ゴミラインをもつ量子桁上げ伝播加算器回路の深さに関する最適化

M2018SE012 柴田心太郎
指導教員：横山哲郎

1 はじめに

近年，量子計算機の分野の研究は活発であり，その中でも有名な量子アルゴリズムである Shor の素因数分解法や Grover のデータベース探索が知られている．これらの量子アルゴリズムは抽象度が高いため量子回路まで落とし込んで記述する必要がある．したがって，最適な量子回路の実現は重要な課題であり，演算回路の基本となる加算器回路を最適に設計することも重要である．また，量子回路では全域，単射な計算のみを扱う．よって，量子回路は出力から入力が一意に定まる可逆性をもつ．また，非可逆回路を可逆回路にする際，可逆性をもたせるために一時的に情報を保存するラインが必要であることが知られている [1].

現在，量子加算器回路にはゴミラインがなく深さが $O(\log n)$ の桁上げ先見方式 [2] や $O(n)$ の桁上げ伝播方式 [3]，ゴミラインがある深さ $O(n)$ の桁上げ伝播方式のもの [4] が知られている．これらの回路はゲート数や深さなどの指標に関してトレードオフ関係にある．しかし，我々の知る範囲において，ゲート数や深さなどの制約に応じて最適な回路を得る一般的方法，及び量子回路の自動設計は明らかにされていない．

本稿では，入力ビット数 4 のゴミラインがある桁上げ伝播方式の in-place な量子加算器回路の設計をし，既存方式 [2, 3] に対して，入力ビット数が小さいときに量子コスト・ゲート数・深さがより最適なことを示し，さらに一般の場合に量子コスト・ゲート数がより最適であることを示す．本稿では，不要なビットをゴミラインがすべて記憶する埋込み方式を提案する．提案方式は既存方式とトレードオフ関係にある新たな方式であり，本稿のアイデアは他の算術・論理演算の量子回路への応用が期待される．

2 関連研究

量子桁上げ先見加算器では，古典的な場合と同様，各 i に関して a_i と b_i の和 s_i と桁上げ c_i を半加算器で得て，桁上げ情報を二分木状に伝播させる．CN（制御付 NOT）ゲートと CCN（二重制御付 NOT）ゲートのみを用いる Draper 他の方式 [2] では，最下位と最上位以外のラインにおいて，伝播の逆計算で不要な情報を含むラインの値を 0 にする．Fredkin ゲートも用いる Mogensen の方式 [5] では，[2] よりゲート数や量子コストが多いが，ancilla 数が約半分である．

量子桁上げ伝播加算器では，古典的な場合と同様，全桁上げの計算後に最上位から最下位へ順に和を求める．Vedral 他の方式 [3] では桁上げの逆計算を先に行うことで不要な情報を含むラインの値を 0 にする．Takahashi 他 [6] は，量子フーリエ変換を用いることで ancilla ラインを必要としない量子回路を構成した．また，未初期化量子ビットと呼ぶ，初期状態に仮定のない量子ビットだが，量子ゲートを適用して状態を遷移させることができるビットを用いることで，少ない初期化量子ビットしか使えない状況下で，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量子回路の計算能力を大きく上回ることを示した [7].

以上の方式はゴミラインをもたない．一方，ゴミラインを許し CN ゲートと Fredkin ゲートを用いて最適化を行う方式 [4] が知られている．

3 準備

本章では，量子演算，及び本稿で扱う量子ゲートと量子回路の性能を表す指標について説明をする．

3.1 量子演算

量子計算機における演算とは，与えられた量子ビットの状態を，目的とする量子ビットの状態へ遷移させる過程のことであり，量子演算を実行する量子ゲートを組み合わせたものが量子回路である．

量子ビットとは，古典計算機で用いられる古典ビットと同様，量子計算機で用いることができるビットのことであり，量子ビットの量子状態をベクトル，演算を行列で記述できる．1 量子ビットの量子状態は，2 つの基底ベクトル $|0\rangle = \binom{1}{0}$, $|1\rangle = \binom{0}{1}$ と，複素数として α, β を用いて，$\alpha|0\rangle + \beta|1\rangle = \binom{\alpha}{\beta}$ のように 2 次元ベクトルで記述することができる．ここで，量子状態を表すベクトルは単位ベクトルであり，$|\alpha|^2 + |\beta|^2 = 1$ である．n 量子ビットの量子状態は，テンソル積を用いて記述でき，ベクトルの次元は，2^n 次元となる．また，古典ビットの 0 と 1 はそれぞれ量子ビットの $|0\rangle$ と $|1\rangle$ を表している．

以上より量子演算は，与えられた量子状態から，目的とする量子状態への写像を表す行列として記述できる．ここで，入出力のベクトルはユニタリ行列となる．したがって，行列 M の共役転置行列 $M^\dagger$ は，逆行列 M^{-1} と一致する．

このことから，ユニタリ行列は逆演算を持つため量子演算は可逆である．演算についての可逆とは，任意の演算の出力に対してその入力が一意に定まることである．

3.2 量子ゲート

本稿で用いる量子ゲートを図 1(a)–(f) に示す．各ラインの左側の変数を入力として右側の式の値が出力となる．図 1(a)(b)(d) は 2 入出力ゲート，図 1(e)(f) は 3 入出力ゲート

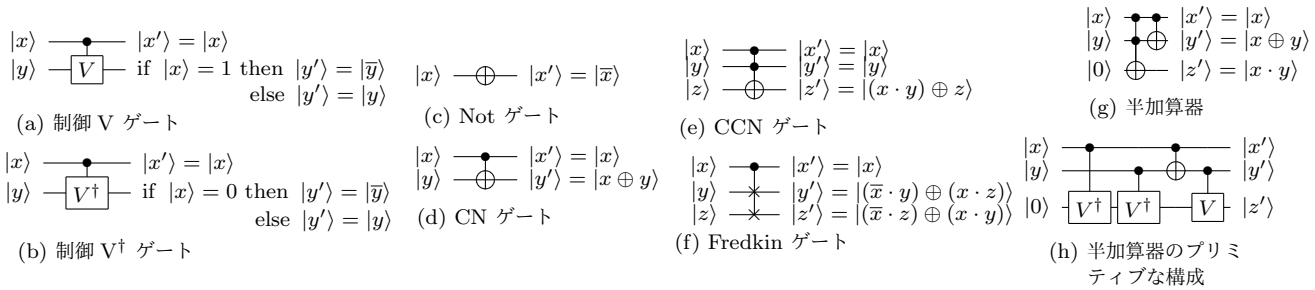


図 1　量子ゲートと半加算器

である．それぞれの縦のラインで論理演算を表す．● で表される制御ビットによって $\oplus$, $\times$, $\boxed{f}$ で表される目標ビットを変化させる．ここで，$\oplus$ は排他的論理和，$\times$ は値の入替え，f は任意の演算を表す．制御 V ゲートは $|x\rangle = 1$ のときのみ $|y'\rangle = |\overline{y}\rangle$ にし，制御 V† ゲートは $|x\rangle = 0$ のときのみ $|y'\rangle = |\overline{y}\rangle$ にする．CN, CCN, Fredkin ゲートは制御ビットがすべて 1 の場合にのみ目標ビットを変化させる．

CN と CCN のゲートを図 1(g) のように並べて半加算器が構成できる．このとき，$|x\rangle$ と $|y\rangle$ は入力ビット，$|0\rangle$ は 0 に初期化された量子ビットである．これらを半加算器に通すことにより $|x'\rangle$, $|y'\rangle$, $|z'\rangle$ が得られる．$|x'\rangle$ は $|x\rangle$ の値をそのまま出力，$|y'\rangle$ は $|x\rangle$ と $|y\rangle$ の入力ビットの和，$|z'\rangle$ は $|x\rangle$ と $|y\rangle$ の桁上げ情報である．

3.3　量子コスト

量子回路の量子コストは，その構成に使われたプリミティブ量子ゲート数である．図 1(a)–(d) はプリミティブ量子ゲートであり量子コストは 1 である．CCN ゲートはプリミティブ量子ゲート 5 つで構成できるので量子コストは 5 である．Fredkin ゲートはプリミティブ量子ゲート 7 つで構成でき，構成されるゲートのうち CN ゲートと制御 V ゲート，CN ゲートと制御 V† ゲートの組がプリミティブであるので量子コストは 5 である．半加算器の量子コストは，図 1(g) より，CCN ゲートと CN ゲートが 1 つずつあることから 6 である構成と，図 1(h) より，プリミティブ量子ゲート 4 つで出来ることから 4 である構成がある．本稿では，図 1(h) の構成を取るため半加算器の量子コストは 4 とする．

3.4　ancilla ラインとゴミライン

ancilla ラインとは，入出力が定数となっているラインのことをいう．ゴミラインとは，計算結果には含まれない不要な情報をもつラインにおいて，入力もしくは出力が変数となっているラインのことをいう．半加算器を例にとる．上から 1 番目のラインは，入出力はともに $|x\rangle$ であり，出力の値は計算結果に含まれないことからゴミラインとなる．また，3 番目のラインの入力ビット $|0\rangle$ は 0 に初期化された量子ビットであることから定数である．

本稿では入出力が共に $|0\rangle$ となっている ancilla ラインのみ，入力は $|0\rangle$ で定数であり出力が変数となっているゴミラインのみを考える．ゴミ出力の変数は g_i と記す．

4　既存方式

本章では，2 章で扱った既存方式の内，[2, 3] の具体的な説明をする．なお，入力ビット列を A, B, それらの和である出力ビット列を S, 桁上げビット列を C とし，A, B, S, C の最下位から i 番目のビットをそれぞれ a_i, b_i, s_i, c_i と表す ($i \in \mathbb{Z}, i \geq 0, c_0 = 0$).

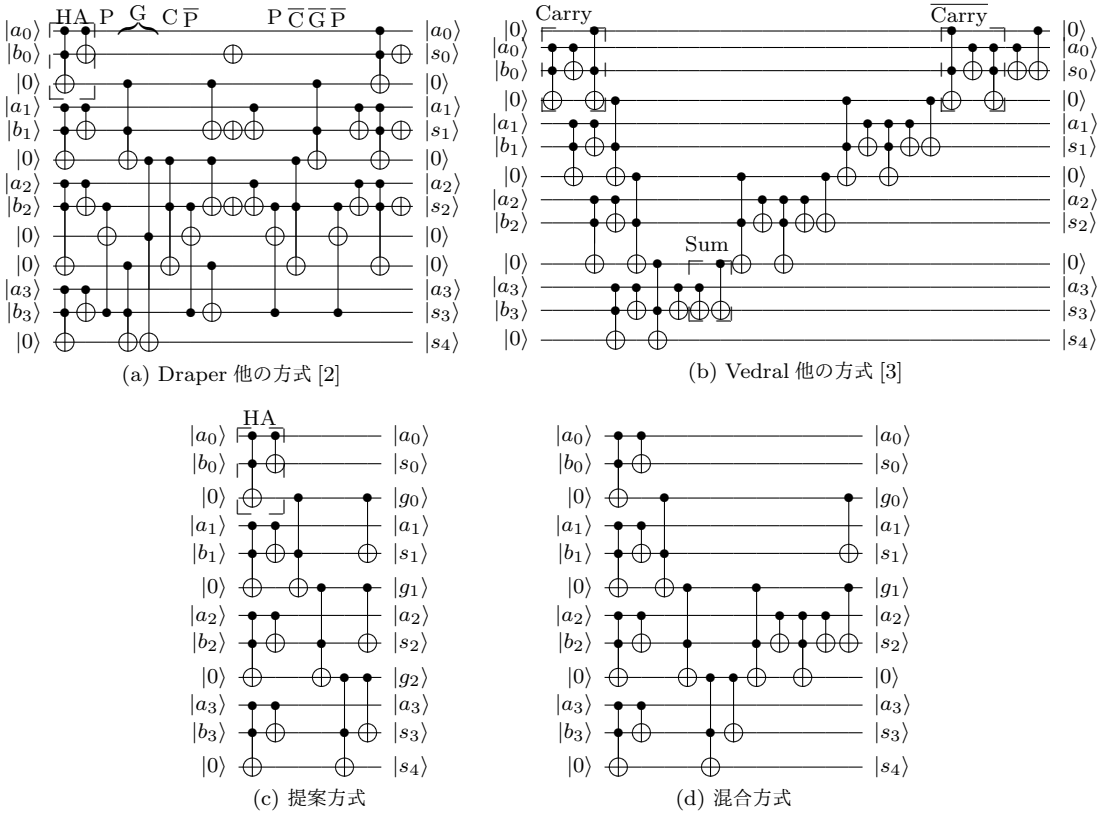
4.1　Draper 他の方式

Draper 他 [2] は，in-place と out-of-place の n ビットの加算器の方式を提案した．ここでは，b_i を入力するラインにおいて和 s_i を出力する in-place の方式を説明する．

図 2(a) は，入力が 4 ビット のときの Draper 他の方式 [2] である．この加算器は全部で 5 段階から構成されている．第 1 段階では，各 i に関して a_i と b_i の和 s_i と桁上げ c_i を半加算器で得る．第 2 段階では，桁上げ情報を二分木状に伝播させる．このとき桁上げ先見加算器における伝播と生成を P, G, C の 3 つの round により行う．P-round は，$i \geq 2$ の第 1 段階で得られた i 番目の和と $i+1$ 番目の和から桁上げ情報を計算し，その値を ancilla ラインに保存する．G-round は，$i \geq 0$ の第 1 段階で得られた i 番目の桁上げ情報と $i+1$ 番目の和から桁上げ情報を計算し，その値を ancillae ラインに保存する．C-round は，P-round, G-round で保存された桁上げ情報から最終的な桁上げ情報を計算する．ここで，$\overline{\mathrm{P}}$, $\overline{\mathrm{G}}$, $\overline{\mathrm{C}}$ はそれぞれ P, G, C の round の逆計算を行う．第 3 段階では，第 2 段階で得られた桁上げ情報を元に最終的な和を計算する．第 4 段階では，第 2 段階の逆計算をして ancilla ラインをゼロクリアする．第 5 段階では，不要な情報を含む ancilla ラインを 0 にする．

4.2　Vedral 他の方式

図 2(b) は，入力が 4 ビット のときの Vedral 他の方式 [3] である．この加算器は全部で 3 段階から構成されている．第 1 段階では，各 i に関して，a_i と b_i の桁上げ c_i を計算し，ancilla ラインに保存する (図 2(b) における Carry)．図を入れたら追加．c_i と a_{i+1} と b_{i+1} から c_{i+1} を計算し，次の ancilla ラインに保存する．この流れを最上位ビットまで行う．第 2 段階では，最上位ビットに対して CN ゲー



(a) Draper 他の方式 [2]　(b) Vedral 他の方式 [3]

(c) 提案方式　(d) 混合方式

図 2　4 量子ビット全加算器（ただし，$A + B = S$ である）

トを適用した後，第 1 段階で得られた桁上げ情報を元に，和を計算する（図 2(b) における Sum）．第 3 段階では，最上位ビット以外のビットに対して，第 1 段階の逆計算を行い ancilla ラインをゼロクリアする（図 2(b) における $\overline{\text{Carry}}$）．その後，第 1 段階で得られた桁上げ情報を元に，和を計算する．

5　提案方式

ゴミラインを使って量子桁上げ伝播加算器の深さを減少させる．このアプローチは，古典的な桁上げ伝播方式の加算器において途中で消去される全ビットを記憶する．これは古典的な加算器の “埋込み” といえる．

5.1　構成方法

提案方式の加算器は 2 段階で構成されている：

1. a_0 と b_0 の和 s_0 と桁上げ c_1 を半加算器で得る．
2. $i \geq 1$ のビットに対して，全加算器 [8] を置く（図 3(a)）．ここで，それぞれの半/全加算器の桁上げビットが入力となるように次の全加算器が置かれる．

提案方式は CN ゲートと CCN ゲートのみを用いる．

全加算器は，$|x\rangle$ と $|y\rangle$ は入力ビット，$|0\rangle$ は 0 に初期化された量子ビット，$|c\rangle$ は一桁前からの桁上げ情報である．これらを全加算器に通すことで，$|c'\rangle$, $|x'\rangle$, $|y'\rangle$, $|z'\rangle$ が得られる．$|c'\rangle$, $|x'\rangle$ はそれぞれ，$|c\rangle$, $|x\rangle$ の値をそのまま出力，$|y'\rangle$ は $|x\rangle$ と $|y\rangle$ と $|c\rangle$ の和を出力，$|z'\rangle$ は $|x\rangle$ と $|y\rangle$ と $|c\rangle$ の桁上げ情報を出力する．また，この全加算器は図 3(b) よりプリミティブ量子ゲート 8 つで構成できるため，量子コストは 8 である．

図 2(c) は，入力が 4 ビット のときの提案方式である．埋込みのためのラインは，入力とは関係がない $|0\rangle$ のラインを用意し，そのラインに桁上げ情報を記憶させた．桁上げ情報を消去せず記憶させたことにより，この回路は可逆である．また，提案方式はゴミラインを許すため，桁上げ情報を記憶させるために使用したラインをゼロクリアしていない．そのため，ゼロクリアするための逆計算を行う必要がなく，その分の回路の量子コスト，ゲート数，深さが削減される．

量子コスト，ゲート数，深さ，ゴミライン数に関しては，第 1 段階において，半加算器のみを置くため量子コストは 4，ゲート数は 2，深さも 2 である．第 2 段階では全加算器を置くため量子コストは 8，ゲート数は 4 ずつ増えていく．深さとゴミライン数においては 1 ずつ増えていく．したがって，n ビットのとき，量子コストは $10n-6$，ゲート数は $4n-2$，深さは $n+2$，ゴミライン数は $n-1$ となる．

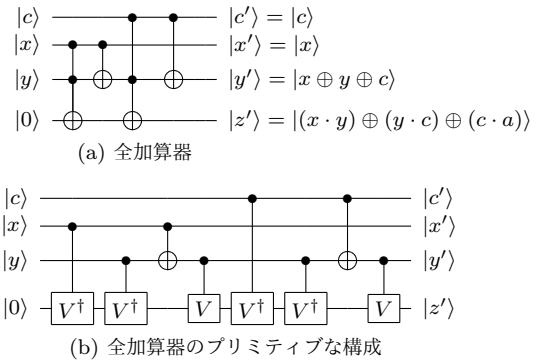
5.2　提案方式の評価

提案方式と既存方式 [2, 3] の量子コスト，ゲート数，深さ，ancilla ライン数，及びゴミライン数は表 1の通りである．

入力数が大きい場合に提案方式は，係数比較した場合，量子コストに関しては [2] より 85.7%，[3] より 66.6% 減少し，ゲート数に関しては [2] より 60.0%，[3] より 50.0%

表 1 in-place な量子加算器回路のコストの比較

量子	Draper 他 [2] ($n \geq 7$)					Vedral 他 [3] ($n \geq 3$)					提案方式 ($n \geq 2$)				
ビット数	量子コスト	ゲート数	深さ	ancilla ライン数	ゴミライン数	量子コスト	ゲート数	深さ	ancilla ライン数	ゴミライン数	量子コスト	ゲート数	深さ	ancilla ライン数	ゴミライン数
1	4	2	2	0	0	12	6	6	1	0	4	2	2	0	0
2	21	9	7	1	0	32	14	13	2	0	12	6	4	0	1
3	40	19	11	2	0	52	22	20	3	0	20	10	5	0	2
4	85	34	18	4	0	72	30	24	4	0	28	14	6	0	3
n	$56n - o(\log n)$	$10n - o(\log n)$	$O(\log n)$	$2n - o(\log n)$	0	$24n - 10$	$8n - 2$	$4n + 8$	n	0	$8n - 4$	$4n - 2$	$n + 2$	0	$n - 1$



(a) 全加算器

(b) 全加算器のプリミティブな構成

図 3 1 量子ビット全加算器

減少した．一方，深さに関しては [3] から n の係数が減少したが，[2] よりも増加した．これは古典的にも桁上げ先見方式の深さが桁上げ伝播方式の深さよりも漸近的に優れていることによる．入力が 1–4 ビットの場合に提案方式は，ゴミライン数以外のすべての指標で既存方式 [2, 3] を上回ることはなかった．したがって入力ビット数が小さいときにも本手法が有効な場合があるといえる．

5.3 混合方式

次に提案方式に既存方式を組込むことを考える．入力ビット数が 4 ビットかつ深さが 15 以内でゴミライン数を 2 まで許す制約があった場合，提案方式に既存方式 [3] を組み込むことで図 2(d) のように構成することができる．図 2(d) では，深さが 11，ゴミライン数が 2 となり，上記の制約を満たす．これは，既存方式 [3] 単体で用いるより深さの点において最適化された．

6 結果

回路の深さに焦点を当てた際，既存方式と提案方式の漸近的な複雑さの解析から，提案方式は入力ビット数が大きいときは既存方式 [3] より最適となり，入力ビット数が小さいとき既存方式 [2, 3] より最適となった．このことからゴミライン数と深さはトレードオフ関係にあるといえる．

また 5.3節から，既存方式の中に提案方式を組込むことにより，ある制約のもとでは既存方式を単体で用いるより，最適化された回路を構成することができる．

7 おわりに

既存方式とトレードオフ関係にある提案方式は，制約によって既存方式よりも最適化できることがある．今回，ゲート数や深さなどの制約に応じて効率的な回路を得る一般的方法，及び量子回路の自動化は明らかにはできなかった．しかし，既存方式を単体で用いるより，ある制約のもと提案方式と組み合わせることでより最適化された回路を設計できることは示せた．このことから，量子コストやゲート数など他の制約を解析し，より多くのトレードオフ関係を見つけることで量子回路の自動化へ応用できると考えられる．

また，制約に応じた効率的な回路の一般的方法の提案及び入力ビット数の小さい場合のコストをさらに解析することは今後の課題である．

参考文献

[1] Fredkin, E. and Toffoli, T.: Conservative logic, *Int. J. Theor. Phys.*, Vol.21, No.3&4, pp.219–253 (1982).

[2] Draper, T.G., Kutin, S.A., Rains, E.M., et al.: A Logarithmic-Depth Quantum Carry-Lookahead Adder, *Quantum Information and Computation*, Vol.6, No.4&5, pp.351–369 (2006).

[3] Vedral, V., Barenco, A. and Ekert, A.: Quantum networks for elementary arithmetic operations, *Phys. Rev. A*, Vol.54, No.1, pp.147–153 (1996).

[4] Van Rentergem, Y. and De Vos, V.: Optimal Design of A Reversible Full Adder, *Int. J. Unconv. Comput.*, Vol.1, pp.339–355 (2005).

[5] Mogensen, T.Æ.: Reversible In-Place Carry-Lookahead Addition with Few Ancillae, *RC 2019*, LNCS, Vol.11497, pp.224–237 (2019).

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

[7] Takahashi, Y. and Tani, S.: Power of Uninitialized Qubits in Shallow Quantum Circuits, *STACS 2018*, LIPIcs, Vol.96, No.57, pp.1–13 (2018).

[8] Golubitsky, O. and Maslov, D.: A Study of Optimal 4-Bit Reversible Toffoli Circuits and Their Synthesis, *IEEE Trans. Comput.*, Vol.61, No.9, pp.1341–1353 (2012).