基般

量子回路と古典回路の相違：加算回路を例として

高橋康博（日本電信電話（株）NTT コミュニケーション科学基礎研究所）

回路計算モデル

回路というと，複雑で具体的なハードウェアを想像される方も多いと思われるが，本稿で扱うのはアルゴリズムを記述する単純で抽象的な計算モデルである．理想的に動く基本的な部品（基本ゲート）が与えられたと想定し，これらを紙の上でも想像の上でも自由に組み合わせ，入力に対して出力を行う仕組みが構成できれば，それを回路と呼ぼうというのである．量子コンピュータは量子力学に基づくコンピュータであり，その上のアルゴリズムを記述するのが量子回路である．一方で，我々の身の回りにあるコンピュータは古典力学に基づくことから，ここでは，その上のアルゴリズムを記述する回路を古典回路と呼ぶ．

本稿では，加算回路の構成を通して量子回路と古典回路の相違を紹介する．前半では，回路を構成する部品やその性質についての相違を述べる．そして後半では，2 つの自然数の加算を計算する単純なアルゴリズムを基に量子回路と古典回路を構成し，それらを比較する．また，量子回路特有の部品を使う加算回路を紹介する．自然数の加算という馴染み深い計算を通して，量子コンピュータの仕組みを感じていただこうというのが本稿の狙いである．

古典回路

我々の身の回りには数多くのコンピュータが存在しており，大きさや性能という観点から見ると実に多種多様である．しかし，それらの計算の仕組みは同一とみなすことができ，古典回路と呼ばれる計算モデルにより表現できる．古典回路は，基本ゲートと呼ばれる部品を組み合わせ，入力を受け付ける部分と出力を行う部分を指定したものと考えれば良い．基本ゲートにはいくつか種類があり，それぞれ短時間で実行可能な古典演算（ビット列の間の対応関係）に対応している．古典回路に入力が与えられると，組み合わせた基本ゲートの種類と組合せ方に従って入力が処理され，出力が得られるという仕組みである．

基本ゲートに対応する演算として否定演算 NOT，論理積演算 AND，論理和演算 OR を考える．NOT は 1 ビット入力 1 ビット出力であり，AND は 2 ビット入力 1 ビット出力である．入出力関係は次の通りである：

$$\mathrm{NOT}(x)=\begin{cases}1 & x=0 \text{ のとき}\\ 0 & x=1 \text{ のとき}\end{cases}$$

$$\mathrm{AND}(x,y)=\begin{cases}1 & (x,y)=(1,1) \text{ のとき}\\ 0 & \text{それ以外のとき}\end{cases}$$

AND と同様に OR も 2 ビット入力 1 ビット出力であり，OR(x, y) は $(x, y) = (0, 0)$ のときのみ 0 である．これらの演算に対応する基本ゲートをそれぞれ NOT ゲート，AND ゲート，OR ゲートと呼ぶこととする．古典回路において，入力や各ゲートの出力はコピーでき，ほかの複数のゲートの入力として利用できると仮定するのが一般的である．ここでは簡単のため，コピーを 1 個作る操作を基本的な操作として許し，複数のコピーが必要な場合にはこれを繰り返す．

古典回路の例を**図 -1** に示す．計算は左から右に向けて進むこととする．図中の x, y は回路の入力であり，この場所が入力を受け付ける部分である．また，PAR(x, y) は回路の出力であり，この場所が出力を行う部分である．x は初めにコピーされ，NOT ゲートと AND ゲートの入力となる．y も同様に処理される．x が入力された NOT ゲート

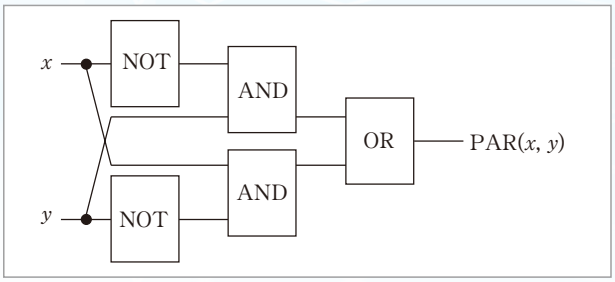

図-1 演算 PAR を実行する古典回路．PAR(x, y) は x と y のパリティを表し，これは排他的論理和と同じである

の出力は NOT(x) であり，これが次の AND ゲートの入力となっている．この AND ゲートの出力は AND (NOT(x), y) となる．各ゲートの定義に従って計算すると，回路の出力は次の演算 PAR の出力と一致することが確認できる：

$$\mathrm{PAR}(x,y)=\begin{cases}1 & (x,y)=(0,1)\text{ または }(1,0)\text{のとき}\\ 0 & \text{それ以外のとき}\end{cases}$$

したがって，図-1 の回路は演算 PAR を実行する古典回路である．PAR(x, y) は x と y のパリティ（$x+y$ の偶奇を表す値）と呼ばれ，以下では $x \oplus y$ と記述する．これは x と y の排他的論理和と同じである．また，以下では 3 ビットのパリティ $x \oplus y \oplus z$ ＝PAR (PAR $(x, y), z$) も扱う．任意の古典演算は，NOT ゲート，AND ゲート，OR ゲートだけからなる古典回路で実行できることが知られており，この意味でこれらのゲートの組は万能であるといわれる．

演算の可逆性

可逆性は，量子演算（量子状態の間の対応関係）には要請されるが，古典演算には一般に要請されない☆1 性質である．演算が可逆であるとは逆演算が存在することであり，演算の出力に対しその入力が一意に定まることと考えれば良い．たとえば NOT は可逆である．それは，出力が 1 の場合には入力が 0 と定まり，出力が 0 の場合には入力が 1 と定まるからである．一方，AND の出力が 0 の場合，入力に 3 通りの可能性が考えられるため，AND は可逆ではない．

☆1 可逆な古典演算からなる回路を対象とする研究分野があり，低消費電力回路等への応用が期待されている．

本稿の後半では古典加算回路と量子加算回路，すなわち加算演算を実行する古典回路と量子回路を扱う．n ビットの加算演算とは，それぞれ n ビットで表現される 2 つの自然数を入力とし，$n+1$ ビットで表現されるその和を出力とする演算 ADD のこととする．この ADD も可逆ではない．簡単にいえば，2 つの自然数の和が 5 であるといわれても，元の 2 つの自然数が定まらないからである．

量子演算を考える際に，たとえば a 量子ビットの状態を入力とし，b 量子ビットの状態を出力とする任意の演算を想像してはいけない．本特集の西村氏の記事にあるように，a 量子ビット入力の量子演算は，複素数を成分とする 2^a 次ユニタリ行列，すなわち 2^a 行 2^a 列のユニタリ行列で表現されるという量子力学の規約があるからである．したがって，入力が a 量子ビットの状態の場合，出力も a 量子ビットの状態である．そして，ユニタリ行列は逆演算を持つので量子演算は可逆である．

量子演算の可逆性により，可逆でない古典演算の入出力関係は量子演算として直接実現できない．したがって，たとえば AND(x, y) を量子演算の出力としたい場合，次の入出力関係を持つ 3 量子ビット入力の量子演算 QAND を実現する：

$$\mathrm{QAND}|x\rangle\,|y\rangle\,|0\rangle = |x\rangle\,|y\rangle\,|\mathrm{AND}(x,y)\rangle.$$

AND(x, y) だけでなく，x, y も出力とすることで，可逆性を保証するのである．この演算は Toffoli 演算とも呼ばれる．x, y を入力として AND(x, y) を出力とする 2 量子ビット入力の量子演算が存在しないことは証明できるため，$|0\rangle$ に初期化された補助的な量子ビットを 3 量子ビット目に付加することは必須である．このような量子ビットは後で述べる量子加算回路の構成に影響する．

ADD に対しては，次のような量子演算 QADD を考える：

$$\mathrm{QADD}|x\rangle\,|y\rangle\,|0\rangle = |x\rangle\,|\mathrm{ADD}(x,y)\rangle.$$

この場合 x, y はそれぞれ n ビットであり，ADD(x, y) は $n+1$ ビットで表現されている．QAND と同様に

y を出力とする形も考えられるが，簡単のため前述の形を考える．QADD が可逆であることは，たとえば 2 つの自然数の和が 5 で，元の自然数の一方が 2 であるといわれれば，もう一方が 3 と定まることに対応する．量子演算の可逆性により，$2n$ ビット入力の ADD に対し，$2n+1$ 量子ビット入力の QADD を考えなければならないのである．

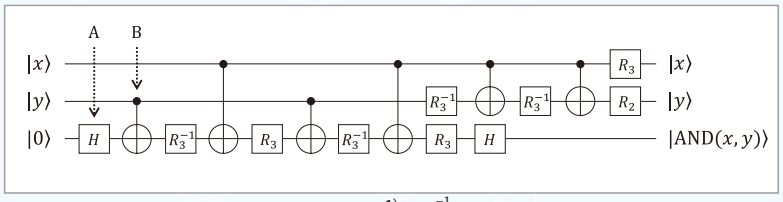

図 -2　演算 QAND を実行する量子回路 [1]．R_3^{-1} は R_3 の逆演算である．また，黒丸と⊕を結んだゲートは CNOT ゲートであり，黒丸は CNOT の制御量子ビットに対応する

量子回路

古典回路と同様に，量子回路は基本ゲートを組み合わせたものであり，基本ゲートには量子コンピュータにおいて短時間で実行可能な量子演算が対応する．このような演算として，ここでは 1 量子ビットユニタリ演算と CNOT (Controlled-NOT) 演算を考え，対応する基本ゲートをそれぞれ 1 量子ビットゲート，CNOT ゲートと呼ぶこととする．これらの演算は本特集の西村氏の記事において解説されているが，特に CNOT の入出力関係は次の通りである：

$$\mathrm{CNOT}|x\rangle\,|y\rangle = |x\rangle\,|x \oplus y\rangle.$$

CNOT は，1 つの量子ビットの状態 $|x\rangle$ が $|1\rangle$ の場合に，もう 1 つの量子ビットの上で NOT を実行する演算であり，$|x\rangle$ によって制御された NOT とみなすことができる．制御に使われる量子ビットは制御量子ビット，NOT が実行される量子ビットは標的量子ビットと呼ばれる．上で述べた QAND も同様に，2 つの量子ビットの状態 $|x\rangle\,|y\rangle$ によって制御された NOT とみなすことができる．したがって，QAND は CNOT の拡張であり，2 つの制御量子ビットと 1 つの標的量子ビットを持つ演算と捉えられる．

1 量子ビットユニタリ演算は無限種類（実数と 1 対 1 対応する程度）存在する．1 ビット入力 1 ビット出力の古典演算は，恒等演算，否定演算，2 種類の定数出力演算の合計 4 種類しか存在しないので，演算の種類だけを比較しても量子演算と古典演算には大きな違いがある．たとえば，次の入出力関係を持つアダマール演算 H と位相シフト演算 R_k は 1 量子ビットユニタリ演算である：

$$H|x\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{(-1)^x}{\sqrt{2}}|1\rangle,\ R_k|x\rangle = e^{2\pi i x/2^k}|x\rangle.$$

ただし，k は任意の自然数である．

量子回路の例を**図 -2** に示す．1 つの横線は 1 つの量子ビットに対応し，計算は左から右に向けて進む．図 -2 の一番上の横線に対応する量子ビットを 1 量子ビット目，その下の量子ビットを 2 量子ビット目，一番下の量子ビットを 3 量子ビット目と呼ぶこととする．$|x\rangle\,|y\rangle\,|0\rangle$ は回路の入力であり，初めに $|0\rangle$ が H ゲート（図 -2 の中の A で示したゲート）の入力となる．したがって状態は次のようになる：

$$|x\rangle|y\rangle\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}|x\rangle|y\rangle|0\rangle + \frac{1}{\sqrt{2}}|x\rangle|y\rangle|1\rangle.$$

次に 2 量子ビット目と 3 量子ビット目の状態が CNOT ゲート（図 -2 の中の B で示したゲート）の入力となる．状態は次のようになる：

$$\frac{1}{\sqrt{2}}|x\rangle|y\rangle|y\rangle + \frac{1}{\sqrt{2}}|x\rangle|y\rangle|y\oplus 1\rangle$$
$$= |x\rangle|y\rangle\left(\frac{1}{\sqrt{2}}|y\rangle + \frac{1}{\sqrt{2}}|y\oplus 1\rangle\right).$$

この段階で，1 量子ビット目の状態はどのゲートの入力にもなっていないので，その状態は $|x\rangle$ から変化していない．また，2 量子ビット目は制御量子ビットとして使われただけなので，同様に，その状態は $|y\rangle$ から変化していない．各ゲートの定義に従って計算を進めると，回路の出力は $|x\rangle|y\rangle|\mathrm{AND}(x, y)\rangle$ となることが確認できる．したがって，図 -2 の回路は演算 QAND を実行する量子回路である [1]．任意の量子演算は，1 量子ビットゲートと CNOT ゲートだけからなる量子回路で実行できることが知られており，この意味でこれらのゲートの組は万能である

といわれる．

任意の古典回路の（何らかの形で可逆化した）入出力関係は量子回路で模倣できる．たとえば，模倣したい古典回路の中にコピーを作る操作があれば，CNOT$|x\rangle|0\rangle = |x\rangle|x\rangle$ というようにCNOTで模倣すれば良い．また，ANDゲートがあれば図-2の量子回路で模倣する．ただし，このような模倣には$|0\rangle$に初期化された補助的な量子ビットが必要となることに注意する．回路の計算コストとは，その回路に含まれる基本ゲートの個数とし，古典回路においてはコピーの回数も計算コストに加えることとする．何らかの問題を解く古典回路に対し，それを模倣する量子回路が上で述べた要領で構成でき，その計算コストは元の古典回路の高々定数倍程度になることが証明できる．したがって，大雑把にいえば，現在のコンピュータで可能な計算は，量子コンピュータでも同程度の時間で可能である．

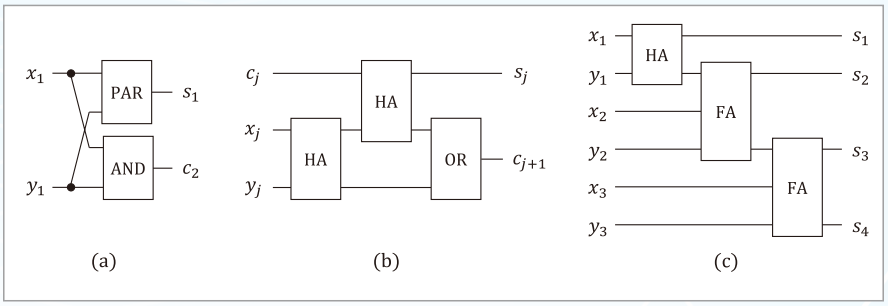

図-3　(a) 半加算器HA．PARは図-1の回路である．(b) 全加算器FA．(c) 桁上げ伝播法に基づく古典加算回路

加算回路

桁上げ伝播法に基づく古典加算回路

2つの自然数の加算アルゴリズムとして思い浮かぶのは，下位の桁から順に，その桁の和の計算とその桁からの桁上げの有無の判定を繰り返すというものであろう．このアルゴリズムは桁上げ伝播法と呼ばれる．nビットで表現される2つの自然数x, yを入力とし，その表現をそれぞれ$x_n \cdots x_1, y_n \cdots y_1$（$x_1, y_1$が最下位桁）とする．各桁からの桁上げの有無を表すビット$c_j\,(1 \le j \le n+1)$を次のように定義する：$c_1 = 0$とし，任意の$1 \le j \le n$に対し，

$$c_{j+1} = \mathrm{AND}(x_j, y_j) \oplus \mathrm{AND}(y_j, c_j) \oplus \mathrm{AND}(c_j, x_j).$$

c_{j+1}を定義する際，上のように3ビットのパリティとするのではなく，3ビットのORとするのが一般的であろうが，これらの定義は同値であり，上の定義が量子加算回路の議論を簡単にする．各桁の和を表すビット$s_j\,(1 \le j \le n+1)$を次のように定義する：任意の$1 \le j \le n$に対し，$s_j = x_j \oplus y_j \oplus c_j$とし，$s_{n+1} = c_{n+1}$．このように定義すると$s_{n+1} \cdots s_1$は$x+y$の$n+1$ビット表現であることが証明される．桁上げ伝播法は，c_1からc_{n+1}まで順にc_jを計算し，それを基に各s_jを出力するアルゴリズムといえる．

桁上げ伝播法に基づく古典回路の1つの構成要素は半加算器である．これは**図-3(a)**の古典回路HAであり，たとえば入力を(x_1, y_1)とすると，出力は(s_1, c_2)となる．もう1つの構成要素は全加算器である．これは**図-3(b)**の古典回路FAであり，たとえば入力を(c_j, x_j, y_j)とすると，出力は(s_j, c_{j+1})となる．そこで**図-3(c)**の古典回路を考える．この回路では初めに(x_1, y_1)がHAの入力となり，その出力は上で述べたように(s_1, c_2)となる．したがって次のFAの入力は(c_2, x_2, y_2)となるため，その出力は(s_2, c_3)となる．これはまさに桁上げ伝播法であり，図-3(c)は桁上げ伝播法に基づく3ビットの加算演算を実行する古典回路である．FAをさらに繋ぎ合わせれば，nビットの加算演算を実行する古典回路となり，その計算コスト，すなわち基本ゲートの個数はnの定数倍程度となることが確認できる．

桁上げ伝播法に基づく量子加算回路

1996年，Vedralらは桁上げ伝播法に基づく量子加算回路を提案した[2]．この回路において，上で述べた全加算器に対応するのが**図-4(a)**の量子回路Cである．Cの入力を$|c_j\rangle\,|x_j\rangle\,|y_j\rangle\,|0\rangle$とすると出力が$|c_j\rangle\,|x_j\rangle\,|x_j \oplus y_j\rangle\,|c_{j+1}\rangle$となり，Cがおよそ全加算

器に対応していることが確認できる．$x_j \oplus y_j$ の代わりに s_j を出力とし，全加算器と正確に対応させることも容易であるが，それをしていないのは回路の後半部分（**図 -4 (c)** の中の C の繰り返しの後の CNOT ゲート以降）を簡単にするためであろう．$|0\rangle$ に初期化された量子ビットを用意し，また，c_j と x_j も出力としているのは，量子演算の可逆性を保証するためである．

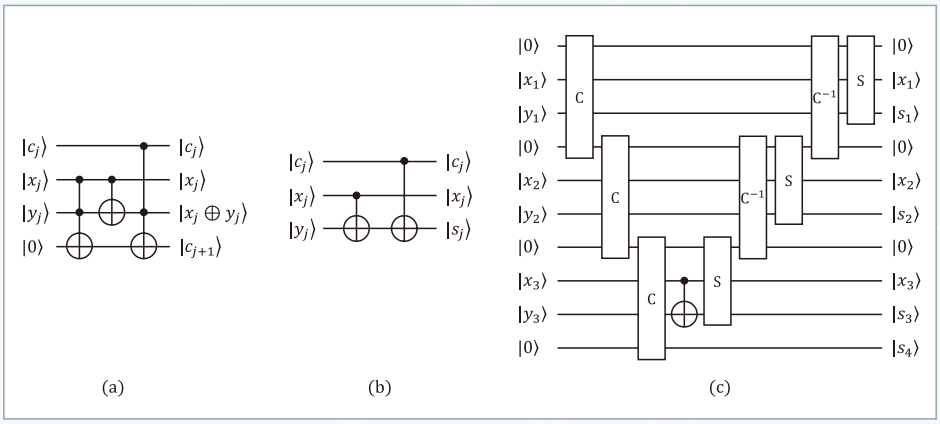

図 -4 (a) c_{j+1} を計算する量子回路 C．2 つの黒丸と⊕を結んだゲートは図 -2 の回路であり，2 つの黒丸は QAND の 2 つの制御量子ビットに対応する．(b) s_j を計算する量子回路 S．(c) 桁上げ伝播法に基づく量子加算回路[2]．C^{-1} は C により定義される演算の逆演算を実行する回路である

Vedral らは，半加算器に対応する回路は構成せず，C で代用している．すなわち，C の入力を $|0\rangle|x_1\rangle|y_1\rangle|0\rangle$ とすると，出力は $|0\rangle|x_1\rangle|s_1\rangle|c_2\rangle$ となることから，C を半加算器として使うのである．**図 -4(b)** の量子回路 S を使い，3 ビットの加算演算を実行する Vedral らの回路は図 -4(c) のように構成される．この回路の中の C の繰り返しが図 -3(c) の回路の中の半加算器と全加算器の繰り返しに対応している．

上で触れたように，C をわずかに変更するだけですべての s_j を出力することは容易であり，加算の結果さえ出力できれば良いという場合には，図 -4(c) の回路の後半部分は必要ない．後半部分で行っているのは，s_j を出力しつつ，入力時に $|0\rangle$ に初期化されていた量子ビット（$x+y$ の最上位ビットを出力する量子ビット以外）を $|0\rangle$ に戻す操作である．このような量子ビットを以下では補助量子ビットと呼ぶ．補助量子ビットを $|0\rangle$ に戻すことにより，これらの再利用が可能となり，より複雑な算術演算回路を構成する際に有用となる．図 -4(c) の回路を拡張して n ビットの加算演算を実行する量子回路を構成すると，計算コストは n の定数倍程度となり，上で述べた古典加算回路と同程度となる．一方，古典加算回路とは異なり，可逆性を保証するための補助量子ビットが n 個必要となるが，桁上げ伝播法に基づく限り Vedral らの回路は自然な構成であろう．

重ね合わせ状態を利用した量子加算回路

Vedral らの回路は，確かに量子回路ではあるものの，実際には重ね合わせ状態を必要とせず，$|000\rangle$ のような古典的な状態を古典的な状態に変化させる量子演算だけを使うものである．すなわち，この回路は本質的に古典回路である．これとは対照的に，Draper は 2000 年，重ね合わせ状態を積極的に使う加算回路を提案した[3]．この回路の重要な構成要素は量子フーリエ変換と呼ばれる量子演算である．たとえば 3 ビットの加算演算を実行する場合，4 量子ビット上の量子フーリエ変換を実行する量子回路が必要となるが，これは **図 -5(a)** の回路 F である．入力を $|y_1\rangle|y_2\rangle|y_3\rangle|0\rangle$ とすると，出力は次の状態 $|\Phi\rangle$ となる：

$$\frac{1}{\sqrt{2^4}}\sum_{j=0}^{2^4-1} e^{2\pi i j y/2^4}\,|j\rangle.$$

ただし，y は $y_3\,y_2\,y_1$（y_1 が最下位桁）によって表現される自然数であり，$|j\rangle$ は j の 4 ビット表現とする．量子フーリエ変換は情報の表現方法を変換する演算であり，通常のビット表現から，重ね合わせられた状態 $|j\rangle$ とその前の係数（位相と呼ばれる）を利用した表現への変換と理解できる．

図 -5(b) は 3 ビットの加算演算を実行する Draper の量子回路である．初めに y を表現する $|y_1\rangle|y_2\rangle|y_3\rangle|0\rangle$（$y_1$ が最下位桁）が F の入力となり，出力は $|\Phi\rangle$ となる．この出力は，$|x_3\rangle$ によって制御された R_2 ゲートの入力となり，その出力が $|x_2\rangle$ によって制御された R_3 ゲートの入力となる．このような処

理が繰り返され，$|x_1\rangle$ によって制御された R_1 ゲートまでの処理により，x が位相に加算される．すなわち，$|\Phi\rangle$ の中の各 $|j\rangle$ の前の係数 $e^{2\pi ijy/2^4}$ が $e^{2\pi ij(x+y)/2^4}$ となる．したがって，$\mathbf{F}^{-1}$ の直前までのゲートにより，次の状態が得られる：

$$\frac{1}{\sqrt{2^4}}\sum_{j=0}^{2^4-1} e^{2\pi ij(x+y)/2^4}\left|j\right\rangle.$$

$\mathbf{F}^{-1}$ は 4 量子ビット上の量子フーリエ変換の逆演算を実行する量子回路である．この $\mathbf{F}^{-1}$ により情報の表現方法を元に戻すと，位相にある $x+y$ の情報が 4 ビット表現 $|s_1\rangle|s_2\rangle|s_3\rangle|s_4\rangle$（$s_1$ が最下位桁）として得られる．このような考え方を基に，n ビットの加算演算を実行する量子回路を構成することができ，その計算コストは n^2 の定数倍程度となる．この回路は補助量子ビットを必要としない．

Vedral らの回路のように，古典回路をある意味自然に模倣することで量子回路を構成すれば，補助量子ビットが必要となる．一方 Draper の回路は，計算コストは大きいものの，量子回路特有のゲートで実行される量子フーリエ変換を使うことで，重ね合わせ状態を積極的に使う真に量子的な回路となっている．このことが，補助量子ビットを必要としないという一見不自然な，しかし理論的にも実用的にも興味深い回路の実現につながっていると考えられる．

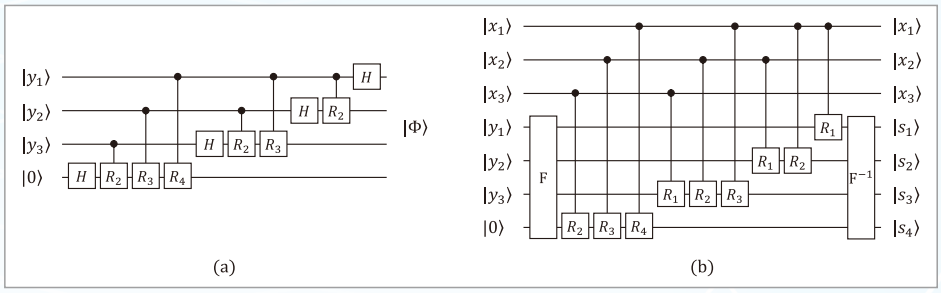

図 -5　(a) 4 量子ビット上の量子フーリエ変換を実行する量子回路 F．黒丸と R_k を結んだゲートは，黒丸が乗っている量子ビットの状態によって制御された R_k ゲートであり，定数個の基本ゲートに分解できる．(b) 量子フーリエ変換に基づく量子加算回路 [3]

関連する話題

Vedral らの回路と Draper の回路の「良いとこどり」はできるであろうか．すなわち，計算コストは n の定数倍程度で，補助量子ビットを使わず量子加算回路を構成できるであろうか．2005 年，筆者らは桁上げ伝播法を使い，これが可能であることを証明した [4]．この回路は真に量子的ではないが，古典回路の自然な模倣でもない．

桁上げ先見法も有名な加算アルゴリズムであり，これに基づく量子回路も構成されている [5]．しかし，この回路も本質的に古典回路であり，筆者の知る限り，真に量子的な加算回路は現時点で Draper の回路のみである．加算回路の構成は単純な問題ではあるが，我々の予想もつかない新しい量子加算回路が発見できれば，計算の側面からの量子性の理解に貢献する興味深いものとなるであろう．

参考文献

1) Nielsen, M. A. and Chuang, I. L. 共著，木村達也 訳：量子コンピュータと量子通信 II ―量子コンピュータとアルゴリズム―，オーム社 (2005).
2) Vedral, V., Barenco, A. and Ekert, A. : Quantum Networks for Elementary Arithmetic Operations, Phys. Rev. A, Vol.54, No.1, pp.147–153 (1996).
3) Draper, T. G. : Addition on a Quantum Computer, arXiv:quant-ph/0008033 (2000).
4) Takahashi, Y. and Kunihiro, N. : A Linear-size Quantum Circuit for Addition with no Ancillary Qubits, Quant. Inf. Comp., Vol.5, No.6, pp.440–448 (2005).
5) Draper, T. G., Kutin, S. A., Rains, E. M. and Svore, K. M. : A Logarithmic-depth Quantum Carry-lookahead Adder, Quant. Inf. Comp., Vol.6, No.4&5, pp.351–369(2006).

(2014 年 3 月 19 日受付)

高橋康博（正会員） takahashi.yasuhiro@lab.ntt.co.jp

2000 年東北大学大学院理学研究科数学専攻博士前期課程修了．同年日本電信電話（株）入社．2008 年電気通信大学大学院電気通信学研究科情報通信工学専攻博士後期課程修了．博士（工学）．量子計算理論の研究に従事．