1, 0, 1\}$ の要素である．冗長 2 進表現における $x_n x_{n-1} \cdots x_1$ $(x_i \in \{-1, 0, 1\})$ は，$\sum_{i=1}^{n} x_i \cdot 2^{i-1}$ という数を表す．以下，各桁の -1 を $\overline{1}$ と表す．通常の 2 進表現と同様，1 つの冗長 2 進表現によって表される数は一意に定まる．しかし逆は成り立たず，ある数を表す冗長 2 進表現は一般に複数存在する．これは通常の 2 進表現とは異なる特徴である．たとえば，5 を 4 桁の冗長 2 進表現で表すと，0101，$011\overline{1}$，$10\overline{1}\overline{1}$，$1\overline{1}01$，$1\overline{1}1\overline{1}$ の 5 通りに表すことができる．ただし，0 の表現は一意である．n 桁の冗長 2 進表現では -2^n+1 から 2^n-1 までの $2^{n+1}-1$ 個の整数を表すことができる．

表 1 各桁における桁上げの有無
Table 1 Carry-out on each digit.

被加数 x_i	加数 y_i	桁上げの有無
1	1	必ず 1 の桁上げ
1	0	下位から 1 の桁上げがあれば
0	1	1 の桁上げ
0	0	桁上げなし
$\overline{1}$	1	
1	$\overline{1}$	
$\overline{1}$	0	下位から $\overline{1}$ の桁上げがあれば
0	$\overline{1}$	$\overline{1}$ の桁上げ
$\overline{1}$	$\overline{1}$	必ず $\overline{1}$ の桁上げ

通常の 2 進表現の加算では，桁上げの伝搬があるため，桁上げ先見加算器を用いても桁数の対数に比例する計算時間がかかる．これに対し冗長 2 進表現では，その冗長性を利用し，加算において桁上げがたかだか 1 桁しか伝搬しないようにすることができることから，加算を桁数に関係なく一定時間で行うことができる．

表 1 に，冗長 2 進表現を用いて 2 進表現における加算と同様の計算を行った場合の，1 つの桁における被加数 x_i と加数 y_i の組合せに対する桁上げの有無を示す．表より，第 i 桁の被加数と加数 x_i，y_i のうち一方が 1 で他方が 0 の場合と，一方が $\overline{1}$ で他方が 0 の場合に，下位からの桁上げが上位に伝搬することがある．

x_i，y_i のうち一方が 1 で他方が 0 の場合，下位からの 1 の桁上げが上位へ伝搬する．しかし，冗長 2 進表現では，1 を 2 桁で表すと 01，$1\overline{1}$ の 2 通りに表せる．そこで下位から 1 の桁上げがあるときは，$0+1=1\overline{1}$ とし，あらかじめ上位に 1 の桁上げをすることで下位からの 1 の桁上げを吸収する．一方，下位から $\overline{1}$ の桁上げがあるときは，$0+1=1\overline{1}$ とすると今度は $\overline{1}$ の桁上げが上位に伝搬する．よって，このときは $0+1=01$ とする．下位から桁上げが起こりえない場合は 01 でも $1\overline{1}$ でもよい．この方法により，桁上げの伝搬をおさえることができる．

x_i，y_i のうち一方が $\overline{1}$ で他方が 0 の場合も同様にして，-1 が $0\overline{1}$，$\overline{1}1$ の 2 通りに表されることを利用する．1 桁下位の加数および被加数を見て，$\overline{1}$ の桁上げが起こりうる組合せのときは $0+\overline{1}=\overline{1}1$ とし，1 の桁上げが起こりうる組合せのときは $0+\overline{1}=0\overline{1}$ と

表 2 第 1 段階の計算規則
Table 2 The calculation rule in the first step.

x_i, y_i	x_{i-1}, y_{i-1}	c_i	s_i
1, 1	—	1	0
1, 0	少なくとも一方が $\overline{1}$	0	1
1, 0	両方とも $\overline{1}$ でない	1	$\overline{1}$
0, 0	—	0	0
1, $\overline{1}$	—	0	0
$\overline{1}$, 0	少なくとも一方が 1	0	$\overline{1}$
$\overline{1}$, 0	両方とも 1 でない	$\overline{1}$	1
$\overline{1}$, $\overline{1}$	—	$\overline{1}$	0

被加算数	$11000\overline{1}0\overline{1}0\overline{1}01$	(2989)
加算数	$10\overline{1}\overline{1}1\overline{1}110011$	(1395)
中間和	$011\overline{1}10100\overline{1}\overline{1}0$	
桁上げ	$10\overline{1}00\overline{1}000011$	
和	$1001\overline{1}00100000$	(4384)

図 4 冗長 2 進表現を用いた加算の例
Fig. 4 Example of addition based on the redundant binary representation.

する．

以上の方針で冗長 2 進数どうしの加算を行うと，各桁の桁上げは必ず 1 つ上位の桁で吸収され，それ以上上位に伝搬しないことが分かる．本方針に基づく加算は次の 2 段階の手順で行うことができる．第 1 段階では，各桁で $x_i + y_i = 2c_i + s_i$ となるように，桁上げ c_i および中間和 s_i を求める．c_i，s_i は，x_i，y_i，および 1 つ下位の x_{i-1}，y_{i-1} の 4 つの数から求められる．第 2 段階では，各桁で $z_i = s_i + c_{i-1}$ を計算する．z_i は s_i，c_{i-1} のみから求まり，桁上げは生じない．ここに，x_i，y_i，c_i，s_i，z_i はすべて $\{1, 0, \overline{1}\}$ の要素である．

表 2 に，第 1 段階での計算規則の例を示す．x_i，y_i のうち一方が 1 で他方が 0 の場合は，x_{i-1}，y_{i-1} のうち少なくとも一方が $\overline{1}$ ならば $c_i = 0$，$s_i = 1$ とし，どちらも $\overline{1}$ でないならば $c_i = 1$，$s_i = \overline{1}$ とする．同様に，x_i，y_i のうち一方が $\overline{1}$ で他方が 0 の場合は，x_{i-1}，y_{i-1} のうち少なくとも一方が 1 ならば $c_i = 0$，$s_i = \overline{1}$ とし，どちらも 1 でないならば $c_i = \overline{1}$，$s_i = 1$ とする．最下位桁では，下位からの桁上げがないので，2 進表現の加算と同様の計算を行う．**図 4** にこの計算規則に従った加算の例を示す．

4. 加算器の設計

名久井ら[2]によって提案された量子回路で論理関数を計算する方法を説明し，それを用いて，冗長 2 進表現を用いた量子回路による加算器の設計を行う．

4.1 量子回路による論理式の表現

加算器を量子回路上で実現するためには，対応する論理演算を量子演算（すなわちユニタリ変換）として実現する必要がある．しかし，一般に，n 変数論理関数 $f : \{0,1\}^n \to \{0,1\}$ は可逆演算ではないため，直接的には量子回路として実現できない．しかし実質的には，

$$|x\rangle|y\rangle \longrightarrow |x\rangle|y \oplus f(x)\rangle$$

により定義されるユニタリ変換 $U_{f-\mathrm{C-NOT}}$ を用いて実現できる．ここで f-C-NOT は，function-controlled-NOT を意味する．

ここで定義された **f-C-NOT 回路**の特徴は，以下のとおりである．

- $|x\rangle$ は n (≥ 1) 量子ビットで構成され，制御ビットとしてのみ用いられる．
- $|y\rangle$ は 1 量子ビットで構成され，目標ビットとしてのみ用いられる．
- 補助量子ビットを用いない．
- NOT ゲートと Toffoli ゲートで構成する．

本節では，以上の定義に基づいた f-C-NOT 回路の構成法について考える．説明の都合上，まず単純に f を実現する回路の構成について述べ，次にこれを一般化した，より効率的な回路を実現する構成について述べる．

論理変数 x_i に対し，x_i 自身またはその否定 $\bar{x}_i$ を**リテラル**，いくつかのリテラルの論理積を**積項**と呼ぶ．関数 f において 1 の値をとるベクトル（真ベクトル）に対し，変数 $x_i = 1$ であれば x_i，$x_i = 0$ であれば $\bar{x}_i$ を考え，これらに関する積項を**最小項**と呼ぶ．以下，真ベクトル v に対する最小項を $m_v(x)$ と表記することにする．論理関数はすべての最小項の（排他的）論理和の形で表現されることが知られており，f-C-NOT 回路も，最小項に対応するユニタリ変換を用いて構成できる．これを**最小項ゲート**と呼ぶ．最小項 $m_v(x)$ に対応する最小項ゲート $M_y^x(v)$ は

$$M_y^x(v) = (\prod_{v_i=0} N_{x_i}) T_y^x (\prod_{v_i=0} N_{x_i})$$

のように NOT ゲートで挟まれた 1 つの $(n+1)$ 量子ビットの Toffoli ゲートによって実現できる．ここで，N_{x_i} は量子ビット $|x_i\rangle$ の状態だけを反転する NOT ゲートを，T_y^x は n 個の制御ビット x と 1 個の目標

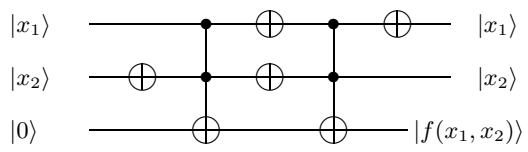

図 5 f-C-NOT 回路の例
Fig. 5 An example of f-C-NOT circuit.

ビット y を持つ Toffoli ゲートを，そして，v_i はベクトル v の変数 x_i に対応する要素を表す．最初の積 $(\prod_{v_i=0} N_{x_i})$ は，対応する最小項の論理式が 1 であるとき，目標ビットの値を反転するようにするために用いられる．また 2 番目の積 $(\prod_{v_i=0} N_{x_i})$ は，制御ビットの状態を元に戻すために用いられる．

以上の M_y^x から，f-C-NOT 回路は

$$U_{f-\mathrm{C-NOT}} = \prod_{\substack{v\in\{0,1\}^n \\ f(v)=1}} M_y^x(v)$$

のように，そのすべての最小項ゲートの積で得られる．最小項の定義からどの最小項ゲートも入力の 1 つの状態だけに対して動作するので，最小項ゲートの積の順序は可換である．最小項ゲートの定義，あるいは Toffoli ゲートの定義から，最小項ゲートの積において目標ビットを $|0\rangle$ にすることにより，$f(x)$ をその最小項の排他的論理和（EXOR）

$$f(x) = \bigoplus_{\substack{v\in\{0,1\}^n, \\ f(v)=1}} m_v(x) \qquad (1)$$

で表現することができる．論理関数 $f(x) = x_1\overline{x}_2 \oplus \overline{x}_1 x_2$ の f-C-NOT 回路を**図 5** に示す．

以上のように構成された f-C-NOT 回路は一般に入力数 n の指数サイズの深さになるため，非効率的な回路が構成される可能性がある．そこで，以下ではより効率的な（より浅い段数の）f-C-NOT 回路の構成について考察する．

ユニタリゲートとしては，最小項ではなく，単なる積項に対するゲートも考えることができる．Toffoli ゲートを NOT ゲートで挟み，積項に対応する状態を制御できるようにしたゲートを積項ゲートと呼ぶ．これは，“don’t care bit” のある最小項ゲートといえる．また逆に，最小項ゲートは入力の n 量子ビットすべてが制御ビットとなっている積項ゲートと考えることができる．この場合も積項ゲートの積は，論理表現における積項の排他的論理和（EXOR）と対応する．

これは，量子回路上で任意の積項の EXOR により表される論理式を計算することができることを意味する．論理式において，このような AND-EXOR 式は **ESOP**（Exclusive-or Sum Of Product）と呼ばれ，任意の論理式は ESOP 形式で表現されることが知られている☆．式 (1) も ESOP の 1 つである．論理関数 f を ESOP で表現したときに，積項の数が最小になる ESOP を f の**最小 ESOP**（または MESOP）と呼ぶ．

これまでの定義より，論理関数の ESOP の積項の数と積項ゲートの数は一致し，ESOP から直接 f-C-NOT 回路を構成することができる．また，積項ゲートは，NOT ゲートに挟まれた Toffoli ゲートであるので，その深さはたかだか 3 であり，積項ゲートは定数段で構成される．したがって，f-C-NOT 回路全体の深さを浅くするためには，その構成要素である積項ゲートの数を少なくすればよいことになる．すなわち，論理関数 f の最小 ESOP が深さが最小の f-C-NOT 回路を与えるため，最小段の f-C-NOT 回路の構成には最小 ESOP を発見をすればよいこととなる．

本論文では各関数に対する最小 ESOP 発見を，カルノー図を用いて行っている．

4.2 各計算段階の量子回路

本節では，冗長 2 進表現を用いた加算器を，2 段階に分けて設計する．すなわち，冗長 2 進表現の加算における 2 段階に対応する量子回路をそれぞれ設計する．

ここで，冗長 2 進表現の数を量子回路でどのように表すかという問題が生じる．本論文では，以下の対応により $\{1, 0, \overline{1}\}$ を表すものとする：

$|01\rangle := 1$
$|00\rangle := 0$
$|10\rangle := \overline{1}$

また，各変数もこれにともないそれぞれ 2 変数を対応させる．

$|x_{i,1}x_{i,2}\rangle := x_i$
$|y_{i,1}y_{i,2}\rangle := y_i$
$|c_{i,1}c_{i,2}\rangle := c_i$
$|s_{i,1}s_{i,2}\rangle := s_i$
$|z_{i,1}z_{i,2}\rangle := z_i$

ただし，$x_{i,j}, y_{i,j}, c_{i,j}, s_{i,j}, z_{i,j} \in \{0,1\}$ $(j \in \{1,2\})$ とする．

冗長 2 進表現の 1 桁を 2 量子ビットで表すことと f-C-NOT 回路を用いることを考慮して，これから設計する量子回路の各段階で行う計算を示す．量子回路全体への入力は

$$|x_{n,1}x_{n,2}\cdots x_{1,1}x_{1,2}\rangle|y_{n,1}y_{n,2}\cdots y_{1,1}y_{1,2}\rangle|0^{6n-2}\rangle$$

☆ ESOP 表現は，リード・マラー（Reed-Muller）表現ともいう．

とする．ここで，最後の $|0^{6n-2}\rangle$ は，第 1 段階で得られる状態 $c_{i,j}$，$s_{i,j}$ や，第 2 段階での出力状態 z_{i_j} に対応する量子ビットである．

まず，第 1 段階の計算により，桁上げ $|c_{i,1}c_{i,2}\rangle$ および中間和 $|s_{i,1}s_{i,2}\rangle$ を求める．第 2 段階で，第 1 段階で得られた $|c_{n,1}c_{n,2}s_{n,1}s_{n,2}\cdots c_{1,1}c_{1,2}s_{1,1}s_{1,2}\rangle$ を用いて，冗長 2 進表現で表された 2 整数の和の各桁の数 $|z_{i,1}z_{i,2}\rangle$ を求める．ここで得られた状態 $|c_{n,1}c_{n,2}z_{n,1}z_{n,2}\cdots z_{2,1}z_{2,2}s_{1,1}s_{1,2}\rangle$ を冗長 2 進表現に変換すると，冗長 2 進表現の 2 整数の和が求められる．

この方法を用いると，n 桁の冗長 2 進表現の 2 整数の和を求める量子回路を設計するのに $(10n-2)$ 量子ビットを要する．

第 1 段階の計算を行う量子回路を与える．桁上げ $|c_{i,1}c_{i,2}\rangle$ および中間和 $|s_{i,1}s_{i,2}\rangle$ を求める際，$i \geq 2$ の場合，$|x_{i,1}x_{i,2}\rangle$，$|y_{i,1}y_{i,2}\rangle$ だけでなく $|x_{i-1,1}x_{i-1,2}\rangle$，$|y_{i-1,1}y_{i-1,2}\rangle$ も必要になる．しかし，$|c_{1,1}c_{1,2}\rangle$，$|s_{1,1}s_{1,2}\rangle$ を求める場合は $|x_{1,1}x_{1,2}\rangle$，$|y_{1,1}y_{1,2}\rangle$ だけが分かればよい．よって，本論文では $|c_{1,1}c_{1,2}\rangle$，$|s_{1,1}s_{1,2}\rangle$ を求める量子回路と $|c_{i,1}c_{i,2}\rangle$，$|s_{i,1}s_{i,2}\rangle$ $(i \geq 2)$ を求める量子回路に分けて設計する．

最下位桁では，通常の 2 進表現と同じ方法で加算を行う．また，$x_{1,1} = x_{1,2} = 1$ あるいは $y_{1,1} = y_{1,2} = 1$ という入力は与えられず．これらの場合 “don’t care” である．以上を考慮して，以下の最小 ESOP が得られる．

$$
\begin{aligned}
c_{1,1} &= x_{1,1}y_{1,1}\\
c_{1,2} &= x_{1,2}y_{1,2}\\
s_{1,1} &= x_{1,1}\overline{y}_{1,2} \oplus \overline{x}_{1,2}y_{1,1}\\
s_{1,2} &= \overline{x}_{1,1}y_{1,2} \oplus x_{1,2}\overline{y}_{1,1}
\end{aligned}
$$

次に，$i \geq 2$ において $|c_{i,1}c_{i,2}\rangle$，$|s_{i,1}s_{i,2}\rangle$ を求める量子回路を設計する．この場合，$|x_{i,1}x_{i,2}\rangle$，$|y_{i,1}y_{i,2}\rangle$ だけでなく，$|x_{i-1,1}x_{i-1,2}\rangle$，$|y_{i-1,1}y_{i-1,2}\rangle$ の情報も必要になる．冗長 2 進表現の加算の第 1 段階の計算規則により，次の最小 ESOP が得られる．

$$
\begin{aligned}
c_{i,1} &= x_{i,1}y_{i,1} \oplus f_1 \oplus f_2\\
c_{i,2} &= x_{i,2}y_{i,2} \oplus f_3 \oplus f_4\\
s_{i,1} &= x_{i,1}\overline{y}_{i,2} \oplus \overline{x}_{i,2}y_{i,1} \oplus f_1 \oplus f_2 \oplus f_3 \oplus f_4\\
s_{i,2} &= \overline{x}_{i,1}y_{i,2} \oplus x_{i,2}\overline{y}_{i,1} \oplus f_1 \oplus f_2 \oplus f_3 \oplus f_4
\end{aligned}
$$

(ただし
$$
\begin{aligned}
f_1 &= \overline{x}_{i,1}\overline{x}_{i,2}\overline{x}_{i-1,2}y_{i,1}\overline{y}_{i-1,2}\\
f_2 &= x_{i,1}\overline{x}_{i-1,2}\overline{y}_{i,1}\overline{y}_{i,2}\overline{y}_{i-1,2}\\
f_3 &= \overline{x}_{i,1}\overline{x}_{i,2}\overline{x}_{i-1,1}y_{i,2}\overline{y}_{i-1,1}\\
f_4 &= x_{i,2}\overline{x}_{i-1,1}\overline{y}_{i,1}\overline{y}_{i,2}\overline{y}_{i-1,1})
\end{aligned}
$$

次に，これまで得られた最小 ESOP から，深さ最小の量子回路を構成する．

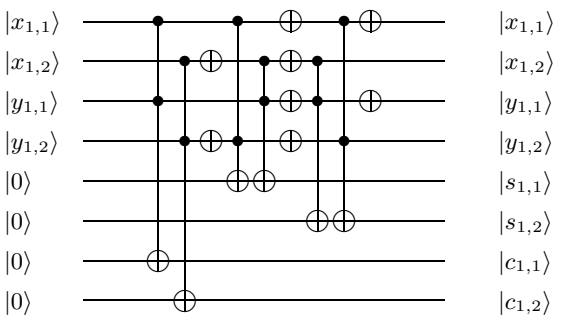

図 6 量子回路 CS_1
Fig. 6 Quantum circuit CS_1.

まず，$c_{1,1}$，$c_{1,2}$，$s_{1,1}$，$s_{1,2}$ を求める最小 ESOP から深さ最小の量子回路を構成する．これらの最小 ESOP について，この中のどの 2 つに対しても，共通の積項は存在しない．よって，それぞれの変数に対して独立に量子回路を構成し，それを組み合わせる．

入力 $|x_{1,1}x_{1,2}\rangle$，$|y_{1,1}y_{1,2}\rangle$ に対し，$|c_{1,1}c_{1,2}\rangle$，$|s_{1,1}s_{1,2}\rangle$ を出力する量子回路 CS_1 を**図 6** に示す．

次に，$i \geq 2$ における桁上げおよび中間和 $c_{i,1}$，$c_{i,2}$，$s_{i,1}$，$s_{i,2}$ を求める最小 ESOP から，深さ最小の量子回路を構成する．

$c_{i,1}$，$c_{i,2}$，$s_{i,1}$，$s_{i,2}$ の最小 ESOP より，$c_{i,1}$，$s_{i,1}$，$s_{i,2}$ および $c_{i,2}$，$s_{i,1}$，$s_{i,2}$ はそれぞれ共通の積項を含んでいる．いくつかの論理式が共通の積項を含む場合，個別にその積項を計算すると NOT ゲートが論理式の数だけ必要となる．しかし，共通する積項をまとめて計算すると必要な NOT ゲートの数を定数個にすることができる．また，NOT ゲート N の性質より，

$$
N^2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I
$$

であるため，隣接する Toffoli ゲートの間にある NOT ゲートの数は，たかだか 1 個になる．

このことを利用して，入力 $|x_{i,1}x_{i,2}\rangle$，$|y_{i,1}y_{i,2}\rangle$，$|x_{i-1,1}x_{i-1,2}\rangle$，$|y_{i-1,1}y_{i-1,2}\rangle$ に対し，$|c_{i,1}c_{i,2}\rangle$，$|s_{i,1}s_{i,2}\rangle$ を出力する量子回路 CS_i $(i \geq 2)$ を**図 7** に示す．冗長 2 進表現の加算の第 1 段階の計算を行う量子回路は i に依存しない段数，つまり $O(1)$ 段で構成できる．

次に，第 1 段階の計算により $c_n, s_n, \cdots, c_1, s_n$ $(c_i, s_i \in \{1, 0, \overline{1}\})$ が求められたとき，第 2 段階の計算を行う量子回路を与える．

s_i, c_{i-1} から第 i $(i \geq 2)$ 桁の計算結果 z_i を求める段階では，上位への桁上げは起こらない．よって，z_i

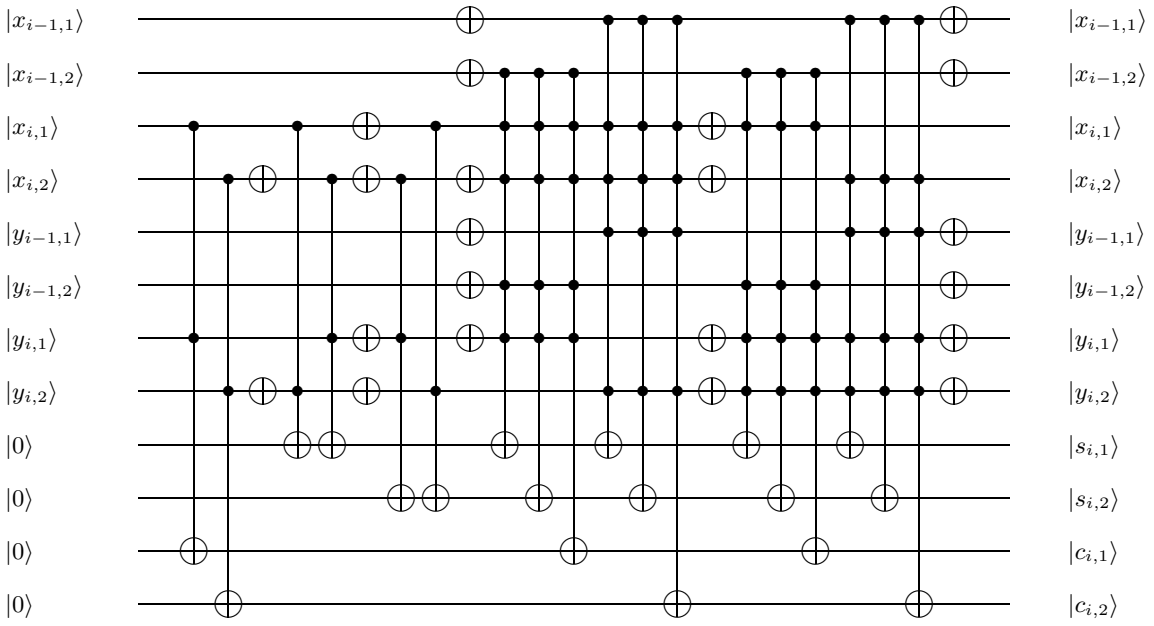

図 7 量子回路 CS_i
Fig. 7 Quantum circuit CS_i.

を求めるためには，s_i，c_{i-1} が分かれば十分である．この段階では桁上げが発生しないので，通常の 2 進表現と同じ方法で加算を行う．それより，以下のような最小 ESOP が得られる．

$$z_{i,1} = c_{i-i,1}\overline{s}_{i,2} \oplus \overline{c}_{i-1,2}s_{i,1}$$
$$z_{i,2} = \overline{c}_{i-i,1}s_{i,2} \oplus c_{i-1,2}\overline{s}_{i,1}$$

ここで得られた $z_{i,1}$，$z_{i,2}$ の最小 ESOP より，$z_{i,1}$，$z_{i,2}$ を計算する深さ最小の量子回路を構成する．$z_{i,1}$，$z_{i,2}$ は，第 1 段階における $s_{1,1}$，$s_{1,2}$ の最小 ESOP と同じ形をしているので，その量子回路を用いることにより，入力 $|c_{i-1,1}c_{i-1,2}\rangle$，$|s_{i,1}s_{i,2}\rangle$ に対し，$|z_{i,1}z_{i,2}\rangle$ を出力する量子回路 Z_i を**図 8** に示す．冗長 2 進表現の加算の第 2 段階の計算を行う量子回路は i に依存しない段数，つまり $O(1)$ 段で構成できる．

4.3 時間計算量

前節で設計した加算器の時間計算量について論じる．

第 1 段階の時間計算量を考察する．入力の冗長 2 進表現の 2 整数から第 i 桁の桁上げ c_i および中間和 s_i を求める量子回路は，第 i 桁のほかに，第 $(i-1)$ 桁さえ分かれば計算することができる．これは，第 i 桁の桁上げと中間和を求める計算は，隣の桁，すなわち，第 $(i+1)$ 桁および第 $(i-1)$ 桁以外の中間和および桁上げを求める計算と，量子回路により並列に行うことができるということを意味している．

このことを利用して，第 1 段階の計算を入力長に依存しない段数で行う量子回路の構成法を示す．

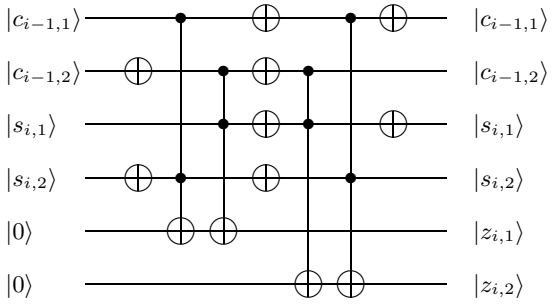

図 8 量子回路 Z_i
Fig. 8 Quantum circuit Z_i.

(1) 奇数桁の桁上げ c_{2i-1} および中間和 c_{2i-1} を量子回路 CS_{2i-1} により計算する．
(2) 偶数桁の桁上げ c_{2i} および中間和 c_{2i} を量子回路 CS_{2i} により計算する．

この構成法により，第 1 段階の計算が定数段のユニタリゲートで行えることを示す．

まず，奇数桁の桁上げおよび中間和 $|c_{2i-1,1}c_{2i-1,2}\rangle$，$|s_{2i-1,1}s_{2i-1,2}\rangle$ を，量子回路 $CS_1, CS_3, \cdots, CS_{2i-1}, \cdots$ により求める．このとき，任意の異なる i, j $(2i-1, 2j-1 \leq n)$ について，CS_{2i-1} と CS_{2j-1} は同じ量子ビットを入力としないので，この計算を並列に行

うことができる．また，CS_{2i-1} の回路の深さについては，$i \geq 2$ については，i および入力長 n の値に関係なく一定の深さで構成できる．また，CS_1 は，ほかの桁上げや中間和を計算する回路よりも少ない段数で構成することができる．よって，奇数桁の桁上げおよび中間和を求める計算は，入力長に関係なく，定数時間で行うことができる．

次に，偶数桁の桁上げおよび中間和 $|c_{2i,1}c_{2i,2}\rangle$，$|s_{2i,1}s_{2i,2}\rangle$ を求める計算に要する時間についても，奇数桁の場合と同様に考えることができる．偶数桁の桁上げおよび中間和を，量子回路 CS_2，$CS_4, \cdots, CS_{2i}, \cdots$ により求める．このとき，任意の異なる i，j $(2i, 2j \leq n)$ について，CS_{2i} と CS_{2j} は同じ量子ビットを入力としないので，この計算を並列に行うことができる．また，CS_{2i} の回路の深さについては，i および入力長 n の値に関係なく一定の深さで構成できる．よって，偶数桁の桁上げおよび中間和を求める計算は，入力長に関係なく，定数時間で行うことができる．

以上より，第 1 段階では CS_i 2 段分の計算時間を要することが分かる．CS_i は定数段で構成可能なので，第 1 段階の計算時間は $O(1)$ であることが示された．

第 2 段階の時間計算量を考察する．第 2 段階の計算は，第 $(i-1)$ 桁の桁上げ c_{i-1} と第 i 桁の中間和 s_i さえ分かっていれば行うことができる．これは，第 2 段階の計算では上位への桁上げが起こらないためである．よって，第 2 段階ではすべての z_i を求める計算を量子回路により並列に行うことができる．

z_i を求める量子回路 Z_i は定数段で構成できるので，第 2 段階の時間計算量を $O(1)$ にするためにはすべての z_i を並列に計算すれば十分である．

以上の結果より，第 1 段階の時間計算量は $O(1)$ であり，第 2 段階の時間計算量は $O(1)$ であるため，加算器全体の時間計算量は $O(1)$ となる．

5. おわりに

本論文では，冗長 2 進表現を用いた量子回路による加算器の設計について考察を行った．これにより古典回路と同様，量子回路においても冗長 2 進表現を用いた加算を定数時間で計算することができることが示された．乗算，除算など，多くの加算を連続して行う演算において，本論文で提案した量子回路を採用することで，演算の終了後に冗長 2 進表現から通常の 2 進表現への変換に $O(\log n)$ 時間要するとしても，全体としては演算時間の大幅な短縮が期待できる．

本論文では，冗長 2 進表現の加算器を，f-C-NOT 回路を用いて設計した．しかし一般には f-C-NOT 回路により構成された量子回路が，深さ最小の量子回路になるとは限らない．本論文で設計した加算器の量子回路が，f-C-NOT 回路に関する制約をなくすことでさらに簡単化できるかどうかの考察が今後の課題としてあげられる．

謝辞 本研究の一部は，科学研究費の補助を受けて行われた．

参考文献

1) 細谷暁夫：量子コンピュータの基礎，サイエンス社 (1999).
2) 名久井行秀，西野哲朗：ある制約の下での量子論理回路の最小化について，2002 夏の LA シンポジウム予稿 (2002).
3) 西野哲朗：量子計算理論，2001 冬の LA シンポジウム予稿 (2001).
4) 西野哲朗：量子コンピュータ入門，東京電機大学出版局 (1997).
5) 大矢雅則：量子コンピュータの数理，丸善株式会社 (1999).
6) 指田真宏，西野哲朗：量子フーリエ状態を並列にコピーする量子回路の設計，2002 夏の LA シンポジウム予稿 (2002).
7) Shor, P.W.: Polynomial-time algorithm for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comput.*, Vol.26, No.5, pp.1484–1509 (1997).
8) 高木直史，安浦寛人，矢島脩三：冗長 2 進加算木を用いた VLSI 向き高速乗算器，電子通信学会論文誌，Vol.J66-D, No.6, pp.683–690 (1983).

(平成 15 年 10 月 1 日受付)
(平成 16 年 3 月 5 日採録)

推薦文

現在の電子計算機とは異なる概念で動作する計算機の 1 つに量子計算機があり，これによる計算の高速化が期待されている．しかし，古典アルゴリズムを量子回路上で行うと，古典回路での実行時間よりも逆に遅くなるという可能性もある．本論文では，冗長 2 進表現を用いた加算が古典的アルゴリズムにおいて定数時間で実行可能であるという結果に基づき，その動作を模倣する量子回路が定数段で構成可能であることを示している．量子計算自体がまだ実用的な段階の研究ではないため，実用的であるとはいい難い．しかし，理論的にみれば，冗長 2 進表現を用いた加算が量子回路で実現可能であることを示すという興味深い内容であり，また，構成された量子回路の正当性などについては問題がないと判断されるため，本論文を推薦するも

のである．

（火の国情報シンポジウム 2003 プログラム委員長 近藤 弘樹）

小林　健了

平成 15 年九州大学工学部電気情報工学科卒業．同年同大学大学院システム情報科学府情報工学専攻修士課程入学，現在に至る．量子計算の研究に従事．

小野　廣隆（正会員）

平成 14 年京都大学大学院情報工学研究科数理工学専攻博士課程修了．博士（情報学）．同年，九州大学システム情報科学研究院助手．組合せ最適化アルゴリズムの研究に従事．

山下　雅史（正会員）

昭和 55 年名古屋大学大学院工学研究科博士後期課程修了．工学博士．豊橋技術科学大学，広島大学を経て，平成 10 より九州大学大学院システム情報科学研究院教授．アルゴリズムの研究に従事．