量子回路と古典回路の相違

高橋　康博

量子回路は，量子コンピューター上のアルゴリズムを表現する代表的な方法の一つである．これは，現在のコンピューター上のアルゴリズムを表現する古典回路の対応物であるが，古典回路における演算を量子回路でそのまま実行できるとは限らず，単純なアルゴリズムを表現した場合にもこれらの回路には相違が生じる．本稿では，このような相違について加算回路を題材として述べるとともに，量子回路研究の最新の話題を紹介する．

キーワード：量子回路，古典回路，可逆性，加算回路

1. はじめに

最近メディアを賑わすことの多い量子コンピューターであるが，その内容を見ると，「ゲート型」または「汎用」と呼ばれる量子コンピューターが頻繁に扱われている．「ゲート型」が意味するのは，基本的な演算を実行するゲートと呼ばれる部品の組み合わせによって処理が表現されることであり，「汎用」が意味するのは，特定の問題を解くことに特化していないことである．本稿で扱う量子コンピューターはこの種類のものであり，量子回路と呼ばれる計算モデルが基礎となっている．

一般的に，われわれの身の回りにあるコンピューターも「ゲート型」で「汎用」とみなすことができ，古典回路と呼ばれる計算モデルが基礎となる．「古典」と呼ぶのは，現在のコンピューターが（量子力学ではなく）古典力学に基づいていることに由来する．本稿では，量子回路と古典回路を比較し，その相違を通して量子コンピューターの振る舞いについて紹介する．相違といってもさまざまな側面が考えられるが，回路ならではの視覚的な表現における相違を中心に議論を進める．

2. 古典回路

現在のコンピューターの基礎となる古典回路の構成要素は，ビット列にビット列を対応させる演算（古典演算）を実行する「古典ゲート」であり，ここでは，以下で述べる NOT ゲートと AND ゲートを用意する．そして，これらを組み合わせることで，入力ビット列を処理して何らかのビット列を出力する仕組みが構成できれば，それが古典回路である．もちろん古典回路を厳密に定義して形式的に議論を進めることも可能ではあるが，本稿では直観的な議論を優先したい．

NOT ゲートが実行するのは否定演算であり，入力 $x \in \{0,1\}$ に対し，その否定 $x \oplus 1 \in \{0,1\}$ を出力する．ここで $\oplus$ は排他的論理和（2 を法とする加算）を表す．また，AND ゲートが実行するのは 2 ビットの論理積演算であり，入力 $x, y \in \{0,1\}$ に対し，その論理積 $x \wedge y \in \{0,1\}$ を出力する．古典回路の例を図 1(a) に示す．$x, y \in \{0,1\}$ は回路の入力であり，それぞれ NOT ゲートの入力となる．そして，NOT ゲートの出力は AND ゲートの入力となり，AND ゲートの出力は NOT ゲートの入力となる．この NOT ゲートの出力が回路の出力であり，これが回路の入力 x, y の論理和 $x \vee y$ と一致することは簡単に確認できる．すなわち，この古典回路は 2 ビットの論理和演算を実行する．

一般に，n ビット列に m ビット列を対応させる任意の古典演算を実行するためには，NOT ゲートと AND ゲートを用意すれば十分であることが知られている．ここで n, m は任意の自然数である．この意味で，これらの古典ゲートの集合は万能であると言われる．

3. 量子回路

古典回路に関する上の議論に対応する形で量子回路の議論を進めよう．量子回路の構成要素は量子状態に量子状態を対応させる演算（量子演算）を実行する「量子ゲート」であり，これを組み合わせ，入力量子状態を遷移させて何らかの量子状態を出力する仕組みが構成できれば，それが量子回路である．まずは NOT ゲートと AND ゲートの量子版を用意する．

3.1 量子 NOT ゲートと量子 AND ゲート

NOT ゲートと AND ゲートの入出力関係をもとに，対応する量子ゲートの入出力関係を定めたい．そもそ

たかはし　やすひろ
NTT コミュニケーション科学基礎研究所
メディア情報研究部
〒 243–0198 神奈川県厚木市森の里若宮 3–1
takahashi.yasuhiro@lab.ntt.co.jp

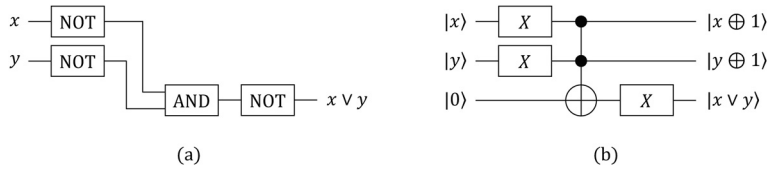

図 1　(a)：2 ビットの論理和演算を実行する古典回路．(b)：(a) の古典回路における処理を模倣する量子回路．二つの黒丸と $\oplus$ を繋いだ量子ゲートはトフォリゲートであり，黒丸に対応するのは制御ビットである．

も NOT ゲートと AND ゲートの入出力関係は，そのまま量子演算の入出力関係と考えてよいのであろうか．量子力学の規約により，n 量子ビット上の量子演算は，複素数を成分とする 2^n 行 2^n 列のユニタリ行列で表現される．たとえば，次の行列

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

は 1 量子ビット上の量子演算を表現している．任意の $x \in \{0,1\}$ に対し，$X|x\rangle = |x \oplus 1\rangle$ となることが簡単に確認でき，NOT ゲートの入出力関係はそのまま量子演算の入出力関係となることがわかる．ここで，

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

である．量子 NOT ゲートは，この量子演算を実行する量子ゲートとする．

量子 AND ゲートを考えよう．古典回路における AND ゲートは，2 ビット入力 1 ビット出力であったが，上で述べた量子力学の規約により，2 量子ビット入力の量子演算は 2 量子ビットの状態を出力する．それでは，任意の $x, y \in \{0,1\}$ に対し，$|x\rangle|y\rangle \mapsto |x\rangle|x \wedge y\rangle$ という入出力関係を考えてはどうであろうか．残念ながら，これは量子演算の一つの性質である可逆性，すなわち，出力から入力が一意に定まるという性質を満たさない．実際，出力 $|0\rangle|0\rangle$ に対して，二つの入力 $|0\rangle|0\rangle$ と $|0\rangle|1\rangle$ が存在する．一般に，2 量子ビット上の量子演算では，$|x\rangle|y\rangle$ を入力として $|x \wedge y\rangle$ を出力の一部とすることはできない．そこで，量子 AND ゲートは，次の 3 量子ビット上の量子演算を実行する量子ゲートとする：任意の $x, y, z \in \{0,1\}$ に対し，

$$|x\rangle|y\rangle|z\rangle \mapsto |x\rangle|y\rangle|z \oplus (x \wedge y)\rangle.$$

たとえば $|x \wedge y\rangle$ を出力する場合には，$z = 0$ とする．すなわち，入力として $|0\rangle$ に初期化された量子ビットを用意し，そこに $x \wedge y$ を保存するのである．上の 3 量子ビット上の量子演算はトフォリ演算と呼ばれ，これを実行する量子ゲートはトフォリゲートと呼ばれる．そこで本稿でも「量子 AND」の代わりに「トフォリ」という呼び名を使う．この演算において $x = y = 1$ の場合は $|z\rangle$ が $|z \oplus 1\rangle$ に遷移し，それ以外の場合は入力がそのまま出力となる．したがって，トフォリ演算は x, y によって制御された否定演算とみなすことができ，x, y を表現する量子ビットは制御ビットと呼ばれる．

3.2　古典回路の量子版

量子 NOT ゲートとトフォリゲートを構成要素とする量子回路の例を図 1(b) に示す．これは，量子ゲートを使って図 1(a) の古典回路における処理を模倣したものであり，いわば図 1(a) の古典回路の量子版である．図 1(b) は，入力量子状態 $|x\rangle|y\rangle|0\rangle$ に対し，量子ゲートが左のものから順に適用され，最終的に $|x \oplus 1\rangle|y \oplus 1\rangle|x \vee y\rangle$ という量子状態が出力されることを表している．可逆性を保証するために $|0\rangle$ に初期化された量子ビットが使われており，古典回路における 2 ビット入力の計算が 3 量子ビットの状態の遷移として実現されている．

図 1(a) の古典回路と図 1(b) の量子回路の対応関係を一般化すれば，現在のコンピューターで可能な計算は量子コンピューターでも可能であることがわかる．すなわち，現在のコンピューターにおける処理は NOT ゲートと AND ゲートからなる古典回路で表現することができ，$|0\rangle$ に初期化された量子ビットを十分用意すれば，量子 NOT ゲートとトフォリゲートを使い，この古典回路と同じ出力を行う量子回路が構成できる．現在のコンピューターにおける計算は，量子コンピューターにおける計算とみなすことができるのである．

3.3　量子ゲートの万能集合

量子 NOT ゲートとトフォリゲートを用意するだけでは，$|0\rangle$ や $|1\rangle$ に重ね合わせ状態を対応させるような量子演算は実行できない．任意の量子演算を実行するためには，どのような量子ゲートを用意すればよいのであろうか．はじめに，1 量子ビット上の任意の量子演算の実行に焦点を当てる．次の行列を考えよう：

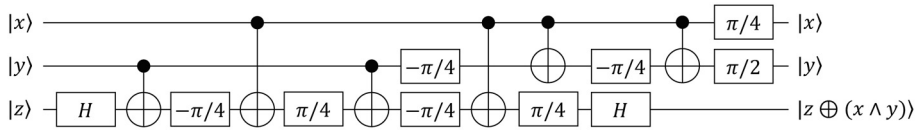

図 2　トフォリ演算を実行する量子回路．H はアダマールゲートであり，実数 $\pi/4, -\pi/4, \pi/2$ で表されている量子ゲートは，それぞれ角度 $\pi/4, -\pi/4, \pi/2$ の位相シフトゲートである．また，黒丸と $\oplus$ を繋いだ量子ゲートは CNOT ゲートであり，黒丸に対応するのは制御ビットである．

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},\ Z(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}.$$

ここで θ は任意の実数である．H と $Z(\theta)$ は 1 量子ビット上の量子演算を表現しており，この量子演算は，それぞれアダマール演算，角度 θ の位相シフト演算と呼ばれる．任意の $x \in \{0,1\}$ に対し，

$$H|x\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{(-1)^x}{\sqrt{2}}|1\rangle,\ Z(\theta)|x\rangle = e^{i\theta x}|x\rangle$$

という関係が成り立つことは簡単に確認できる．アダマール演算は，入力が $|0\rangle$ の場合も $|1\rangle$ の場合も，重ね合わせ状態を出力する．また，角度 θ の位相シフト演算は，入力が $|0\rangle$ の場合はそのまま出力するが，$|1\rangle$ の場合は係数（位相と呼ばれる）$e^{i\theta}$ を付加する．

このような H と $Z(\theta)$ は 1 量子ビット上の量子演算の基礎となる．より正確には，$J(\theta) = HZ(\theta)$ とするとき，任意の 2 行 2 列のユニタリ行列 U に対し，ある実数 $\alpha, \beta, \gamma, \delta$ が存在して，$U = e^{i\alpha}J(0)J(\beta)J(\gamma)J(\delta)$ となる [1]．係数 $e^{i\alpha}$ による量子演算の差異は，ここでの文脈では無視できるため，1 量子ビット上の量子演算は H と $Z(\theta)$ で表現できる．アダマール演算と（任意の実数 θ に対する）角度 θ の位相シフト演算を実行する量子ゲートをそれぞれアダマールゲート，角度 θ の位相シフトゲートと呼ぶことにする．1 量子ビット上の任意の量子演算を実行するためには，これらの量子ゲートを用意すれば十分であることになる．

一般に，n 量子ビット上の量子演算は，1 量子ビット上の量子演算とともに，2 量子ビット上の量子演算である CNOT 演算を使うことで表現できる [2]．ここで CNOT は制御否定 (Controlled NOT) を意味し，次の入出力関係をもつ：任意の $x, y \in \{0,1\}$ に対し，$|x\rangle|y\rangle \mapsto |x\rangle|x \oplus y\rangle$．トフォリゲートと同様に，$x$ によって制御された否定演算とみなすことができ，x を表現する量子ビットは制御ビットと呼ばれる．CNOT 演算を実行する量子ゲートを CNOT ゲートと呼ぶことにしよう．上で述べたように，1 量子ビット上の任意の量子演算の実行には，アダマールゲートと（任意の実数 θ に対する）角度 θ の位相シフトゲートを用意すれば十分であった．したがって，n 量子ビット上の任意の量子演算を実行するためには，これらに加えて CNOT ゲートを用意すればよい．すなわち，これらの量子ゲートの集合は（量子コンピューターにおける計算に対して）万能となる．この万能集合は，無限種類の角度の位相シフトゲートを含むため，2 節で述べた古典ゲートの万能集合とは異なり無限集合である．ただし，任意の量子演算を近似的に実行する場合は，位相シフトゲートを角度 $\pi/4$ のものだけに制限できる [2]．

図 1(b) の量子回路は量子 NOT ゲートとトフォリゲートで構成されている．上の万能集合に属する量子ゲートだけを使い，同じ量子演算を実行する量子回路を構成しよう．簡単に確認できるように，$X = HZ(\pi)H$ が成り立つため，量子 NOT ゲートは二つのアダマールゲートと角度 π の位相シフトゲートで置き換えられる．また，トフォリ演算は図 2 の量子回路で実行できるため，トフォリゲートはこの量子回路で置き換えられる．これらの置換により求める量子回路が得られる．以下で扱う量子回路は上で述べた万能集合に属する量子ゲートから構成されるとしよう．便宜的にトフォリゲートを使う場合は，図 2 の量子回路の略記と考える．

4. 加算回路

二つの自然数の加算を行う量子回路と古典回路の相違をみよう．入力は二つの n ビット列 $a_n \cdots a_1$, $b_n \cdots b_1 \in \{0,1\}^n$ であり，それぞれ自然数 a, b の 2 進数表現とする．ただし，a_1, b_1 が最下位ビットである．以下では，われわれになじみ深い単純な加算アルゴリズムである桁上げ伝搬法を扱う．

4.1 桁上げ伝搬法に基づく古典回路

桁上げ伝搬法における最初の処理では，入力ビット列における最下位ビット a_1, b_1 をもとに，最下位桁からの桁上げの有無を判定する．次に，この桁上げの情報と入力ビット列における一つ上位のビット a_2, b_2 をもとに，この桁からの桁上げの有無を判定する．同様の操作を最上位桁まで繰り返せば，各桁からの桁上げの有無を判定でき，$a + b$ の 2 進数表現が得られる．

この処理を正確に述べるため，各桁からの桁上げの

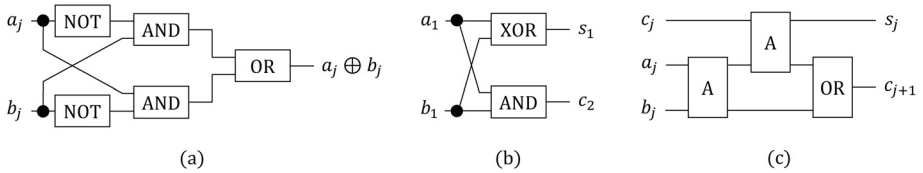

図 3　(a)：排他的論理和を計算する古典回路．OR は図 1(a) の古典回路である．(b)：半加算器と呼ばれる古典回路．XOR は (a) の古典回路である．(c)：全加算器と呼ばれる古典回路．A は半加算器である．

有無を表現するビット 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，$c_{j+1} = (a_j \wedge b_j) \oplus (b_j \wedge c_j) \oplus (c_j \wedge a_j)$．また，入力の各桁と各 c_j から計算されるビット s_j $(1 \le j \le n+1)$ を次のように定義する：$s_{n+1} = c_{n+1}$ とし，任意の $1 \le j \le n$ に対し，$s_j = a_j \oplus b_j \oplus c_j$．このとき，$n+1$ ビット列 $s_{n+1}s_n \cdots s_1$（s_1 が最下位ビット）は $a+b$ の 2 進数表現となることがわかる．桁上げ伝搬法は，c_2 から順に c_{n+1} まで計算を行い，これをもとに各 s_j を計算するアルゴリズムである．

桁上げ伝搬法に基づく古典回路の構成には排他的論理和を計算する古典回路が必要となるが，これは図 3 (a) のように構成できる．図中の黒丸は，ビットがコピーされる点を表しており，たとえば入力 a_j はコピーされて，NOT ゲートと AND ゲートの入力となる．量子回路における黒丸とは用法が異なることに注意する．桁上げ伝搬法に基づく古典回路の構成要素の一つは図 3(b) のように構成される半加算器と呼ばれる古典回路である．入力が a_1, b_1 の場合，これは XOR と表された図 3 (a) の古典回路の入力となり，$a_1 \oplus b_1 = s_1$ が出力される．入力 a_1, b_1 は AND ゲートの入力ともなり，$a_1 \wedge b_1 = c_2$ が出力される．

もう一つの構成要素は，図 3(c) のように構成される全加算器と呼ばれる古典回路である．任意の $2 \le j \le n$ に対し，入力が c_j, a_j, b_j の場合，a_j, b_j は半加算器の入力となり，$a_j \oplus b_j$ と $a_j \wedge b_j$ が出力される．次に，c_j と $a_j \oplus b_j$ が半加算器の入力となり，$a_j \oplus b_j \oplus c_j = s_j$ と $c_j \wedge (a_j \oplus b_j)$ が出力される．最後に，$c_j \wedge (a_j \oplus b_j)$ と $a_j \wedge b_j$ が OR と表された図 1(a) の古典回路の入力となるが，この出力 $(a_j \wedge b_j) \vee (c_j \wedge (a_j \oplus b_j))$ が $(a_j \wedge b_j) \oplus (b_j \wedge c_j) \oplus (c_j \wedge a_j) = c_{j+1}$ と一致することは，上の式において $\vee$ を $\oplus$ と置き換えられることを使って確認できる．これらの入出力関係により，桁上げ伝搬法に基づく古典回路は，最初に半加算器を適用し，その後は全加算器を繰り返し適用することで構成できる．$n = 3$ の場合は図 4(a) のようになる．

4.2　桁上げ伝搬法に基づく量子回路

桁上げ伝搬法に基づいて，さまざまな量子回路が構成されているが，ここでは最も基本的な構成だと思われる Vedral et al. の量子回路を取り上げる [3]．Vedral et al. は全加算器に対応する量子回路を図 4(b) のように構成した．可逆性を保証するために，$|0\rangle$ に初期化された量子ビットを用意している．任意の $1 \le j \le n$ に対し，入力量子状態を $|c_j\rangle|a_j\rangle|b_j\rangle|0\rangle$ とするとき，この状態は各ゲートにより次のように遷移する：

$$\begin{aligned}|c_j\rangle|a_j\rangle|b_j\rangle|0\rangle &\mapsto |c_j\rangle|a_j\rangle|b_j\rangle|a_j \wedge b_j\rangle \\ &\mapsto |c_j\rangle|a_j\rangle|a_j \oplus b_j\rangle|a_j \wedge b_j\rangle \\ &\mapsto |c_j\rangle|a_j\rangle|a_j \oplus b_j\rangle|c_{j+1}\rangle.\end{aligned}$$

ここで $(a_j \wedge b_j) \oplus (c_j \wedge (a_j \oplus b_j)) = c_{j+1}$ が成り立つことに注意する．全加算器に対応するこの量子回路を C と表そう．C の入出力関係により，これを繰り返し適用すれば，すべての c_j が計算できる．$n = 3$ の場合の Vedral et al. の量子回路は図 4(c) のようになる．

図 4(c) における状態の遷移をみる．入力量子状態は

$$|0\rangle|a_1\rangle|b_1\rangle|0\rangle|a_2\rangle|b_2\rangle|0\rangle|a_3\rangle|b_3\rangle|0\rangle$$

である．入力の時点で $|0\rangle$ に初期化されている量子ビットを（s_4 を保存するためのものを除いて）補助量子ビットと呼ぶことにする．上で述べた C による状態遷移から，すべての C が適用された後の状態は

$$|0\rangle|a_1\rangle|s_1\rangle|c_2\rangle|a_2\rangle|a_2 \oplus b_2\rangle|c_3\rangle|a_3\rangle|a_3 \oplus b_3\rangle|s_4\rangle$$

となる．この時点で $c_2, c_3, c_4 = s_4$ が計算されている．次に，CNOT ゲートと $C^\dagger$ により，状態は

$$|0\rangle|a_1\rangle|s_1\rangle|c_2\rangle|a_2\rangle|b_2\rangle|0\rangle|a_3\rangle|s_3\rangle|s_4\rangle$$

となり，一つの補助量子ビットの状態が $|0\rangle$ に戻される．その後の CNOT ゲートと $C^\dagger$ により，同様に一つの補助量子ビットの状態が $|0\rangle$ に戻され，状態は

$$|0\rangle|a_1\rangle|b_1\rangle|0\rangle|a_2\rangle|b_2 \oplus c_2\rangle|0\rangle|a_3\rangle|s_3\rangle|s_4\rangle$$

となる．最後の二つの CNOT ゲートにより，状態は

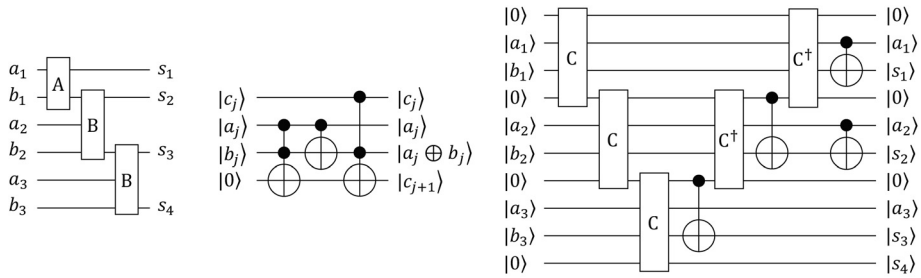

図 4 (a)：桁上げ伝搬法に基づく古典回路．B は全加算器である．(b)：全加算器に対応する量子回路．(c)：桁上げ伝搬法に基づく量子回路 [3]．C は (b) の量子回路である．また，$C^\dagger$ は C が実行する量子演算の逆演算を実行する量子回路であり，この場合，C に含まれる量子ゲートを逆の順に適用する量子回路とすればよい．

$$|0\rangle|a_1\rangle|s_1\rangle|0\rangle|a_2\rangle|s_2\rangle|0\rangle|a_3\rangle|s_3\rangle|s_4\rangle$$

となるが，ここでは s_1, s_2 が計算され，所望のビット列 $s_4s_3s_2s_1$ が得られたことになる．

図 4(a) の古典回路と図 4(c) の量子回路を比較しよう．上で述べた状態遷移からわかるように，古典回路における半加算器とその後の全加算器の繰り返しに対応するのが，量子回路の中で C を繰り返している部分である．したがって，これらの回路の大きな相違として，量子回路における C の繰り返しの後の処理と補助量子ビットの存在が挙げられる．補助量子ビットは可逆性を保証するために使われているが，$C^\dagger$ などを使った後半の処理は何を目的としているのであろうか．

上で述べた状態遷移から，C の繰り返しの直後の状態において s_1 と s_4 が得られており，残りの s_2 と s_3 はその状態に二つの CNOT ゲートを適用するだけで計算できることがわかる．それをせずに，$C^\dagger$ などを使った処理を行っているのは，所望のビット列 $s_4s_3s_2s_1$ を得るだけでなく，補助量子ビットの状態を $|0\rangle$ に戻すためである．これは量子ビット数の節約に貢献する．たとえば，加算回路を直列的に何度も適用する状況を考えよう．加算回路が補助量子ビットの状態を元に戻すのであれば，加算回路の適用回数に依存して，多くの補助量子ビットを用意する必要はなく，一度の加算回路の適用に必要な補助量子ビットだけを用意しておき，それを再利用できる．このような処理が行われる背景には，多くの（特に $|0\rangle$ に初期化された）量子ビットを現実的に用意するのは困難であるという現状がある．古典回路では考慮する必要のないこのような補助量子ビットについては，その個数や最終状態も含めて検討が必要となるのである．

4.3 重ね合わせ状態を利用した加算回路

量子回路特有の状態遷移に基づく加算回路は，量子回路と古典回路の相違を考えるうえで有用であろう．ここでは，このような Draper の量子回路について紹介する [4]．量子回路の詳細は煩雑になるため省略し，大雑把な回路構成と状態遷移を述べる．簡単のため $n = 3$ の場合を扱い，入力量子状態は $|a_1\rangle|a_2\rangle|a_3\rangle|b_1\rangle|b_2\rangle|b_3\rangle|0\rangle$ とする．Vedral et al. の量子回路の場合とは入力ビット列の並べ方が異なっているが，便宜的なものである．Draper の量子回路では，初期状態が $|a_1\rangle|a_2\rangle|a_3\rangle$ である量子ビットの状態は一切変化しない．したがって，初期状態が $|b_1\rangle|b_2\rangle|b_3\rangle|0\rangle$ である量子ビットの状態の遷移のみを考える．Draper の量子回路による状態遷移は次のような三段階で捉えられる：

$$\begin{aligned}|b_1\rangle|b_2\rangle|b_3\rangle|0\rangle &\mapsto \frac{1}{\sqrt{2^4}}\sum_{j=0}^{2^4-1} e^{2\pi ijb/2^4}|j\rangle \\ &\mapsto \frac{1}{\sqrt{2^4}}\sum_{j=0}^{2^4-1} e^{2\pi ij(a+b)/2^4}|j\rangle \\ &\mapsto |s_1\rangle|s_2\rangle|s_3\rangle|s_4\rangle.\end{aligned}$$

ここで j は 0 から 15 までの自然数を表しており，$|j\rangle$ はその（4 量子ビットによる）2 進数表現とする．この遷移を大雑把に述べれば，はじめに，通常のビット列として表現された b を，重ね合わせ状態の位相を利用した表現に変換する．この変換は量子フーリエ変換と呼ばれ，それを実行する量子回路が使われる．次に，位相の上で b に a を加算する．ここでの加算は，ビット列の表現における加算ではなく，位相を利用した表現における加算であり，さまざまな角度の位相シフトゲートで実行される．そして，量子フーリエ変換の逆演算を実行する量子回路により，位相を利用した $a + b$ の表現を通常のビット列の表現に戻す．重ね合わせ状態を利用した情報の表現と位相の上での操作は，もちろん古典回路にはありえない量子回路特有のものである．

4.4 量子回路と古典回路の評価尺度

上で述べた加算回路を利用して，量子回路と古典回路の評価尺度を紹介する．量子回路と古典回路で共通に使われるのは，回路に含まれるゲートの個数である．たとえば，入力が二つの n ビット列の場合，Vedral et al. の量子回路に含まれる量子ゲートの個数は n に比例する程度であり，Draper の量子回路では n^2 に比例する程度である．また，桁上げ伝搬法に基づく古典回路では n に比例する程度である．詳細は省略するが，ゲートの並列な実行可能性を考慮する「深さ」という評価尺度も量子回路と古典回路において広く使われる [5, 6].

量子回路特有の評価尺度としては，補助量子ビットの個数がある．たとえば，Vedral et al. の量子回路において，補助量子ビット数は n である．出力量子状態において s_{n+1} を保存するための量子ビットは補助量子ビットとは数えないのが一般的である．そして，Draper の量子回路は補助量子ビットを全く使わない [4]．たとえば，Vedral et al. の量子回路と Draper の量子回路の比較から，補助量子ビットを全く使わず，かつ，量子ゲートの個数を n に比例する程度にできるかという問題が考えられるが，筆者らは，桁上げ伝搬法に基づいて，この条件を満たす量子回路を構成した [7].

5. 最新の話題

量子回路と古典回路の相違に関わる二つの最新の研究の方向性について触れる．一つは，量子回路において頻繁に利用される $|0\rangle$ に初期化された量子ビットに関するものであり，もう一つは，量子回路と古典回路の計算能力の相違に関するものである．

5.1 初期化量子ビットに代わる計算資源

量子回路において，$|0\rangle$ に初期化された量子ビットがある意味自然に必要となる一方で，このような量子ビットを現実的に多数用意することが困難であることはすでに述べた．幸い加算回路の場合は，量子ゲートの個数を大きく増加させることなく，このような量子ビットを使わない量子回路を構成できる．しかし，初期化量子ビットを使い，より複雑な計算を行っている場合に，量子ゲートの個数に影響を与えず，初期化量子ビット数を大きく削減できるかどうかは不明であり，一般には期待できない．したがって，初期化量子ビットに代わる計算資源を検討することは自然であろう．

新たな計算資源として筆者らの最新の研究において扱っているのは，未初期化量子ビットである．これは，初期状態に仮定のない（どのような初期状態でもよい）量子ビットであるが，通常どおり量子ゲートを適用して状態を遷移させることができるものである．初期化の必要がないため，簡単に用意できる一方で，その利用価値はほとんど研究されてこなかった．筆者らの研究では，少ない初期化量子ビットしか使えない状況において，未初期化量子ビットをもつ量子回路の計算能力は未初期化量子ビットをもたないものの計算能力を大きく上回るという証拠を示している [8]．初期化量子ビットの完全な代替物とはならないにしても，未初期化量子ビットの能力の解明と現実的な利用が期待される．

5.2 強い制約をもつ量子コンピューターの分析

十分多くの量子ビット上の任意の量子演算を実行する量子コンピューターの実現には，少々時間がかかるであろう．そこに至る重要な課題は，何らかの制約はもつが，その分実現性の高い量子コンピューターをもとに，現在のコンピューターにおける不可能を可能にすること，すなわち，現在のコンピューターに対する優位性を実証することである．たとえば，自然数の素因数分解を行う高速量子アルゴリズム [9] の実行に特化した量子コンピューターを実現し，十分大きい自然数に適用するという方向性が考えられる．しかし，この量子アルゴリズムは複雑な量子演算を必要とするため，これも近々実現というわけにはいかないようである．

最近，このような方向性をさらに進めた研究，すなわち，現在またはごく近い将来の技術だけで実現される強い制約をもつ量子コンピューターをもとに，現在のコンピューターに対する優位性を実証しようという研究に注目が集まっている [10]．素因数分解とは異なり，優位性の検証は容易ではない可能性はあるが，たとえば，実行時間に強い制約をもつ量子コンピューターにより，現在のコンピューターでは生成が困難な（ビット列上の）確率分布を生成しようという試みがある．このような議論の中心的な役割を果たしているのは量子回路である．実現性の議論とともに，古典回路の計算能力との比較を厳密に議論できる点は極めて都合がよい．実現性の高さと古典回路に対する優位性の両立を目指し，上で述べた評価尺度について強い制約を課すなど，さまざまな量子回路について分析が進められている．

6. おわりに

本稿では，量子回路と古典回路の相違について，加算回路を題材として述べた．また，これに関連して，量子回路に関する最新の研究について触れた．本稿の内

容に興味をもっていただいた方には，次のステップとして，量子回路による表現をもとに高速量子アルゴリズムに触れたり [2]，本特集で取り上げられている量子アニーリングのような他の量子計算モデルと量子回路を比較したりという方向性をお勧めしたい．理論・実験問わず量子コンピューターの研究は着実に進展しており，これとともに，社会からの量子コンピューターへの期待は激しく高まっていると感じている．

参考文献

[1] V. Danos, E. Kashefi and P. Panangaden, "Parsimonious and robust realizations of unitary maps in the one-way model," *Physical Review A*, **72**, article number: 064301, 2005.

[2] M. A. Nielsen and I. L. Chuang（木村達也訳），『量子コンピュータと量子通信 I ―量子力学とコンピュータ科学―』，オーム社，2004.

[3] V. Vedral, A. Barenco and A. Ekert, "Quantum networks for elementary arithmetic operations," *Physical Review A*, **54**, pp. 147–153, 1996.

[4] T. G. Draper, "Addition on a quantum computer," arXiv: quant-ph/0008033, 2000.

[5] T. G. Draper, S. A. Kutin, E. M. Rains and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," *Quantum Information and Computation*, **6**(4), pp. 351–369, 2006.

[6] D. Bera, F. Green and S. Homer, "Small depth quantum circuits," *ACM SIGACT News*, **38**, pp. 35–50, 2007.

[7] Y. Takahashi and N. Kunihiro, "A linear-size quantum circuit for addition with no ancillary qubits," *Quantum Information and Computation*, **5**(6), pp. 440–448, 2005.

[8] Y. Takahashi and S. Tani, "Power of uninitialized qubits in shallow quantum circuits," In *Proceedings of the 35th Annual Symposium on Theoretical Aspects of Computer Science (STACS'18)*, pp. 57:1–57:13, 2018.

[9] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, **26**, pp. 1484–1509, 1997.

[10] A. W. Harrow and A. Montanaro, "Quantum computational supremacy," *Nature*, **549**, pp. 203–209, 2017.