2023 年度　修士論文

量子測定を用いた初期状態依存の量子回路最適化手法の改善と性能評価

(Improvement and performance evaluation of initial-state dependent optimization for quantum circuits using quantum measurement)

東京大学大学院
理学系研究科 物理学専攻
澤田研究室

山崎峻

2024 年 1 月 24 日

概要

古典コンピュータでは指数増大の計算時間が必要な問題を，量子コンピュータでは多項式時間で解ける場合があることが示されている。しかし，現在の量子コンピュータは NISQ (Noisy Intermediate-Scale Quantum) と呼ばれ，ノイズの影響が大きく量子ビット数も限られている。ビット数を増やし誤り訂正を行うことが目指されるが，元々の誤りを低減させることも主要な研究課題の一つである。このうち，特に CNOT ゲートはノイズの影響を大きく受け，エラー率低減のボトルネックとなっている。本研究では，回路に存在する CNOT ゲートの数を減らす手法である量子回路最適化プロトコル Aqcel を改善し実装することで計算精度を向上させた。また，量子回路最適化にかかる時間も短縮することができた。さらにベンチマーク回路を用いて，Aqcel がより実用に近い回路にも応用可能であることを確認した。

目次

第 1 章

序論

1.1 背景

ショアの素因数分解アルゴリズム [1] など一部の問題では，古典計算では指数関数的計算量を必要とするのに対し量子計算 [2] では多項式計算時間で計算可能であることがわかっている。グローバーの探索アルゴリズム [3] でも古典計算で多項式計算量計算量を要するのに対し量子計算ではさらに計算量を下げることができると考えられている。このように，特定の問題に対して量子計算が古典計算に対して計算量の観点で優位である例が報告されている。

しかし，現状誤り訂正の実装された理想的な汎用型量子コンピュータは実現されておらず，ノイズの影響を受けエラー率の高い NISQ (Noisy Intermediate-Scale Quantum) に留まっている。誤り耐性型汎用量子コンピュータを実現するために大量の量子ビットとゲートを用いて任意の精度で誤りを補正する [4, 5] ことが目指されている一方，エラー率そのものを下げることも重要な研究課題である。特に NISQ においては CNOT が主なエラー源となっている [6]。すなわち，CNOT ゲートそのもののエラー率を下げる，または回路中に使われる CNOT ゲート数を減らすことは誤り訂正量子コンピュータの実現を加速させることにつながる。

本研究ではミドルウェア的なアプローチのうちトランスパイル研究に含まれる量子回路最適化に着目することで，回路中の CNOT 数を減らすことを目指した。量子回路最適化手法には大きく二種類の手法がある。初期状態に依存する量子回路最適化と回路の等価性を保つ量子回路最適化であり，図 1.1 にその関係を示した。最適化後に取りうる回路の集合を表した時に，回路の等価性を保つ量子回路最適化後に取りうる回路の集合は，初期状態に依存する量子回路最適化後の回路の集合に包含される。例として，図 1.1 の (c) の回路を示した。 (c) の回路は第 0 量子ビットが状態 $|0\rangle$ であり，第 0 量子ビットを制御ビット，第 1 量子ビットを標的ビットとする CNOT ゲートが作用している回路である。CNOT ゲートは制御ビットが $|0\rangle$ ならば標的ビットに操作をせず，$|1\rangle$ ならば標的ビットに X ゲートをかける量子ゲートである。したがって， (c) の回路の第 0 量子ビットが $|0\rangle$ であるという初期条件を用いれば回路中の CNOT ゲートは消去することができ，図中 (d) の回路と等価になる。これが初期状態に依存する量子回路最適化である。一方，初期状態に依存せず回路の等価性を保った場合，図 1.1 中の (a) の CNOT ゲートは消せない。これが回路の等価性を保つ量子回路最適化である。

本研究では初期状態に依存する量子回路最適化プロトコルである Aqcel(Advancing Quantum

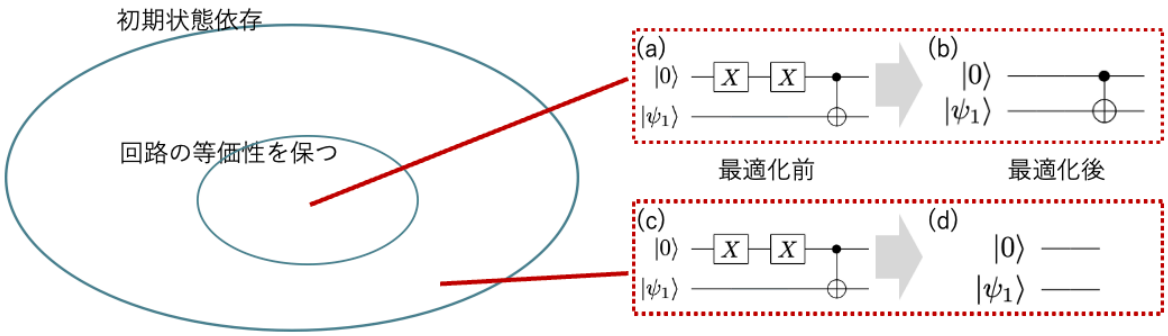

図 1.1　最適化後に取りうる回路の集合 (左図) とその具体例 (右図) 。

Circuit by ICEPP and LBNL) [7] に着目した。AQCEL では制御ゲートが出てくるたびに回路を測定，初期化，再構成および実行を繰り返すことで初期状態に依存した回路最適化を行う。ゆえに，測定エラーなどに由来する精度の低下には改善の余地がある。また最適化そのものにかかる時間も比較的長く，先行研究に比べてより実用的な回路に効果的に適用可能であるかも未検証であった。

1.2　目的

本研究では第 4 章で量子回路最適化プロトコルである AQCEL [7] の改善手法を実装することにより，CNOT の数を減らし計算精度を向上させることを目指す。また，最適化にかかる計算時間も短縮することも目指す。

また，第 5 章で先行研究に比べより実用に近い回路への AQCEL の応用可能性について検証する。

AQCEL を利用して量子計算を行うときには、次のような工程となる。まず最適化すべき量子回路を用意する。次に，AQCEL により量子回路を最適化する。続いてトランスパイルによって量子コンピュータが認識可能な形に変換する。最後に指定した shot 数回分計算を繰り返し状態の確率分布を得る。

なお，本研究では将来的に大きなサイズの問題を量子コンピュータを使って解くことを想定し，最適化の過程も量子計算が古典計算に対して計算量的に優位な過程を想定した。

1.3　構成

第 2 章，第 3 章では本研究を理解する上で必要な量子コンピュータに関する知識，および初期状態に依存する量子回路最適化プロトコル AQCEL について説明する。第 4 章では AQCEL の改善手法 Skip-measurement を実装する。また，実機で Skip-measurement を用いた場合と用いなかった場合のベンチマーク回路における計算精度と最適化にかかった計算時間を比較する。第 5 章では AQCEL の応用可能性について検証する。具体的には AQCEL を用いて実用に近い条件の QPS を最適化した場合の回路と特定の条件に特化して設計された QPS シミュレーション回路を最適化した場合との CNOT 数を比較し，実用に近い条件でも AQCEL が効果的であることを確認する。第 6 章では本研究をまとめ考察するとともに将来の展望について述べる。

第 2 章

量子コンピュータ

2.1 量子回路

2.1.1 量子ビット

量子ビットは量子情報の最小単位である。本研究で用いた IBM の超伝導量子コンピュータでは標準で二準位量子系を用いており，基底 $|0\rangle$ および $|1\rangle$ で一般の状態を表現する。つまり，

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,(\alpha, \beta \in \mathbb{C}) \tag{2.1}$$

$$\text{ただし} \alpha^2 + \beta^2 = 1 \tag{2.2}$$

また，絶対位相 $e^{i\theta}$ のずれを除き等しいベクトルは同じ物理状態に対応するとみなす。すなわち，

$$e^{i\theta}\,|\psi\rangle \sim |\psi\rangle\,(\forall\theta \in \mathbb{R}) \tag{2.3}$$

このような量子状態は半径 1 の球上の点で表される。すなわち，

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle\,(\theta, \phi \in \mathbb{R}) \tag{2.4}$$

ここで，$e^{i\phi}$ を相対位相と呼ぶ。量子状態をブロッホ球 (図 2.1) 上の点として表現する。ブロッホ球において，表 2.1 のように定義する。z 軸正方向を $|0\rangle$，負方向を $|1\rangle$ と定義する。式 2.4 における θ, ϕ はぞれぞれ状態ベクトル $|\psi\rangle$ が z 軸となす角度，状態ベクトル $|\psi\rangle$ の xy 平面への射影が x 軸となす角度に対応する。また，状態ベクトル $|\psi\rangle$ は大きさ 1 のベクトルである。

2.1.2 量子レジスタ

量子ビットを複数並べたものを量子レジスタと呼ぶ。各量子ビットの状態をテンソルで掛け合わせて

$$|0\rangle \otimes |0\rangle \otimes |0\rangle \otimes \cdots =: |000\cdots0\rangle \tag{2.5}$$

$$=: |0\rangle\,(2\text{ 進数表記}) \tag{2.6}$$

と表す。また，式 2.5 において右辺，左辺それぞれの右側から順に第 0 量子ビット，第 1 量子ビット，$\cdots$ と定義する。式 2.6 の記法を用いて一般の 2^n 次元の状態ベクトル $|\psi\rangle$ を

$$|\psi\rangle = c_0\,|0\rangle + c_1\,|1\rangle + \cdots + c_{2^n-1}\,|2^n - 1\rangle \tag{2.7}$$

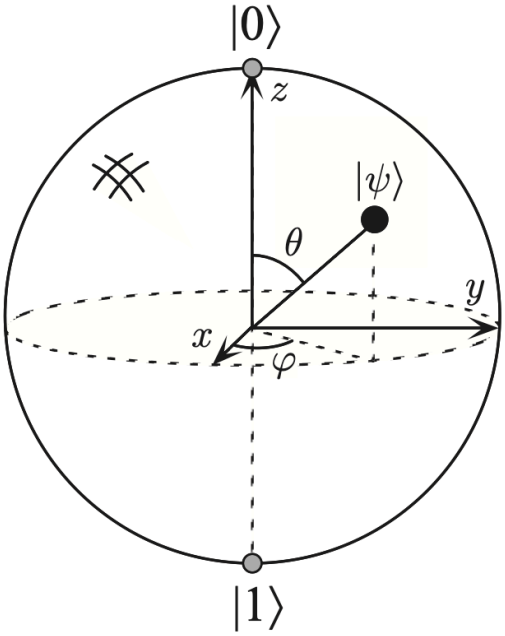

図 2.1　ブロッホ球 [2]。

軸	方向	状態	$\|0\rangle, \|1\rangle$ 基底での表現
z 軸	正	$\|0\rangle$	$\|0\rangle$
	負	$\|1\rangle$	$\|1\rangle$
x 軸	正	$\|+\rangle$	$\frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$
	負	$\|-\rangle$	$\frac{\|0\rangle-\|1\rangle}{\sqrt{2}}$
z 軸	正	$\|0_y\rangle$	$\frac{\|0\rangle+i\|1\rangle}{\sqrt{2}}$
	負	$\|1_y\rangle$	$\frac{\|0\rangle-i\|1\rangle}{\sqrt{2}}$

表 2.1　ブロッホ球の軸と状態の対応。

のように表現する。また，現在標準的に使われている IBM の超伝導量子コンピュータにおいては，量子計算の際は全ての状態を $|0\rangle$ に初期化し，初期状態 $|0\rangle$ から計算を開始する。

2.1.3　量子ゲート

量子レジスタに対する操作を量子ゲートと呼ぶ。量子ゲートには，量子ビットに量子ゲートを作用させた後の状態も正規化条件

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,(\alpha, \beta \in \mathbb{C}) \tag{2.8}$$

$$\text{ただし} \alpha^2 + \beta^2 = 1 \tag{2.9}$$

ゲート	説明	行列表示	例
X	x 軸周りの π 回転	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$X\|0\rangle = \|1\rangle$ $X\|1\rangle = \|0\rangle$
Y	y 軸周りの π 回転	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$Y\|0\rangle = i\|1\rangle$ $Y\|1\rangle = -i\|0\rangle$
Z	z 軸周りの π 回転	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$Z\|0\rangle = \|0\rangle$ $Z\|1\rangle = -\|1\rangle$
$R_x(\theta)$	x 軸周りの θ 回転	$\begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$	$R_x(\theta)\|0\rangle = \cos\frac{\theta}{2}\|0\rangle - i\sin\frac{\theta}{2}\|1\rangle$ $R_x(\theta)\|1\rangle = -i\sin\frac{\theta}{2}\|0\rangle + \cos\frac{\theta}{2}\|1\rangle$
$R_y(\theta)$	y 軸周りの θ 回転	$\begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$	$R_y(\theta)\|0\rangle = \cos\frac{\theta}{2}\|0\rangle + \sin\frac{\theta}{2}\|1\rangle$ $R_y(\theta)\|1\rangle = \cos\frac{\theta}{2}\|0\rangle - \sin\frac{\theta}{2}\|1\rangle$
$R_z(\theta)$	z 軸周りの θ 回転	$\begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$	$R_z(\theta)\|0\rangle = e^{-i\frac{\theta}{2}}\|0\rangle$ $R_z(\theta)\|1\rangle = e^{i\frac{\theta}{2}}\|1\rangle$
H (アダマールゲート)	$\frac{1}{\sqrt{2}}(\|0\rangle + \|+\rangle)$ 軸周りの π 回転	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$H\|0\rangle = \frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$ $H\|1\rangle = \frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$

表 2.2　主な単一量子ゲート。

を満たす必要があるという制約条件がある。したがって，複素数ベクトル空間であることに注意すると，量子ゲートはユニタリ演算子であることがわかる。

主な量子ゲートを表 2.2, 2.3 に示した。なお，説明にブロッホ球 (図 2.1) を用いた。ただし，軸についての回転は反時計まわりを正方向とし，グローバル位相の違いは無視した。特に，表 2.3 中の CNOT ゲートは制御ビットが $|0\rangle$ ならば標的ビットに操作をせず，$|1\rangle$ ならば標的ビットに X ゲートをかける量子ゲートである。

2.1.4　量子回路

量子レジスタに量子ゲートをかけ表したものを量子回路と呼ぶ。図 2.2 に $n = 5$ ビットの量子回路の例を示す。本研究では上から順に第 0 量子ビット，第 1 量子ビット，$\cdots$，第 n 量子ビットと定義する。IBM の超伝導量子コンピュータに倣い，初期状態を全ての量子ビットで $|0\rangle$ から開始した。本研究でも，以降これに倣う。それぞれの量子ビットで左から右に時刻が進行する。例えば第 0 量子ビットでは最初に H ゲートを作用させた後，次に CNOT ゲート (制御ビット) が作用する。この場合，制御ゲートの制御ビットでは量子ビットの状態を変化させないということに注意すると，第 0 量子ビットの終状態は $R_z(\phi)SR_x(\theta)H|0\rangle$ となる。また，図 2.2 では左詰めにせずゲートを表記しているが，実際の量子コンピュータにおける計算では量子回路図上で縦並びのゲートが同時刻に作用するとは限らない。むしろ，実行可能なゲートから順に，即ち図 2.2 では左詰めにゲートが作用していくことが多い。また，各ゲートを超伝導量子コンピュータ上で実行するパルス時間もゲートや条件により異なるため，量子回路の横軸は必ずしも実際の量子計算における時刻を正確に表していないことに注意が必要である。

ゲート	説明	行列表示
C_x^{ij} または CNOT	制御ビット i の状態に応じ標的ビット j に X を作用させる	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
C_y^{ij}	制御ビット i の状態に応じ標的ビット j に Y を作用させる	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{pmatrix}$
C_z^{ij}	制御ビット i の状態に応じ標的ビット j に X を作用させる	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$
$C_{R_x}^{ij}$	制御ビット i 状態に応じ標的ビット j に R_x を作用させる	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ 0 & 0 & -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
$C_{R_y}^{ij}$	制御ビット i の状態に応じ標的ビット j に R_y を作用させる	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ 0 & 0 & \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
$C_{R_z}^{ij}$	制御ビット i の状態に応じ標的ビット j に R_z を作用させる	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{-i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$
CC_x^{ijk}(トフォリ)	制御ビット i,j の状態に応じ標的ビット k に X を作用させる	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$
$MC_x^{ij\cdots k}$	制御ビット $i,j,\cdots$ の状態に応じ標的ビット k に X を作用させる	$\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & 0 & 1 \\ 0 & \cdots & \cdots & 1 & 0 \end{pmatrix}$

表 2.3　主な二量子ゲート，多量子ゲート。

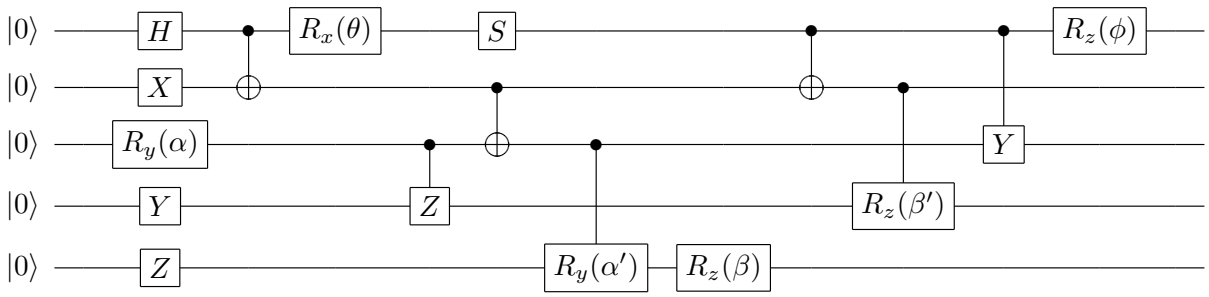

図 2.2　量子回路の例。

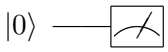

図 2.3　$|0\rangle$ の状態が測定される量子回路。

2.1.5　測定

終状態から情報を読み出す操作を測定という。状態

$$|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle + \cdots + c_{2^{n-1}} |2^n - 1\rangle \tag{2.10}$$

に対し Z 基底の測定では状態 $|k\rangle$ を確率 $p_k = |c_k|^2$ で得られることを利用し，測定を繰り返し状態 $|0\rangle, |1\rangle, \cdots, |2^{n-1}\rangle$ を測定した回数をヒストグラムとして記録することで $|c_0|^2, |c_1|^2, \cdots, |c_{2^{n-1}}|^2$ を推定する。

2.2　statevector simulator

第 2.1 節で説明した量子回路を実際に実行する際には，IBM の超伝導方式量子コンピュータだけでなく，古典コンピュータ上で理想的な状態ベクトルを計算し比較する方法がよく用いられる。本節ではこのような古典コンピュータ上での計算に用いられる statevector simulator について説明する。なお，本研究で用いた Aer Simulator についてのみ説明する。Aer Simulator では測定時に変数で指定した shot 数の回数だけ測定を繰り返しヒストグラムを作成する。デフォルトでは，実際の量子コンピュータでの量子的なゆらぎは考慮されず，理想的な分布を返すように設定されている。ただし，統計エラーは考慮される。具体例を用いて説明する。図 2.3 の回路の場合，shot 数=1024 では 1024 回，$|0\rangle$ が観測される。一方，図 2.4 の回路の場合，shot 数を $m = 1024$ として，ヒストグラム上で $|0\rangle$ 状態が m_0 回観測される確率 $p(m_0)$ は，

$$p(m_0) = {}_mC_{m_0} \left(\frac{1}{2}\right)^{m_0} \left(\frac{1}{2}\right)^{m-m_0} \tag{2.11}$$

$$= {}_mC_{m_0} \left(\frac{1}{2}\right)^{m} \tag{2.12}$$

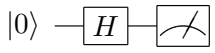

図 2.4　$\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle$ の状態が測定される量子回路。

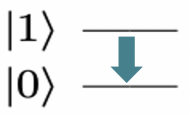

図 2.5　縦緩和。

と表される。

2.3　量子エラー

実機上で量子計算を行う上では必ずエラーが発生する。本節では代表的な量子エラーについて説明する。

2.3.1　初期化エラー

実機の量子コンピュータにおいて初期状態に戻す操作の際，望ましい状態 $|0\rangle$ に精度良く初期化されない場合がある。これを初期化エラーと呼ぶ。

2.3.2　デコヒーレンス

デコーヒーレンスには縦緩和と横緩和がある。縦緩和は量子ビットの $|0\rangle$，$|1\rangle$ の間の変化を指し，緩和時間 T_1 で特徴づけられる。$|1\rangle$ 状態にある量子ビットは時定数 T_1 の指数関数で $|0\rangle$ に縦緩和する。自然放出などの物理現象が縦緩和の要因である (図 2.5) 。横緩和は磁化の横軸成分 $M(t)$ が時刻 t に伴い指数関数的に減衰していく過程のことである。緩和時間 T_2 を用いて

$$M(t) = M(0)e^{\frac{t}{T_2}} \tag{2.13}$$

と表される。

ここで，$T_1 \gg T_2$ であるため横緩和がより重要である。量子コンピュータにおける精度の高い計算においてはこれらのデコヒーレンスを最小限に抑え，コヒーレンス時間を最大化することが求められ，量子誤り訂正や量子ビット設計の最適化，および低温技術などによって実現が目指されている。

2.3.3　ゲートエラー

量子ゲートを量子ビットに作用させた時に，意図しない操作となる場合がある。これをゲートエラーと呼ぶ。

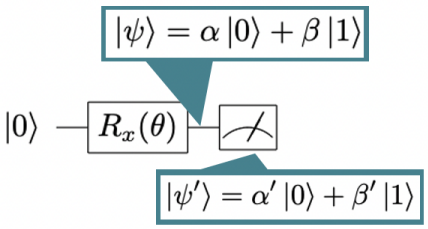

図 2.6　測定エラーの概要。

2.3.4　測定エラー

測定をする際，量子ビットの情報を正しく読み取れないことがある。これを測定エラーと呼ぶ。図 2.6 に測定エラーの概要を示す。図 2.6 中では真の状態 $|\psi\rangle$ が測定エラーにより偽の状態 $|\psi'\rangle$ として観測されてしまう状況を表している。

2.4　忠実度

量子状態の距離を定量化する測度の一つに忠実度がある。状態 ρ, σ の間の忠実度 $F(\rho, \sigma)$ は以下の式で定義される。

$$F(\rho, \sigma) \coloneqq \mathrm{tr}\sqrt{\rho^{\frac{1}{2}}\sigma\rho^{\frac{1}{2}}} \tag{2.14}$$

一般に量子コンピュータでの測定は z 基底で行う。よって，z 基底での測定のみで得られた確率分布 p^ρ, p^σ を用いて表した忠実度を古典忠実度 (classical fidelity) と呼ぶ。古典忠実度 $F'(\rho, \sigma)$ は以下の式で表される。ただし，k は量子ビット列のノーテーションである。

$$F'(\rho, \sigma) \coloneqq \left(\sum_k \sqrt{p_k^\rho p_k^\sigma}\right)^2 \tag{2.15}$$

第 3 章

量子回路最適化プロトコル AQCEL

本章では初期状態に依存した量子回路最適化プロトコルである AQCEL(Advancing Quantum Circuit by ICEPP and LBNL) [7] について紹介する。AQCEL では主に制御ゲートが出てくるたびに制御ビットを測定することで不要な制御ゲートを削除する。第 3.1 節では AQCEL の原理について説明する。 第 3.2 節ではこれまでの研究で明らかになった AQCEL の性能について説明する。

3.1 AQCEL の原理

本節では最初に AQCEL の全体像について概要を見た上で，それぞれの手順について順に詳細に説明する。AQCEL は以下の手順で最適化を行なっている。

1. 多制御ゲートの分解
2. 隣り合うゲートの消去
3. 不要なビット制御削除
4. 隣り合うゲートの消去
5. 不要な量子ビットの削除

3.1.1 多制御ゲートの分解

多制御ゲートを分解してトフォリ，CNOT ゲートを用いて分解する (図 3.1) [8, 9, 10]。分解する理由は，第 3.1.3 節で紹介するように AQCEL では制御ビットを測定する段階があり，多制御ゲートに対して全ての制御ビットを測定すると指数回測定になってしまうからである。仮に多制御ゲートの制御ゲートを全て測定すると，制御ビット数が n，shot 数が m であった場合，測定回数 N_{meas} は

$$N_{\mathrm{meas}} = 2^n m \tag{3.1}$$

となる。これは n や m が大きくなると非常に大きくなるので，トフォリ以下に分解して多項式回数の測定にする。CNOT ゲート以下にしない理由は第一に計算時間を短縮するためだ。第二に，トフォリ以下の分解の方がより制御ゲートを削除できる場合があるためだ。図 3.2 に例を示した。AQCEL では図中左の回路のようにトフォリ以下に多制御ゲートを分解する。このような場合，後の第 3.1.3 節での不要な

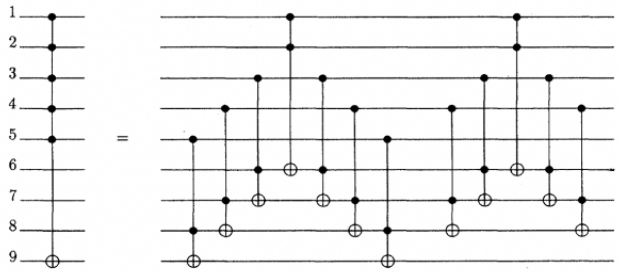

図 3.1　トフォリの分解の例。

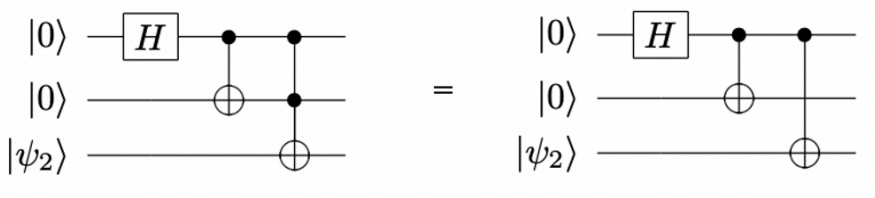

図 3.2　トフォリ以下の分解に優位性がある例。

ビット制御削除の段階で，図中トフォリゲートの第 1 量子ビットに対する制御ビットが削除できる。なぜなら，トフォリゲート直前で第 $0,1$ 量子ビットの状態は $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ であり，もし一方の量子ビットの状態が $|0\rangle$ であればもう片方は $|0\rangle$，$|1\rangle$ ならばもう片方は $|1\rangle$ と測定せずともわかるからだ。

3.1.2　隣り合うゲートの消去

隣り合うゲートで Identity になるものがあれば消す。

3.1.3　不要なビット制御削除

不要なビット制御削除は次の擬似コードの手順で進める。ただし，CC_U^{ijk} ゲートを制御ビットを第 i,j 量子ビット，標的ビットを第 k 量子ビットとする制御 U ゲートとする。第 i,j 量子ビットが $|0\rangle\otimes|0\rangle$，$|0\rangle\otimes|1\rangle$，$|1\rangle\otimes|0\rangle$，$|1\rangle\otimes|1\rangle$ である確率をそれぞれ $|c_{00}|^2, |c_{01}|^2, |c_{10}|^2, |c_{11}|^2$ とする。第 i,j 量子ビット測定時の閾値を ϵ とする。$C_{U'}^{lm}$ ゲートを制御ビットを第 l 量子ビット，標的ビットを第 m 量子ビットとする制御 U' ゲートとする。第 l 量子ビットが $|0\rangle$，$|1\rangle$ である確率を $|c_0|^2, |c_1|^2$ とする。第 l 量子ビット測定時の閾値を ϵ' とする。

for all gate g in circuit **do**
  if g is gate CC_U^{ijk} **then**
    measure control qubits i, j for several times
    if $|c_{11}|^2 > \epsilon$ **then**
      if $|c_{00}|^2 < \epsilon$ and $|c_{01}|^2 < \epsilon$ and $|c_{10}|^2 < \epsilon$ **then**
        turn gate CC_U^{ijk} into U gate on qubit k
      else if $|c_{01}|^2 < \epsilon$ **then**
        turn gate CC_U^{ijk} into gate C_U^{jk}
      else if $|c_{10}|^2 < \epsilon$ **then**
        turn gate CC_U^{ijk} into gate C_U^{ik}
      end if
    else
      delete gate CC_U^{ijk}
    end if
  else if g is gate $C_{U'}^{lm}$ **then**
    measure control qubit l for several times
    if $|c_1|^2 > \epsilon'$ **then**
      if $|c_0|^2 < \epsilon'$ **then**
        turn gate $C_{U'}^{lm}$ into gate U' on qubit m
      end if
    else
      delete gate $C_{U'}^{lm}$
    end if
  end if
end for

図 3.3 に簡単な不要なビット制御削除の例を示す。最初に，CNOT ゲートが出てくる。直前の第 1 量子ビットを繰り返し測定しヒストグラムを得る。$|1\rangle$ が閾値 ϵ よりも小さいので CNOT を削除する。次に，再び CNOT ゲートが出てくる。再び第 1 量子ビットの測定を行い $|0\rangle$ が閾値 ϵ' よりも小さいので CNOT を削除し第 2 量子ビットに X を作用させる。同様に，最後のトフォリについて，第 0 量子ビットを測定し $|0\rangle$，$|1\rangle$ の測定される確率が閾値よりも大きいので第 0 量子ビットの制御ビットは残す。トフォリ直前の第 1 量子ビットでも測定を行い $|0\rangle$ が閾値よりも小さいので制御ビットを削除する。

不要なビット制御削除における測定は制御ビットのみを測定する。第 3.1.1 節での説明と同様，もし仮に全量子ビットを不要なビット制御削除のたびに測定すると仮定すると制御ビット数が n，shot 数が m であった場合，基底は $|0\cdots000\rangle, |0\cdots001\rangle, \cdots |1\cdots111\rangle$ の 2^{n-1} つある。よって測定回数 N_{meas} は

$$N_{\mathrm{meas}} = 2^n m \tag{3.2}$$

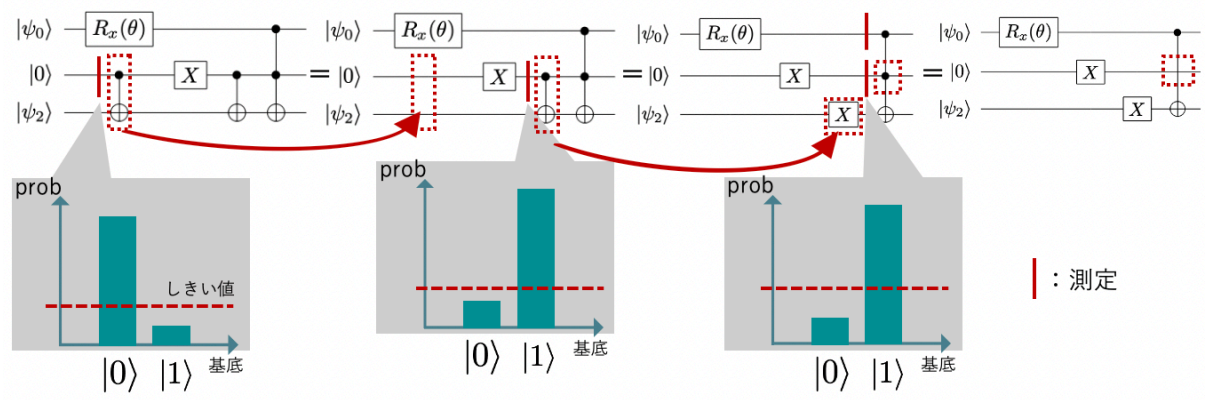

図 3.3　AQCEL による回路最適化の簡単な例。赤線で測定位置を表す。

となってしまう。例えば $n = 3$ 量子ビット，shot 数 $m = 1024$ では 8192 回の測定および回路の初期化，回路の再構成が必要となり非効率である。よって制御ビットのみを測定するように実装されている。

3.1.4　隣り合うゲートの消去

隣り合うゲートで Identity になるものがあれば消す。

3.1.5　不要な量子ビットの削除

何も演算の作用していない量子ビットがあれば消す。

3.2　AQCEL の過去の結果

本節では過去の AQCEL についての先行研究でどのような結果が得られたかを紹介する。

3.2.1　ベンチマーク回路量子パートンシャワーシミュレーション回路

先行研究 [7] ではベンチマーク回路として量子パートンシャワー (Quantum Parton Shower：QPS) シミュレーション回路 (図 3.4) [11] が用いられた。QPS を計算する際の量子回路は任意の初期条件に対応できるように多量子制御ゲートを余分に持たせており，AQCEL により制御ゲートを削除する余地が大きい。また，量子コンピュータの計算では出力の確率分布を得るために，同じ条件での計算を多数回行う。このとき統計誤差を十分小さくするために実行回数が膨大になる場合があり，この意味で一度量子回路最適化をして生成した回路を繰り返し使うことは全体の計算量を下げる可能性がある。

評価に用いた QPS 回路の条件は図 3.5，表 3.1 の通りである。初期状態は $|f_1\rangle$ とした。結合定数は $g_1 = 2$, $g_2 = g_{12} = 1$ とした。ただし，フェルミオン f_1 とボソン ϕ の間の結合定数を g_1，フェルミオン f_2 と ϕ の間の結合定数を g_2，$f_1\bar{f}_2(\bar{f}_1 f_2)$ と ϕ との間の結合定数を g_{12} とした。また，1 つのステップで一つの粒子のみ反応するものとした。

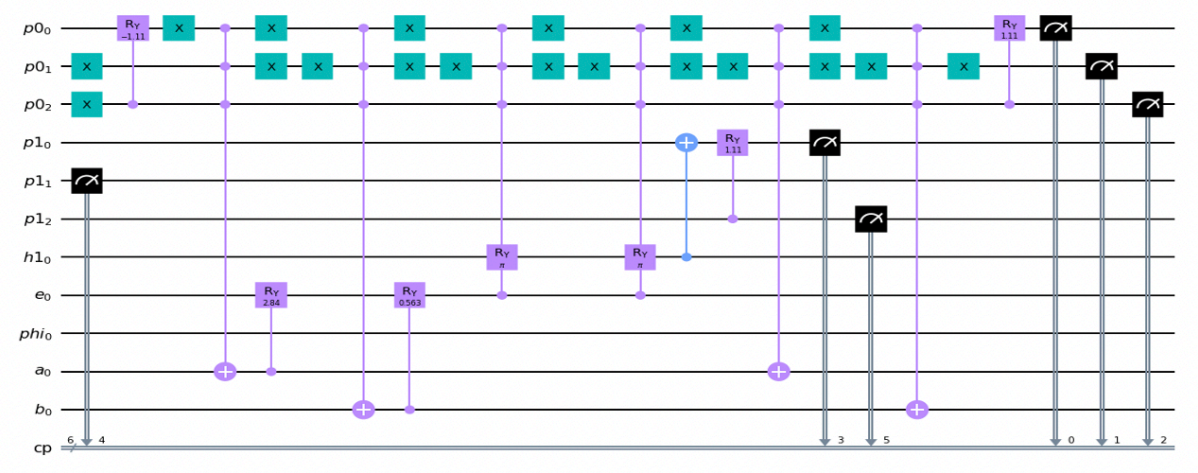

図 3.4　AQCEL適用前の QPS1 ステップシミュレーション回路 [12]。

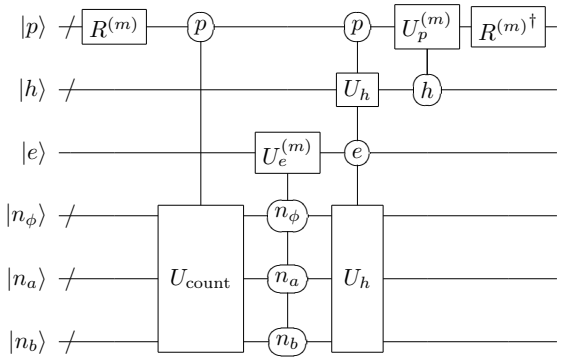

図 3.5　QPS(Quantum Parton Shower) シミュレーション回路 [11]。

量子レジスタ	表す対象	量子ビット数
$\|p\rangle$	粒子の状態	$3(N+n)$
$\|h\rangle$	放出の履歴	$N\lceil\log_2(N+n)\rceil$
$\|e\rangle$	放出が起きたか	1
$\|n_\psi\rangle$	ボソンの数	$\lceil\log_2(N+n)\rceil$
$\|n_a\rangle$	f_a の数	$\lceil\log_2(N+n)\rceil$
$\|n_b\rangle$	f_b の数	$\lceil\log_2(N+n)\rceil$

表 3.1　N ステップ QPS シミュレーション回路におけるレジスタとその意味 [11]。第一列がレジスターのラベル，第二列が表す対象，第三列が初期粒子数が n の場合に使用する量子ビット数を表す。

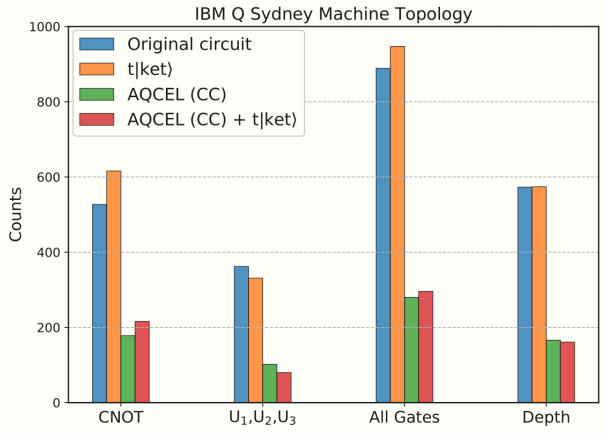

図 3.6　これまでの QPS シミュレーション 1 ステップ回路に AQCEL を適用した結果 [7]。実機は IBM Quantum Sydney Machine を用いている。横軸は CNOT，U_1, U_2, U_3，全てのゲートの合計，深さ (depth)，縦軸はそれらの個数。

3.2.2　先行研究における結果

先行研究 [7] における結果の一部を示す (図 3.6) 。図 3.6 中に示したように，元の回路 (Original circuit) に比べて AQCEL を用いることで (AQCEL (CC)) ベンチマーク回路中の CNOT 数を半分以下に減らすことができると示された。

第 4 章

AQCELの改善手法 Skip-measurement

本章では第 3 章で紹介した AQCEL(Advancing Quantum Circuit by ICEPP and LBNL) の改善手法である Skip-measurement について紹介する。従来の AQCEL では，制御ゲートが出てくるたびに測定する必要があった。したがって，実機の量子コンピュータにおいては回路を初期化したり再構成したりすることに時間を要し，最適化にかかる計算時間が増大する原因となっていた。また，回路の深い部分において量子ビットを測定することになっていたため，ノイズの影響を受けやすくなっていた。Skip-measurement を適用した AQCEL では，初期状態または以前の測定の結果から明らかに状態がわかる量子ビットについて測定を省略した。その結果，ベンチマーク回路とした QPS(Quantum Parton Shower) のシミュレーション回路において測定回数が減少したとともに，特定の閾値 (threshold) 領域において従来の AQCEL よりも精度が向上することがわかった。本章では第 4.1 節で Skip-measurement の原理について説明する。次に， 第 4.2 節で Skip-measurement の性能による AQCEL の性能改善を評価する方法について説明する。最後に，第 4.3 節で性能評価の結果を示した。

4.1 Skip-measurement

Skip-measurement は初期状態や以前の測定の結果を利用することで従来の AQCEL よりも性能を改善させた，AQCEL の改善手法である。Skip-measurement において，単一制御ゲートに対して測定を省略する規則は以下の表 4.1 の通りである。なお，初期状態は IBM の超伝導方式量子コンピュータにおいて標準である，$|0\rangle$ から開始することを想定した。二制御 U ゲート C_U^{ijk} (制御ビットを第 i, j 量子ビット，標的ビットを第 k 量子ビットとする) において第 i, j 量子ビットの前回の測定結果が $|00\rangle, |11\rangle$ のみの重ね合わせ状態の場合に対しては表 4.2 のように例外的に処理した。

Skip-measurement を適用することで図 4.1 のように測定を省略できる。図中左の回路においては制御ビットが出てくるたびに測定が必要であったため，4 ヶ所での測定が必要であった。一方，図中右の回路では初期状態および以前の測定の結果を用いることで，明らかに状態のわかっているビットについては測定を省略し，1 ヶ所を測定するのみで十分になる。

また，Skip-measurement により，図 4.2 のような例においても測定を省略することができる。Skip-measurement によって想定されるメリットは以下の 2 点である。

状況	量子状態	制御ビットに対する操作
(I) 前回の測定結果がない (初期状態から初めて測定する) かつ		
何もかかっていない	$\|0\rangle$	測定を省略
X のみかかっている	$\|1\rangle$	測定を省略
その他	不明	再度測定
(II) 前回の測定結果が $\|0\rangle$ かつ		
何もかかっていない	$\|0\rangle$	測定を省略
X のみかかっている	$\|1\rangle$	測定を省略
その他	不明	再度測定
(III) 前回の測定結果が $\|1\rangle$ かつ		
何もかかっていない	$\|1\rangle$	測定を省略
X のみかかっている	$\|0\rangle$	測定を省略
その他	不明	再度測定
(IV) 前回の測定結果が $\|0\rangle$ と $\|1\rangle$ の重ね合わせ状態かつ		
何もかかっていない	$\|0\rangle$ と $\|1\rangle$ の重ね合わせ状態	測定を省略
X のみかかっている	$\|0\rangle$ と $\|1\rangle$ の重ね合わせ状態	測定を省略
その他	不明	再度測定

表 4.1　制御ゲートに対する Skip-measurement の原理。

状況	量子状態	制御ビットに対する操作
(V) 前回の測定結果が $\|00\rangle, \|11\rangle$ の重ね合わせ状態の場合かつ		
何もかかっていない	$\|00\rangle, \|11\rangle$ の重ね合わせ状態	片方の測定を省略
片方に X のみかかっている	$\|01\rangle, \|10\rangle$ の重ね合わせ状態	片方の測定を省略
両方に X のみかかっている	$\|00\rangle, \|11\rangle$ の重ね合わせ状態	片方の測定を省略
その他	不明	両方を再度測定

表 4.2　二制御 U ゲート C_U^{ijk} に対する Skip-measurement の原理。表中では制御ビットである第 i, j 量子ビットについて記載した。

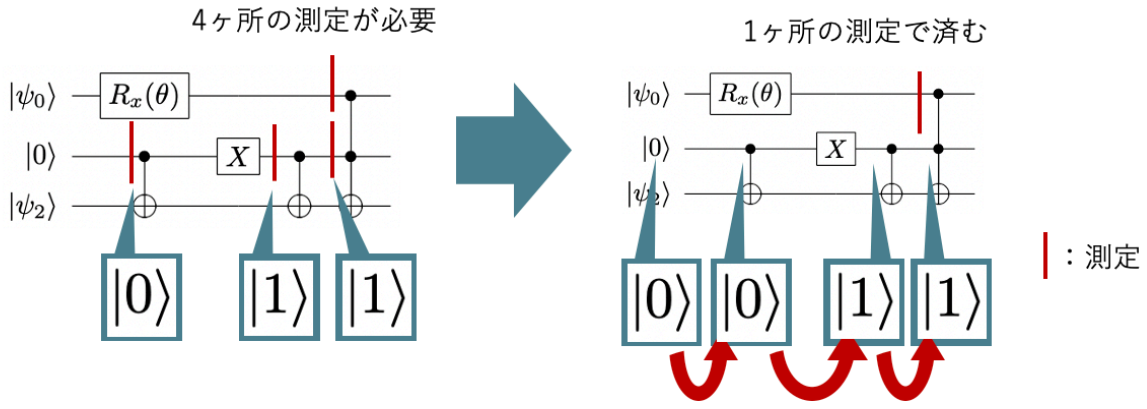

図 4.1　Skip-measurement の原理。左図は従来の AQCEL を回路に適用した例，右図は Skip-measurement を AQCEL を適用した場合の例。

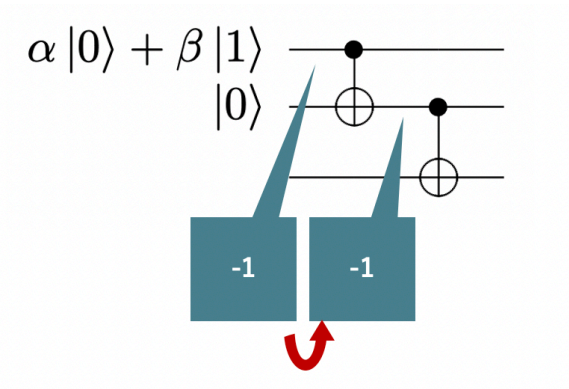

図 4.2　Skip-measurement によって他に測定を省略できる例。「−1」は量子ビットの状態が $|0\rangle$ と $|1\rangle$ の重ね合わせ状態であることを表す。

1. 最適化時間の短縮
2. 忠実度の向上

第一に，最適化時間の短縮について，図 4.3 に比較的長い回路に適用する例を示すことで，実際にどの程度測定を省略できるかをみる。図 4.3 は横磁場イジング模型の回路に従来の AQCEL を適用した場合と Skip-measurement ありの AQCEL を適用した場合の例である。図中上の回路においては従来の AQCEL を適用した場合である。制御ビットが出てくるたびに測定が必要であったため，24 ヶ所ある CNOT ゲートに対してそれぞれ測定をしなけらばならなかった。したがって，測定も 24 ヶ所で必要であった。一方，下図においては初期状態 $|0\rangle$ および以前の測定の結果を用いることで 6 ヶ所，つまり従来の 25% にまで測定回数を減らすことができた。また，再構成した回路のそれぞれの深さ (depth) を合計した深さについて，実機の量子コンピュータでは測定のたびに状態を壊してしまうため再度初期化および回路を再構成しなければならないことに注意すると，上の回路では深さが 372×(shot 数) であるのに対し，下の回路では 77% 削減された 85×(shot 数) の深さであることが計算で求められる。

第二に，忠実度について一般に時系列が後の状態やゲート操作後の状態はノイズの影響を受ける。よって，比較的深さの浅い部分で測定された結果を用いると，実機では最終的な結果の忠実度が向上すると考えられる。

4.2　評価方法

プログラムには表 4.3 のバージョンを用いた。ベンチマーク回路には第 3.2.1 節で紹介した QPS の 1 ステップ回路 (図 4.4) を用いた。また，評価の指標には第 2.4 節で説明した古典忠実度および測定回数を用いた。忠実度について，AQCEL 適用前の QPS1 ステップシミュレーション回路を ibmq_ qasm_ simulator を用いて古典コンピュータで計算して得られた shot 数を 4096 での理想的な確率分布と，AQCEL 適用後の QPS1 ステップシミュレーション回路を量子コンピュータの実機 ibm_ kawasaki で計算して得られた shot 数 4096 での確率分布との間の古典忠実度を，Skip-measurement ありとなしでそれぞれ求めた (図 4.5) 。さらに，同様の操作を閾値を変えて行い，グラフ上にプロットした。

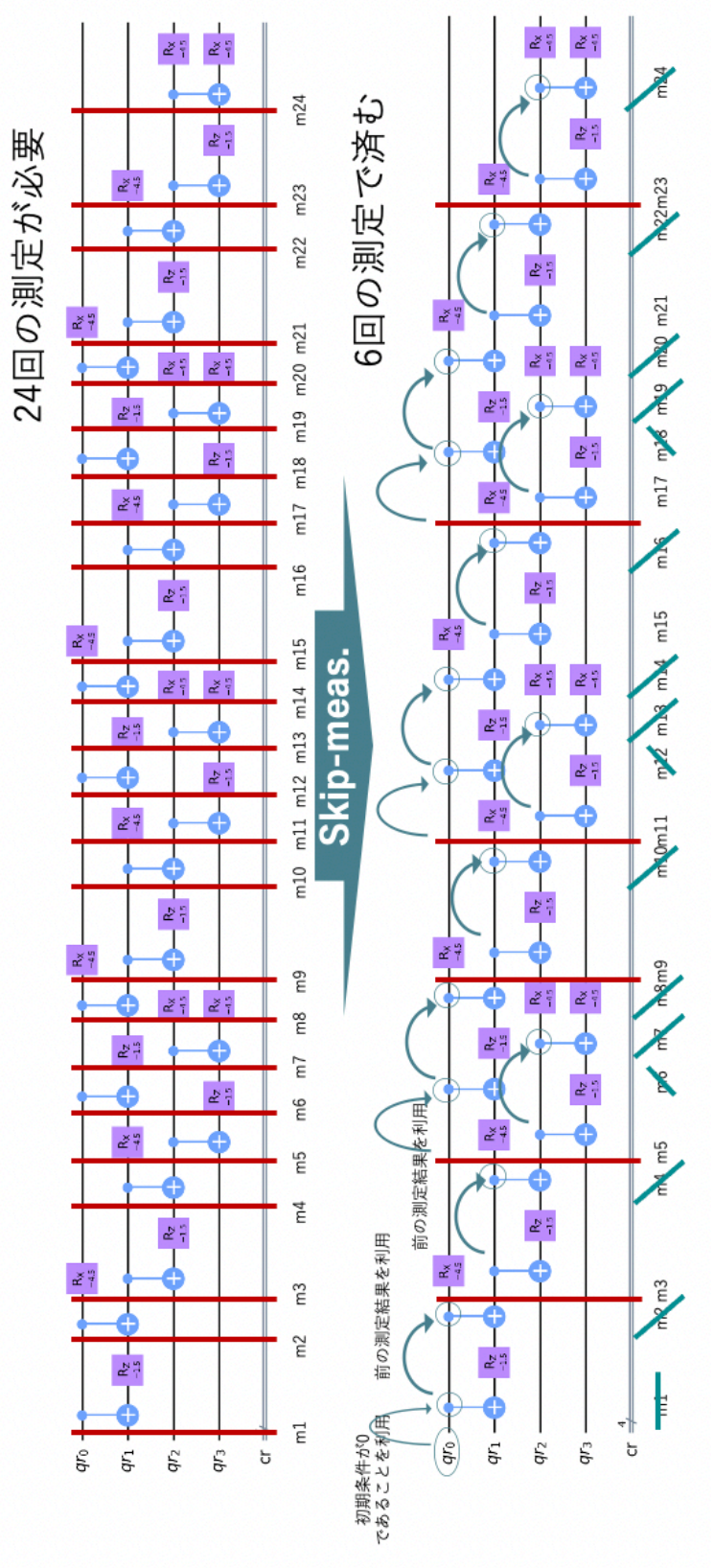

図 4.3　Skip-measurement ありの AQCEL をイジングモデルに適用した例。上の回路は従来の AQCEL を適用した場合の測定位置。下の回路は Skip-measurement ありの AQCEL を適用した例。赤線で測定の位置を示した。また，$m1, m2, \cdots, m24$ で測定の位置をナンバリングした。

	バージョン	引用文献
Python	3.8	[13]
Qiskit	0.21.0	[14]
Terra	0.15.2	
Aer	0.6.1	
Ignis	0.4.0	
pytket	0.6.1	

表 4.3　用いた Python やライブラリのバージョン。

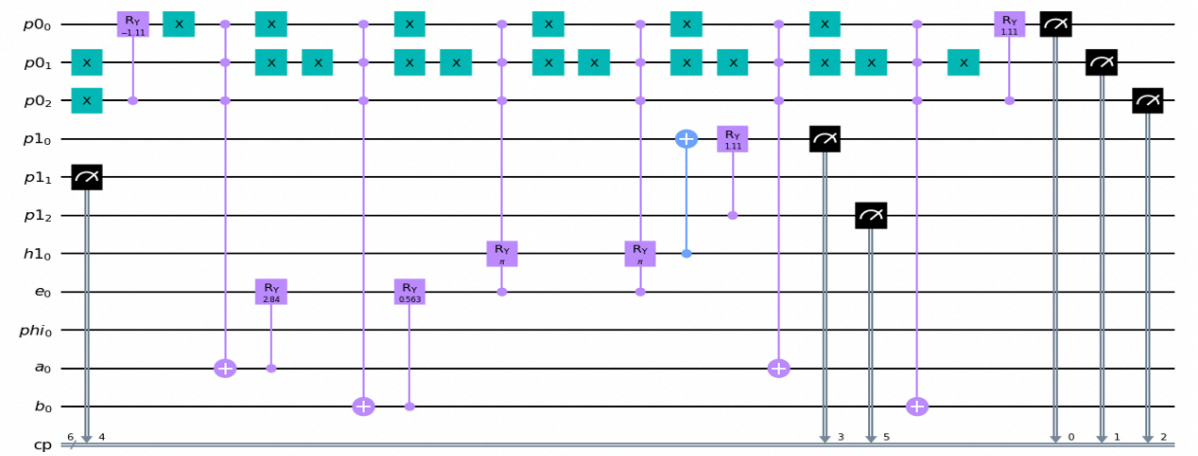

図 4.4　評価に用いた QPS1 ステップのシミュレーション回路。

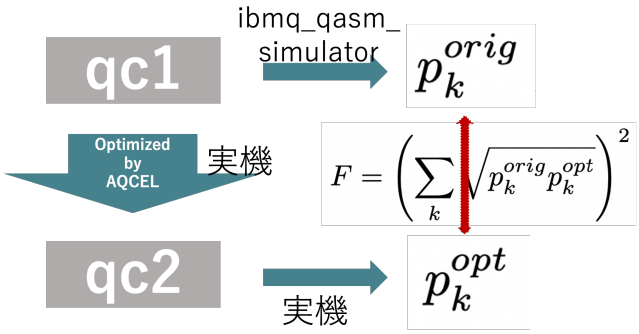

図 4.5　評価に用いた忠実度の条件。qc1 は AQCEL を適用する前の元の回路。qc2 は AQCEL 適用後の回路を表す。

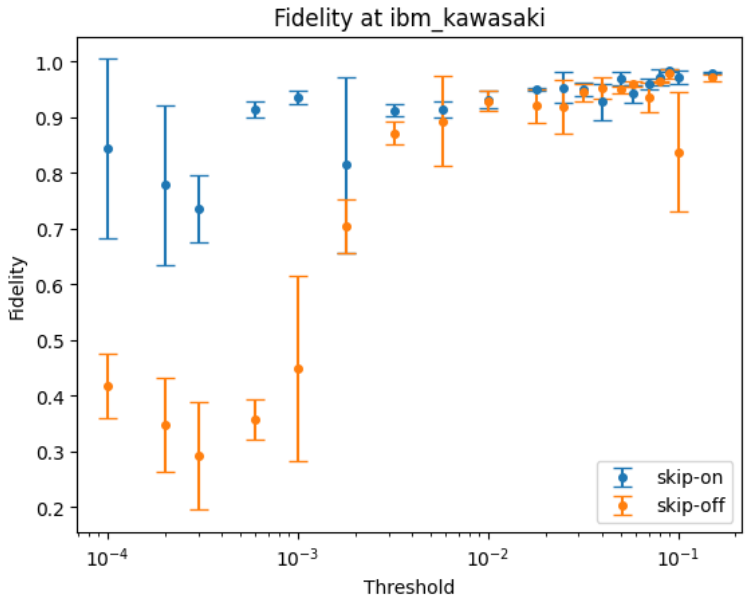

図 4.6　実機 ibm_ kawasaki での QPS1 ステップシミュレーション回路での古典忠実度 (classical fidelity : 縦軸) と閾値 (threshold : 横軸) の関係。「skip on」は Skip-measurement ありの場合，「skip off」は Skip-measurement なしの場合を表す。

4.3　評価

古典忠実度についての結果を示す。図 4.6 ではエラーバーは各点 3 回ずつ実験し，二乗平均平方根：RMS (Root Mean Square) を用いた。忠実度は Skip-measurement ありの AQCEL では，閾値 1.0×10^{-4} から 1.0×10^{-3} の範囲で従来の AQCEL よりも高かった。

測定回数は図 4.7 のようになった。

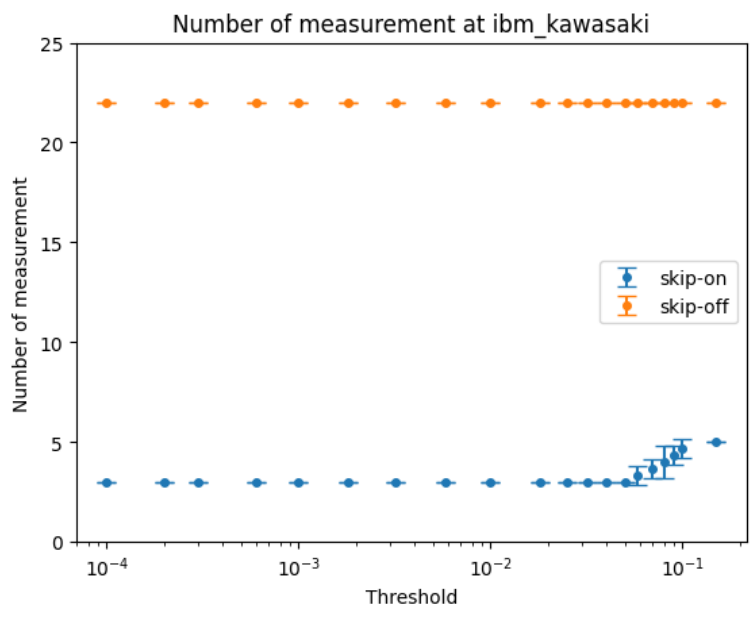

図 4.7　QPS1 ステップシミュレーション回路での閾値と測定回数の関係。横軸が閾値で縦軸が測定回数を表す。「skip on」は Skip-measurement ありの場合，「skip off」は Skip-measurement なしの場合を表す。

第 5 章

量子回路最適化プロトコル AQCELの性能評価

本章では第 3 章で紹介した量子回路最適化プロトコル AQCEL(Advancing Quantum Circuit by ICEPP and LBNL) [7] をより実用的な回路に適用することで，AQCELの応用可能性について検証する。これまでの AQCELの先行研究では第 3.2 節のように，QPS(Quantum Parton Shower) の 1 ステップに特化し設計された回路 SQPS(Specialized in 1 step Quantum Parton Shower) のような限られた条件の回路に対し実装されていた。今後の更なる応用を見込み，N ステップに一般化された回路 NQPS(N step Quantum Parton Shower) にも AQCELが効果的であることを検証する。本研究では，AQCELを，NQPS($N = 1$) に適用した場合と，1 ステップに特化し設計された SQPSに適用した場合とで最適化後の CNOT 数を比較した。これにより，実用的な形に近い NQPSにも AQCELが効果的であることを確認した。第 5.1 節では，今回の手法の評価方法について説明する。第 5.2 節ではその結果を示す。

5.1 評価方法

プログラムでは表 5.1 のバージョンを用いた。ベンチマーク回路には第 3.2.1 節で紹介した QPSの 1 ステップ回路を用いた。第 A 章に NQPSおよび SQPSの回路を示す。評価には第 2.4 節で説明した古典

	バージョン	引用文献
Python	3.8	[13]
Qiskit	0.21.0	[14]
Terra	0.15.2	
Aer	0.6.1	
Ignis	0.4.0	
pytket	0.6.1	

表 5.1　用いた Python やライブラリのバージョン。

	NQPS$(N=1)$	SQPS
用いた回路	NQPS1 ステップ回路	SQPS1 ステップ回路
Qiskit 標準の decomposer で decompose した場合の CNOT 数	936	155
Qiskit 標準の decomposer で decompose した場合の深さ	1230	256
Qiskit 標準の decomposer で decompose した場合の量子ビット数	20	11
最適化の方法	AQCEL	
AQCEL で用いたシミュレータ	statevector simulator	
AQCEL で用いた shot 数	1024	
回路を回すのに用いたシミュレータ	statevector simulator	
回路を回すのに用いた shot 数	2000	

表 5.2　NQPS と SQPS の条件。NQPS1 ステップ回路は一般の N ステップの QPS に対応できるように設計された回路において $N=1$ とした回路。SQPS1 ステップ回路は 1 ステップの QPS に特化して設計された回路。

忠実度を用いた。

表 5.2 のように条件を変え，CNOT 数を比較した。双方の回路は 1 ステップの QPS を計算するという点で同じ現象をシミュレートしており，また終状態の分布も同一になるはずである。NQPS では N ステップの QPS に対応できるように実装されている反面，1 ステップの回路としては必ずしも最適な形にはなっていないと考えられ，N ステップへの一般化のために必ずしも効率的と言えない回路が含まれていると考えられる。

5.2　評価

結果は表 5.3 の通りになった。また，AQCEL 前の NQPS$(N=1)$ シミュレーション回路と AQCEL 後の NQPS$(N=1)$ シミュレーション回路，AQCEL 前の SQPS シミュレーション回路と AQCEL 後の SQPS シミュレーション回路，AQCEL 前の NQPS$(N=1)$ シミュレーション回路と AQCEL 前の SQPS シミュレーション回路，AQCEL 後の NQPS$(N=1)$ シミュレーション回路と AQCEL 後の SQPS シミュレーション回路それぞれの間での statevector simulator による $p0_0, p0_1, p0_2, p1_0, p1_1, p1_2$ ビットでの古典忠実度はそれぞれ 1 となった。なお，本実験での statevector simulator では 2.2 で解説した統計エラーは含まれていない。

		NQPS$(N=1)$ (decomposed)	SQPS(decomposed)	$\frac{\text{NQPS (N=1)(decomposed)}}{\text{SQPS (decomposed)}}$
CNOT 数	AQCEL 前	936	155	6.04
	AQCEL 後	62 (93% 減)	41 (74% 減)	1.51 (1.3)

表 5.3　Qiskit 標準の decomposer で decompose した後の NQPS, SQPS の AQCEL 前後での CNOT 数。第 1, 2 列における括弧内は削減率。最右列における括弧は百分率同士の比。

第 6 章

結論と考察および今後の展望

本研究では初期条件に依存した量子回路最適化プロトコル AQCEL(Advancing Quantum Circuit by ICEPP and LBNL) の改善手法 Skip-measurement を実装することにより，特定の閾値における計算精度の改善を実現した。また，最適化にかかる計算時間を減らすことに成功した。さらに，先行研究に比べより実用的な回路に AQCEL を適用することで AQCEL の応用可能性について確認した。本章では本研究における要旨をまとめるとともに第 4 章，第 5 章における結果を考察し，また今後の展望について述べる。

6.1 AQCEL の改善手法 Skip-measurement

第 3.1 節で説明したように，AQCEL とは初期状態に依存した量子回路最適化プロトコルであり，改善手法 Skip-measurement を実装した。Skip-measurement では以前の測定結果を古典コンピュータ上に保存し繰り返し使用することで，計算精度の改善と測定回数の減少を目指した。ベンチマーク回路である量子パートンシャワー (QPS: Quantum Parton Shower) 1 ステップシミュレーション回路，IBM の超伝導型量子コンピュータ ibm_ kawasaki，shot 数 4096 回で各閾値で 3 度ずつ最適化および理想的な状態と比較した時の古典忠実度の計算を行い，Skip-measurement ありの場合となしの場合で閾値に対する古典忠実度を比較した。結果，1.0×10^{-4} から 1.0×10^{-3} の閾値で Skip-measurement ありの場合が Skip-measurement なしの場合に比べて高い古典忠実度となることが示された。また，測定回数についても同じ閾値範囲の 1.0×10^{-4} から 1.0×10^{-3} で Skip-measurement ありの場合測定回数 3 回となり，Skip-measurement なしの 22 回と比較して測定回数が減少することが示された。

1.0×10^{-4} から 1.0×10^{-3} の範囲で Skip-measurement ありの場合が Skip-measurement なしの場合に比べて古典忠実度が高かった理由は初期化エラー，デコヒーレンス，ゲートエラー，測定エラーなどの種々のエラーを回避できるためと考えられる。6.1 で種々のエラーを示した。各エラーの詳細については第 2.3 節で説明した。第 1 量子ビットの最初の CNOT ゲート直前で測定をした際に初期化エラーにより $|0\rangle$ が測定されない場合は，理想的な結果と異なる結果が出るため古典忠実度が下がる要因になると考えられる。仮に正しく初期化が行われていたとしてもデコヒーレンスによって第 1 量子ビットの最初の CNOT ゲート直前での測定において $|0\rangle$ が正しく測定されない可能性もあり，これも古典忠実度を下げる要因と考えられる。図 6.1 中の X ゲートがゲートエラーを引き起こした場合，第 1 量子ビットの 2 番

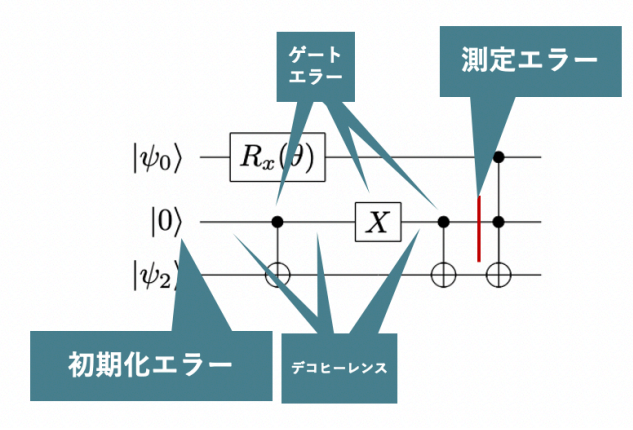

図 6.1　Skip-measurement で回避したと考えられるエラーの例。赤線で測定位置を示した。また，吹き出しで考えられるエラーとその該当箇所を示した。

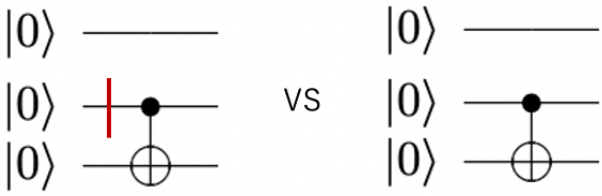

図 6.2　測定エラーの例。赤線で測定位置を示した。

目の CNOT ゲート直前で測定をした際に正しい状態 $|1\rangle$ が測定されない可能性があり，古典忠実度を下げる。さらに，図 6.2 に示したように，以前の測定情報 $|0\rangle$ がわかっている場合は測定をする (図中左) よりも測定をせず以前の情報を使う (図中右) 方が測定エラーを回避できる可能性がある。ただし，以前の測定結果に誤りがある場合は再度測定した方が正しい結果に訂正される可能性もあり，一概に古典忠実度を下げるのみではないことに注意が必要である。

測定回数は，以前の情報を古典コンピュータに保存し繰り返し使用できるパターン (表 4.1，4.2) が多くあったため減少したと考えられる。

閾値にピークが生じる要因は閾値が高すぎる場合，偽の状態も正しい状態と判断してしまうためであり，閾値が低すぎるとノイズの影響を受け正しい状態を判定できなくなるトレードオフ関係が存在するためと考えられる。図 6.3 に閾値のピーク要因の概略を図で示した。

エラーバーが大きくなる理由は，最適化によって生成される回路が確率的に二極化してしまうためと考えられる。図 6.4 に回路同士の RMS によるエラーバーをつける前の閾値と古典忠実度の関係を示した。エラーバーは各点ごとに 3 回ずつ測定をした場合の RMS を用いた。例えば，閾値 1.0×10^{-4} の Skip-measurement ありの点では，古典忠実度が 0.96(1) に 2 点集まっているのに対し，0.61(1) 付近に 1 点あることがわかる。また，閾値 1.8×10^{-3} の Skip-measurement ありの点でも，古典忠実度が 0.93(1) に 2 点集まっているのに対し，0.60(1) に 1 点あることがわかる。同様に，閾値 1.8×10^{-4} の Skip-measurement ありの点，閾値 $3.2 \times 10^{-4}, 1.0 \times 10^{-3}, 5.8 \times 10^{-3}, 1.0 \times 10^{-1}$ の Skip-measurement なしの点でも同様に古典忠実度の値が二極化していることがわかる。このように，エラーバーが大きい点

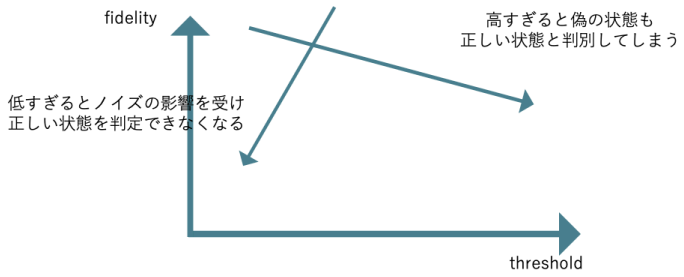

図 6.3　閾値のピーク要因。横軸が閾値。縦軸が忠実度。

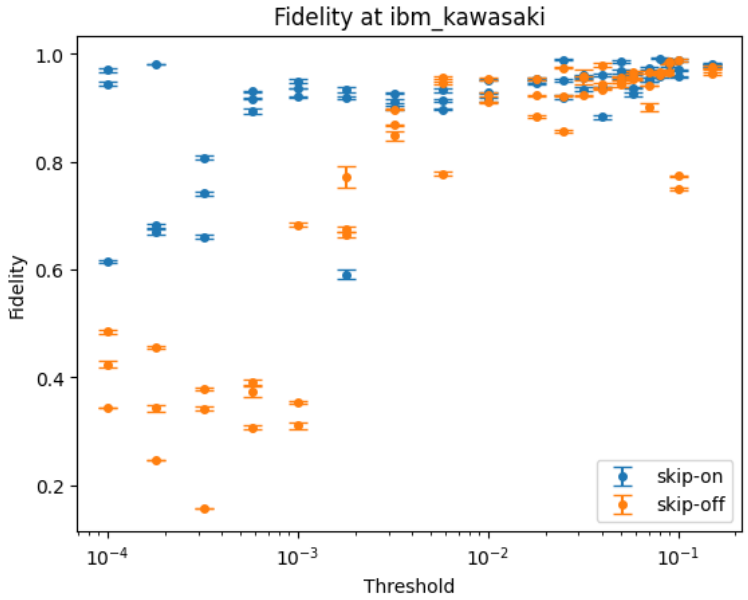

図 6.4　RMS によるエラーバーをつける前の閾値と忠実度の関係。横軸が閾値。縦軸が忠実度。

では古典忠実度が二極化しており，これはゲートエラーや測定エラーなどの要因により確率的に回路が大きく変わってしまうためと考えられる。

本研究によって，量子パートンシャワー (QPS: Quantum Parton Shower) 1 ステップシミュレーション回路，IBM の超伝導型量子コンピュータ ibm_ kawasaki，shot 数 4096 回，1.0×10^{-4} から 1.0×10^{-3} の閾値で Skip-measurement あり AQCEL を用いた場合，Skip-measurement なしの場合に比べて計算精度が向上することが示された。これにより，将来的な誤り耐性量子コンピュータの実現を早めることにつながる。また，測定回数を減らすことで最適化にかかる計算時間についても従来の AQCEL に比べて減少していると考えらえる。したがって，古典コンピュータに対して量子コンピュータを使うことで計算量的に優位である問題サイズを，従来の AQCEL に比べて小さくすることにつながると考えられる。

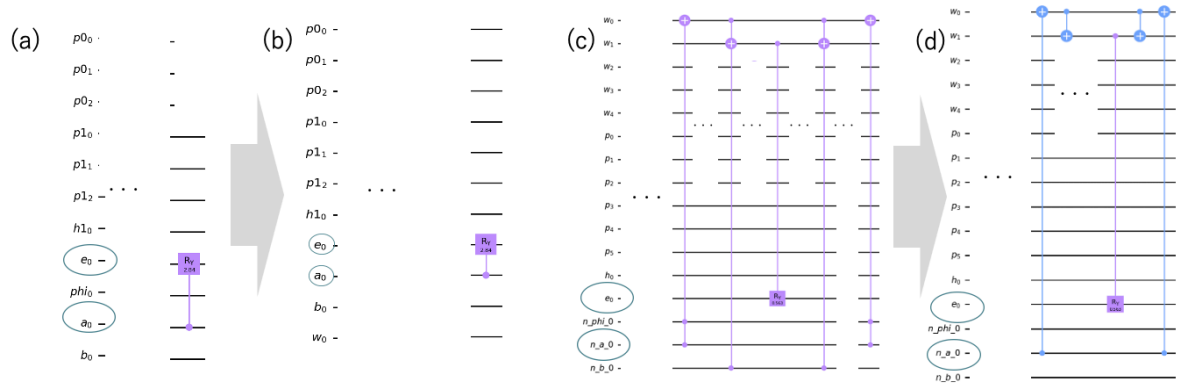

図 6.5　AQCEL後の NQPS(decomposed) 回路での CNOT 数が SQPS(decomosed) よりも多くなった要因の例。 (a) は AQCEL による最適化前の SQPS回路の一部。(b) は AQCEL による最適化後の SQPS回路の一部。(c) は AQCEL による最適化前の NQPS回路の一部。(d) は AQCEL による最適化後の NQPS回路の一部。

6.2　量子回路最適化プロトコル AQCELの性能評価

第 5 章では AQCEL の応用可能性について検証した。先行研究 [7] における QPS1 ステップシミュレーションに特化して設計された回路 SQPS(Specialized in 1 step Quantum Parton Shower) だけでなく，より実用的な形に近い回路 NQPS(N step Quantum Parton Shower) にも AQCEL が応用可能であるかどうかを確かめた。具体的には NQPS で $N = 1$ としたシミュレーション回路と SQPS シミュレーション回路それぞれに AQCEL を適用し，最適化後の回路をそれぞれ Qiskit 標準の decomposer を用いて分解した回路における CNOT 数を比較した。結果，NQPS(decomposed) の CNOT 数は SQPS(decomposed) の約 1.51 倍と同程度のオーダーとなり，より実用に近い回路 NQPS($N = 1$) でも効果的であると考えられる。

図 6.5 に，表 5.3 で AQCEL後の NQPS(decomposed) 回路での CNOT 数が SQPS(decomosed) よりも多くなった要因について例を示した。 (b) の回路では a_0 ビットの情報を直接 C_{R_y} ゲートにより e_0 ビットに伝達しているのに対し， (d) の回路では一度作業ビットに情報を伝達した後に e_0 ビットに情報を伝達し再度作業ビットを元の状態に戻している。同様の構造が複数現れるため AQCEL後の NQPS(decomposed) 回路での CNOT 数が SQPS(decomosed) よりも多くなると考えられる。実際の回路を見てみると，NQPS の回路である図 A.1，A.2，A.3，A.4 では多制御ゲートを SQPS ではそのままの形で残しているのに対し SQPS では図 A.6 のようにトフォリのまま回路を実装していることがわかる。，NQPS ではトフォリまで分解した形で設計されている。AQCEL では多制御ゲートの効率的な分解までを含めて最適化しているので，SQPS では AQCEL によってより少ない CNOT 数まで最適化できるのに対し，NQPS では CNOT を減らしきれないことが予想される。実際，decomose する前の NQPS では多制御ゲート 0 個，トフォリ 149 個，一制御ゲート 25 個 (C_{R_y}, C_x, C_H の合計) なのに対し，SQPS で多制御ゲート 6 個，トフォリ 0 個，CNOT6 個 (C_{R_y}, C_x の合計) である。

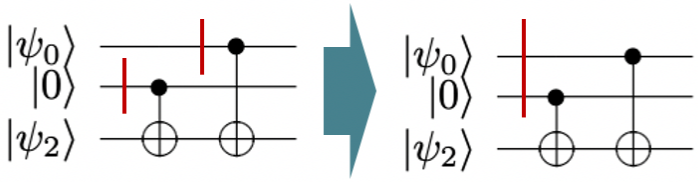

図 6.6　測定位置の工夫により回路の再構成の数を減らせる例。赤線で測定位置を表している。

6.3　今後の展望

今後の AQCEL では，さらに計算精度を上げていくことが展望として考えられる。例えば，今回の Skip-measurement では表 4.1，4.2 のように X ゲートが間にかかっている場合についての測定の省略を行った。一方，$R(\pi)$ ゲートなどがかかっている場合には対応しておらず，このような X ゲートと同様の操作を行うゲートに Skip-measurement を対応させることは一般に計算精度を上げることにつながる。

また，さらに最適化にかかる計算時間を短縮させることも課題だ。図 6.6 の例では測定を 2 ヶ所別々に行っている。従来の AQCEL では左図のように第 1 量子ビットの CNOT ゲート直前で測定した後に初期化・回路の再構成と実行をし第 0 量子ビットの CNOT 直前で再び測定をし，再度初期化・回路の再構成と実行を行う。一方，右図では第 1 量子ビットの CNOT ゲート直前で測定を行うとともに，次に第 0 量子ビットの測定が必要になることを見越し第 0 量子ビットも測定を行う。これにより初期化・回路の再構成と実行を 1 回にまとめて実行できることから，最適化にかかる計算時間の短縮につながりうる。

AQCEL の適用範囲を広げていくことも課題だ。現状，限られた初期条件でなければ制御ゲートの削除を行うことができず，一般に回路のエンタングルメントが大きくなるほど制御ゲートを削除できる割合も低下していくと考えられる。

謝辞

ここに，本研究にあたりご支援，お力添えくださった方々に深く感謝を申し上げます。指導教員である澤田龍准教授には，研究に関することだけではなく，生活のことから事細かにご相談に乗っていただきました。毎朝のミーティングから技術的なことまで，困った時はなんでも惜しまず支援してくださいました。本当にありがとうございました。寺師弘二准教授には研究テーマ説明から始まり，毎週のミーティングでのアドバイスなど幾度となくご指導いただきました。さらに UC バークレーとの共同研究の渡航など，研究そのもの以外の部分でも大変お世話になりました。寺師先生の知恵とアドバイスに感謝しています。田中純一教授には普段のミーティングで何度もアドバイスをいただきました。飯山悠太郎助教には環境構築から始まり技術的に深いところまで時間を惜しまず助言いただきました。永野廉人特任助教には研究の相談だけでなく渡航時の身の回りの相談に始まり，たくさんのアドバイスをいただきました。齊藤真彦助教には技術的に自力では実装が難しかった部分について丁寧に解説していただきました。Sanmay Ganguly 特任助教にはミーティングでのご助言をいただきました。稲田聡明助教には学部時代の授業に始まりわからないことを教えていただきました。Wai Yuen Chan 特任研究員にはミーティングで鋭い質問をしていただきました。加地俊瑛特任研究員には自力では難しかった技術的な部分について協力して研究してくださいました。また，毎朝のミーティングでも鋭い視点でディスカッションしてくださると共に新しい情報を吸収し私に教えてくださいました。難波俊雄助教は居室においてわからないことを時間を作って教えてくださいました。改めて ICEPP の量子コンピューティンググループ，および ICEPP の皆様に感謝申し上げます。

本研究においては Lawrence Berkeley National Laboratory の方々にもお世話になりました, Christian W. Bauer は大学院生になったばかりで研究の右左もわからない自分とのディスカッションに付き合ってくださり，大学院において研究を前に進める大きなきっかけをくださいました。Benjamin Nachman さんにはコードの共有から帰国後のフォローまでお忙しい中大変懇意にご協力いただきました。お二方，および LBNL の方々の助けなしでは，何も成し遂げられなかったでしょう。改めて，関わってくださった全ての LBNL の関係者の方々に深い感謝を申し上げます。

張元豪さんには研究とは何かイチから叩きこんでいただき，最後まで研究に伴走しサポートしていただくだけでなく，定期的に大丈夫か確認の連絡をとってくださいました。張さんの支えがあったからこそ，私はここまで来られました。一緒に量子コンピュータの研究をした同期の楽しい時間を過ごさせてくれた同期，先輩，後輩の皆さん，ありがとうございました。最後に，素晴らしい研究環境を用意してくださった浅井祥仁教授，ICEPP 事務室の皆様，家族に感謝申し上げます。

参考文献

[1] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.*, 41(2):303–332, 1999.

[2] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition, 2011. ISBN 1107002176.

[3] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855. doi: 10.1145/237814.237866. URL https://doi.org/10.1145/237814.237866.

[4] D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, page 176–188, New York, NY, USA, 1997. Association for Computing Machinery. ISBN 0897918886. doi: 10.1145/258533.258579. URL https://doi.org/10.1145/258533.258579.

[5] Kosuke Fukui, Akihisa Tomita, Atsushi Okamoto, and Keisuke Fujii. High-threshold fault-tolerant quantum computation with analog quantum error correction. *Phys. Rev. X*, 8:021054, May 2018. doi: 10.1103/PhysRevX.8.021054. URL https://link.aps.org/doi/10.1103/PhysRevX.8.021054.

[6] Ibm quantum, 2021. URL https://quantum-computing.ibm.com.

[7] Wonho Jang, Koji Terashi, Masahiko Saito, Christian W. Bauer, Benjamin Nachman, Yutaro Iiyama, Tomoe Kishimoto, Ryunosuke Okubo, Ryu Sawada, and Junichi Tanaka. Quantum gate pattern recognition and circuit optimization for scientific applications. *EPJ Web of Conferences*, 251:03023, 2021. ISSN 2100-014X. doi: 10.1051/epjconf/202125103023. URL http://dx.doi.org/10.1051/epjconf/202125103023.

[8] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, Nov 1995. ISSN 1094-1622. doi: 10.1103/physreva.52.3457. URL http://dx.doi.org/10.1103/PhysRevA.52.3457.

[9] Yang Liu, Gui Lu Long, and Yang Sun. Analytic constructions of general n-qubit controlled gates, 2007.

[10] Dmitri Maslov. Advantages of using relative-phase toffoli gates with an application to multiple control toffoli optimization. *Physical Review A*, 93(2), Feb 2016. ISSN 2469-9934. doi: 10.1103/physreva.93.022311. URL http://dx.doi.org/10.1103/PhysRevA.93.022311.

[11] Benjamin Nachman, Davide Provasoli, Wibe A. de Jong, and Christian W. Bauer. Quantum algorithm for high energy physics simulations. *Physical Review Letters*, 126(6), Feb 2021. doi: 10.1103/physrevlett.126.062001.

[12] Wonho Jang, Koji Terashi, Masahiko Saito, Christian W. Bauer, Benjamin Nachman, Yutaro Iiyama, Ryunosuke Okubo, and Ryu Sawada. Initial-state dependent optimization of controlled gate operations with quantum computer. *Quantum*, 6:798, September 2022. ISSN 2521-327X. doi: 10.22331/q-2022-09-08-798. URL http://dx.doi.org/10.22331/q-2022-09-08-798.

[13] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[14] Gadi Aleksandrowicz et al. Qiskit: An Open-source Framework for Quantum Computing, 2019. URL https://doi.org/10.5281/zenodo.2562111.

第 A 章

本研究で出力された量子回路

A.1　第 5.2 節で出力された量子回路

NQPSでは回路は図 A.1，A.2，A.3，A.4 の回路から図 A.5 の回路に変わった。SQPSでは回路は図 A.6 の回路から図 A.7 の回路に変わった。また，図 A.8 から図 A.32 にそれぞれの回路を Qiskit 標準の decomposer を用いて decompose した回路を示した。

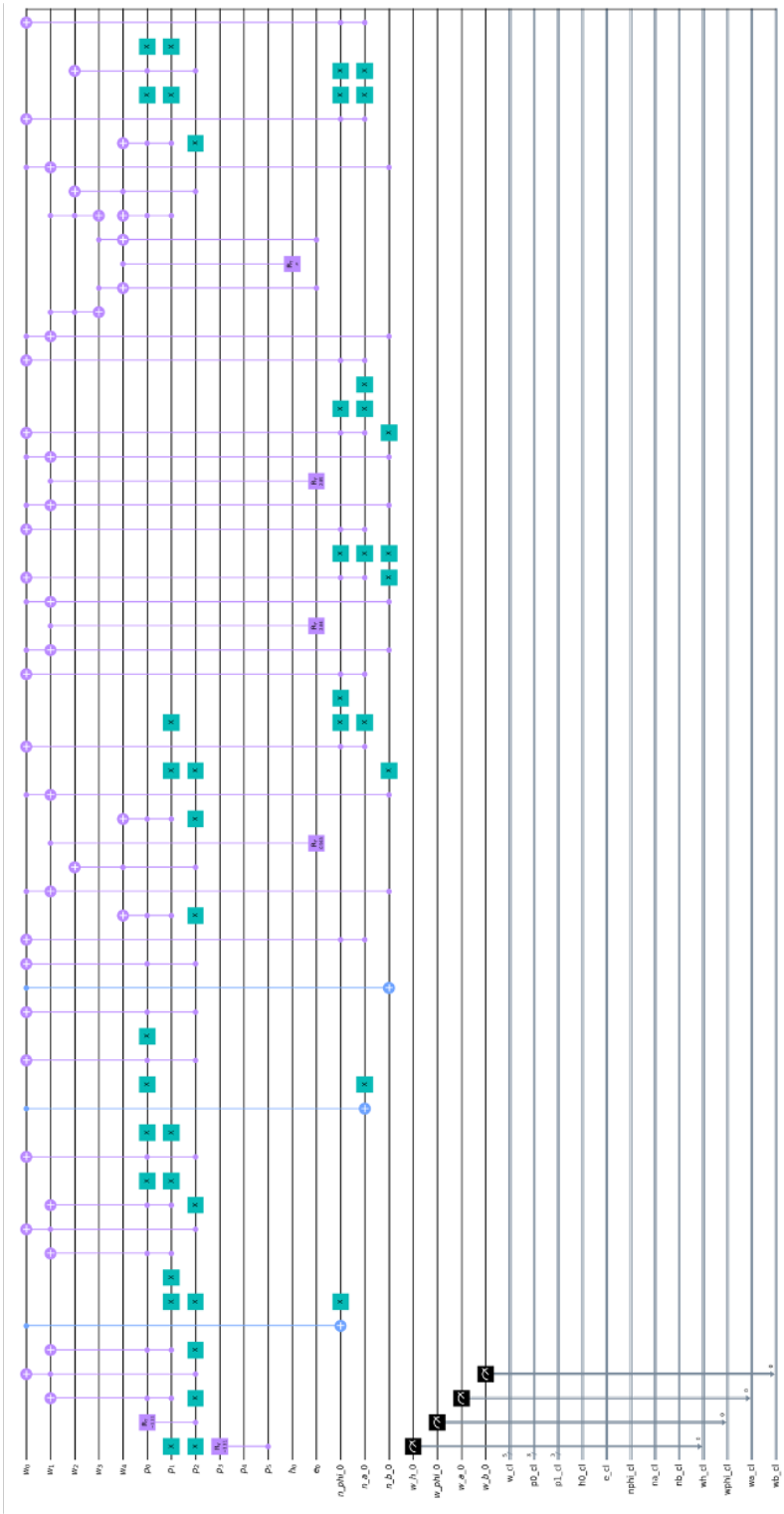

図 A.1　NQPS1 ステップシミュレーション回路の AQCEL 適用前の回路 (1/4)。

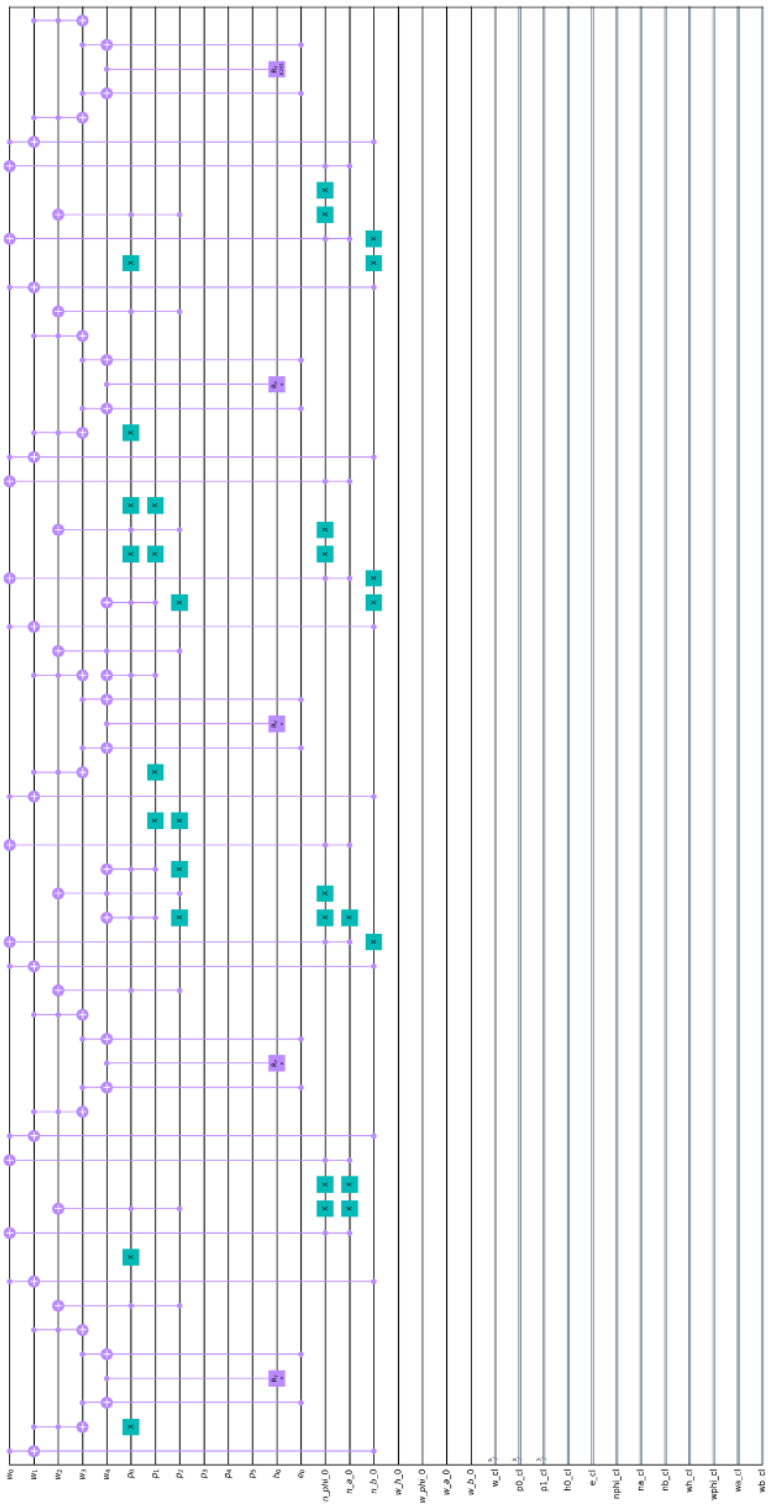

図 A.2　NQPS1 ステップシミュレーション回路の AQCEL 適用前の回路 (2/4)。

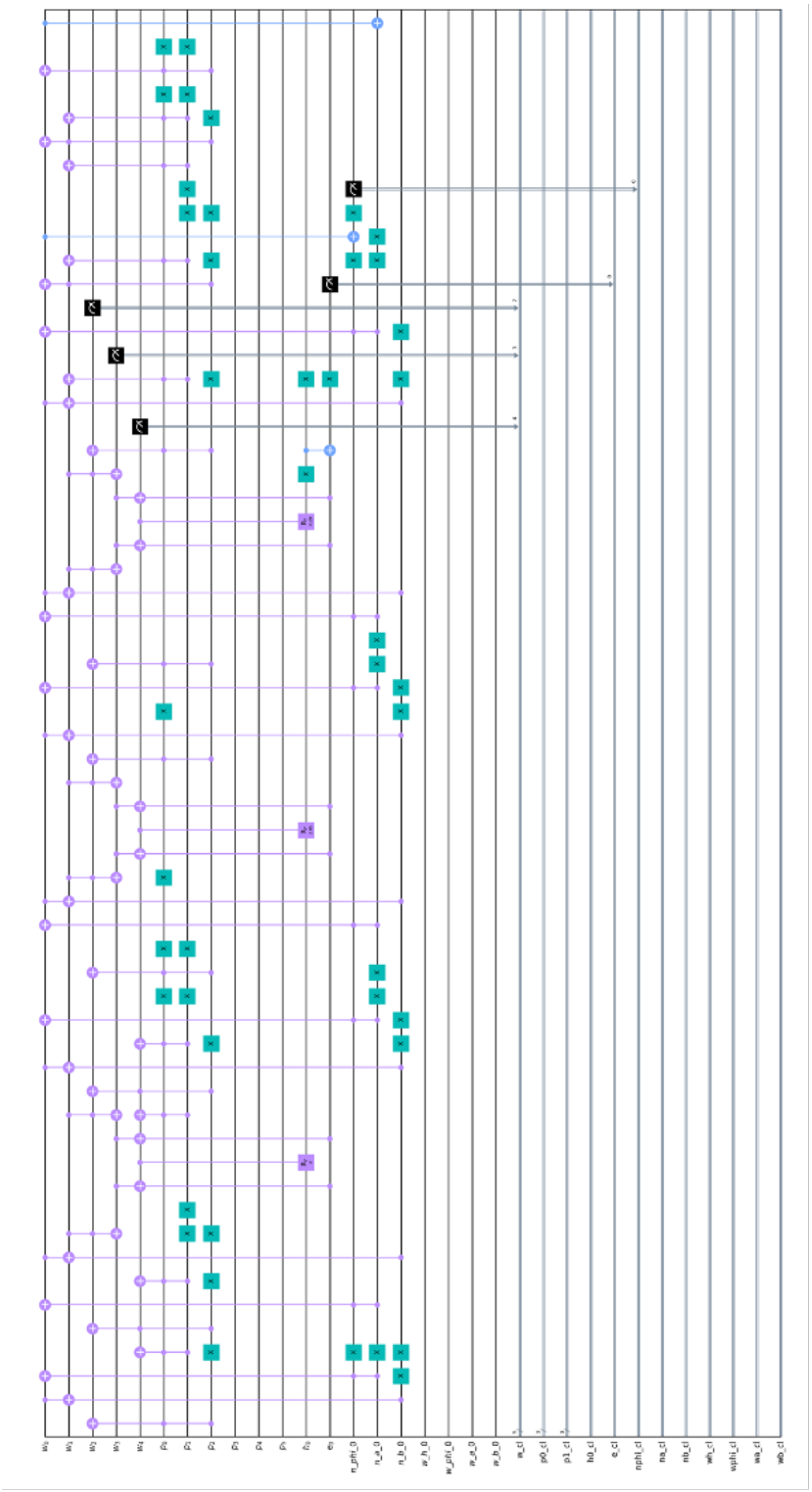

図 A.3　NQPS1 ステップシミュレーション回路の AQCEL 適用前の回路 (3/4)。

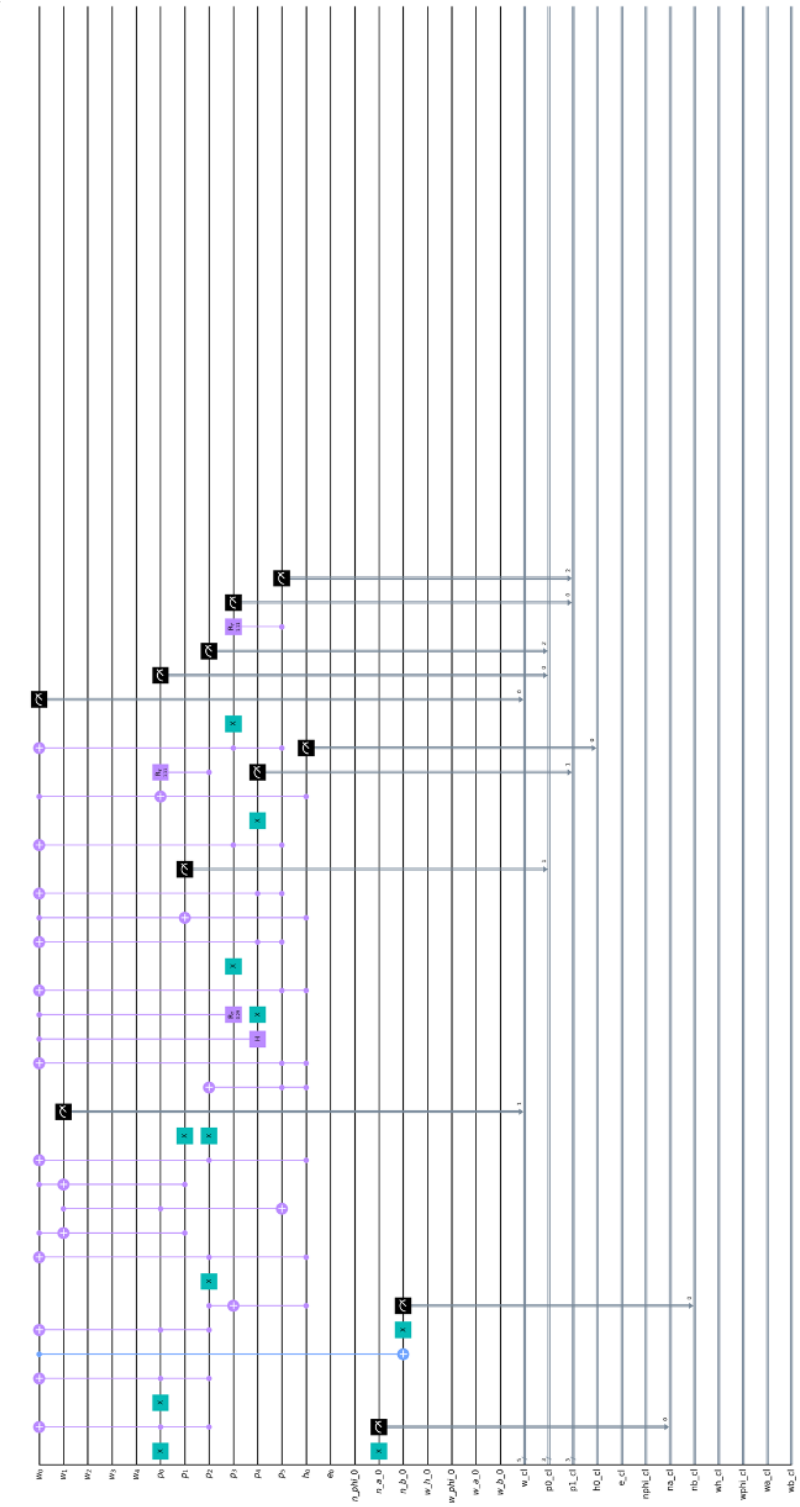

図 A.4　NQPS1 ステップシミュレーション回路の AQCEL 適用前の回路 (4/4)。

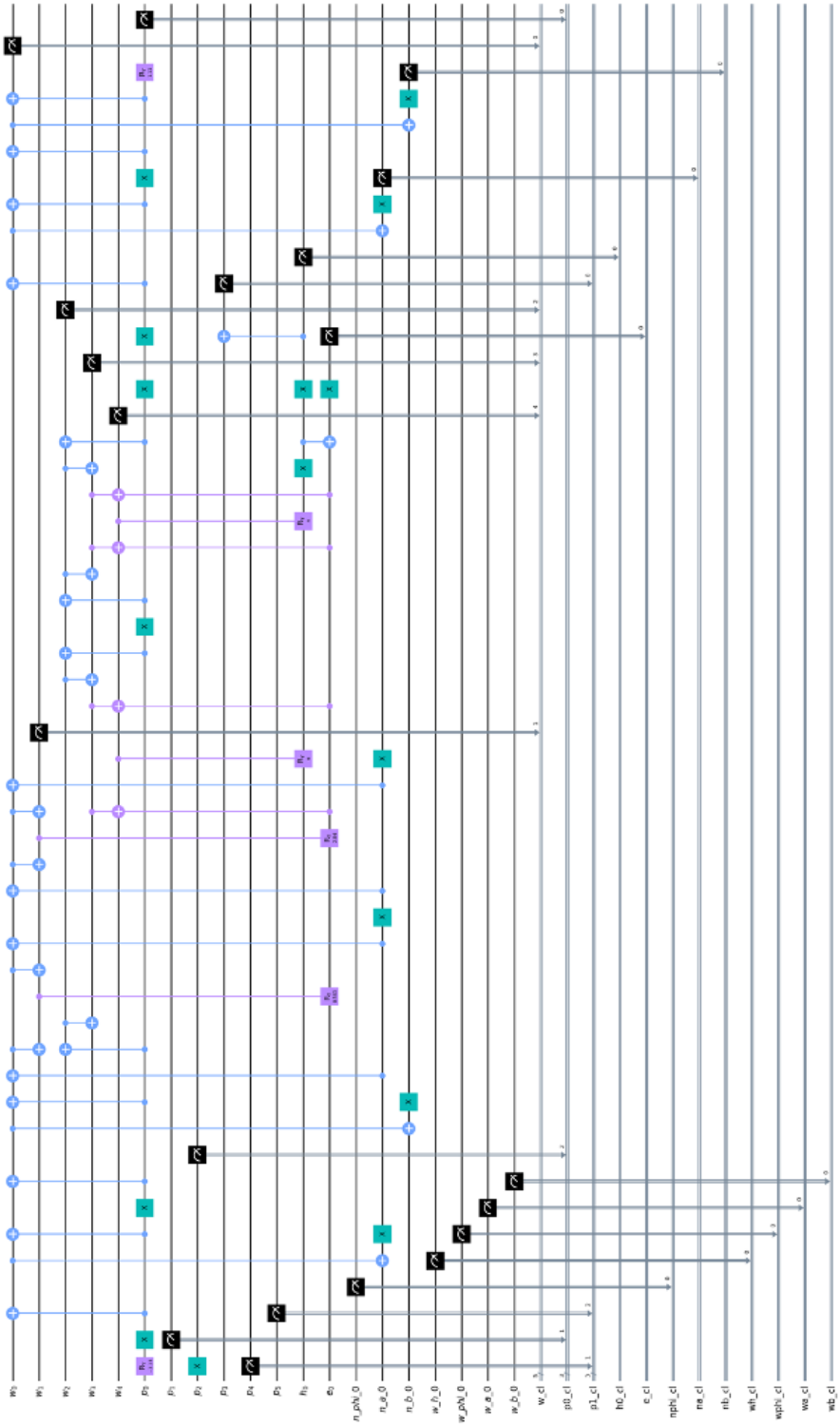

図 A.5　NQPS1 ステップシミュレーション回路の AQCEL 適用後の回路。

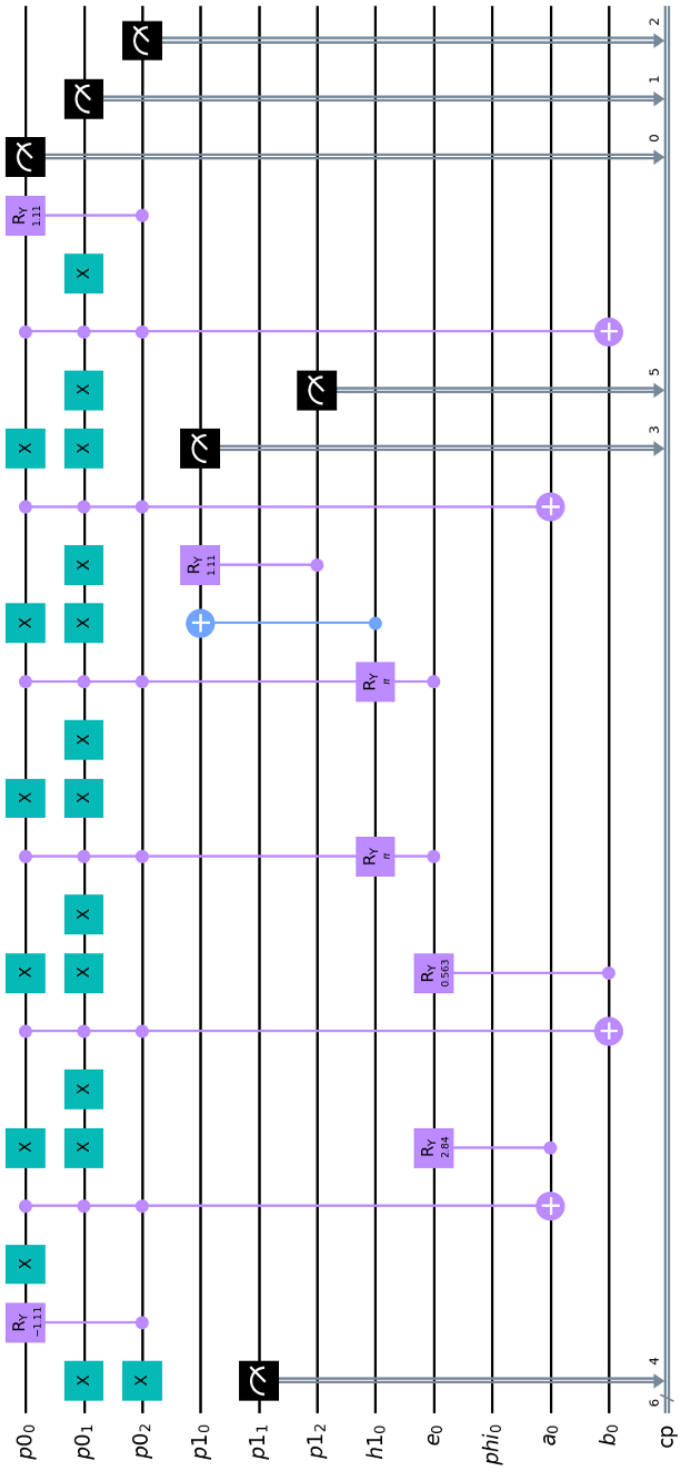

図 A.6　SQPS1 ステップシミュレーション回路の AQCEL 適用前の回路。

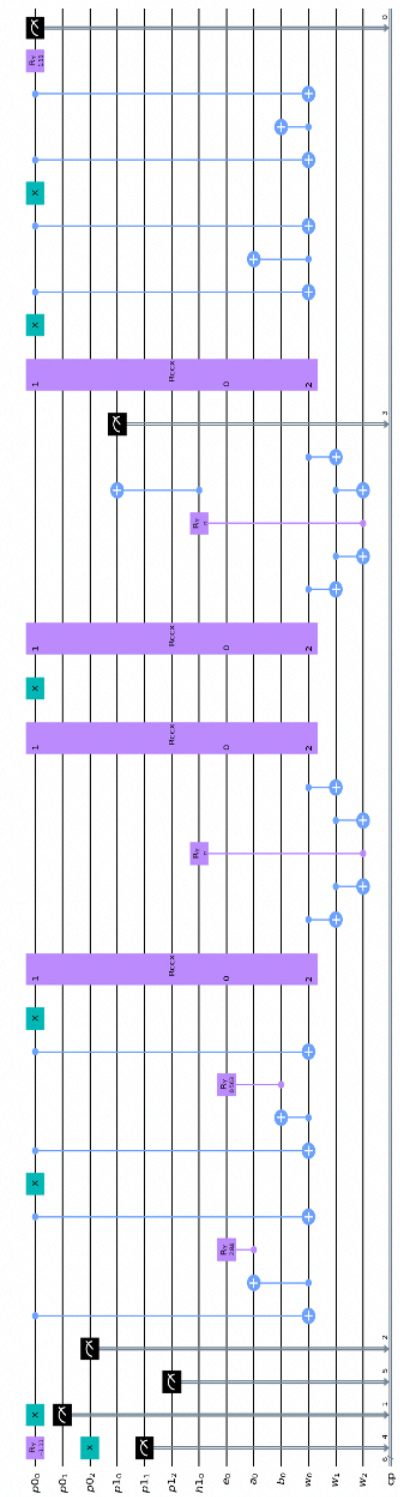

図 A.7　SQPS1 ステップシミュレーション回路の AQCEL 適用後の回路。

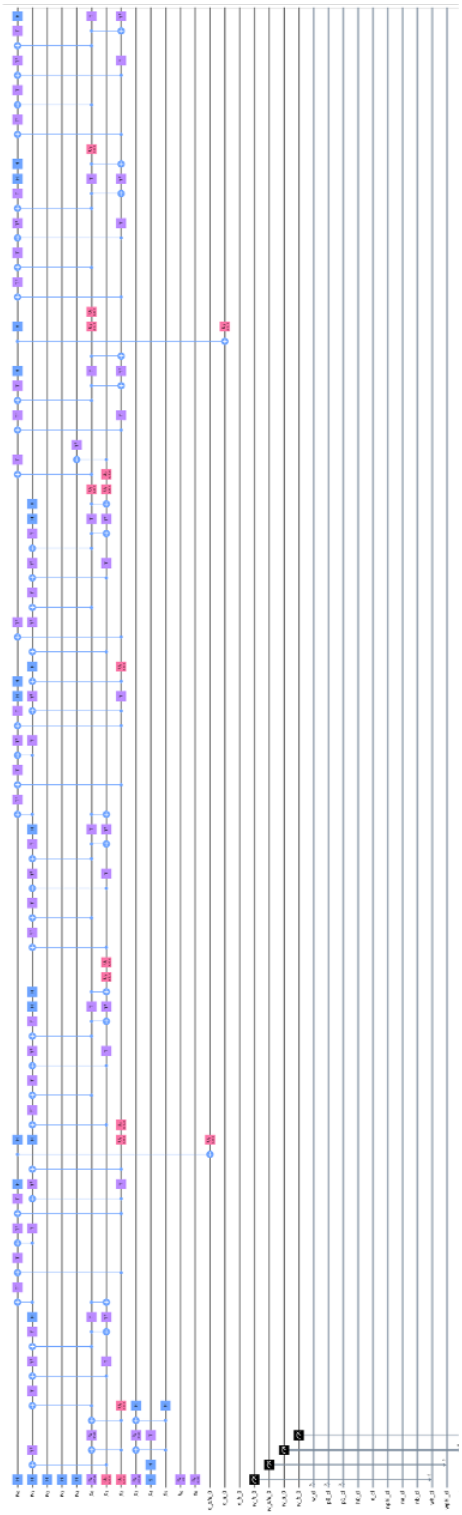

図 A.8　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (1/15)。decompose は Qiskit 標準の decomposer で行った。

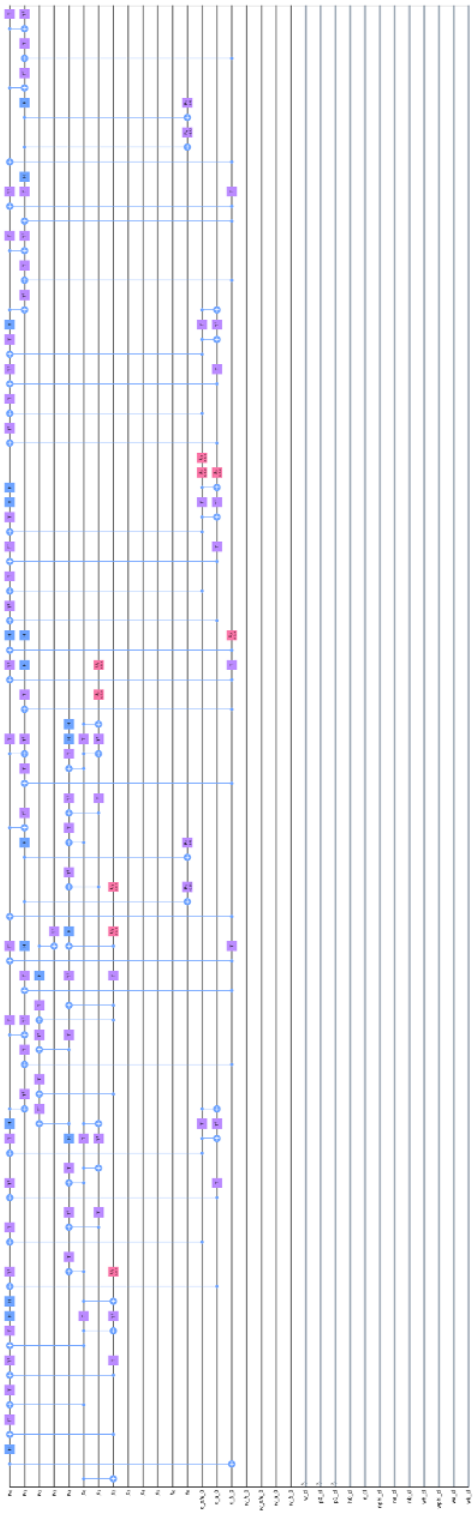

図 A.9　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (2/15)。decompose は Qiskit 標準の decomposer で行った。

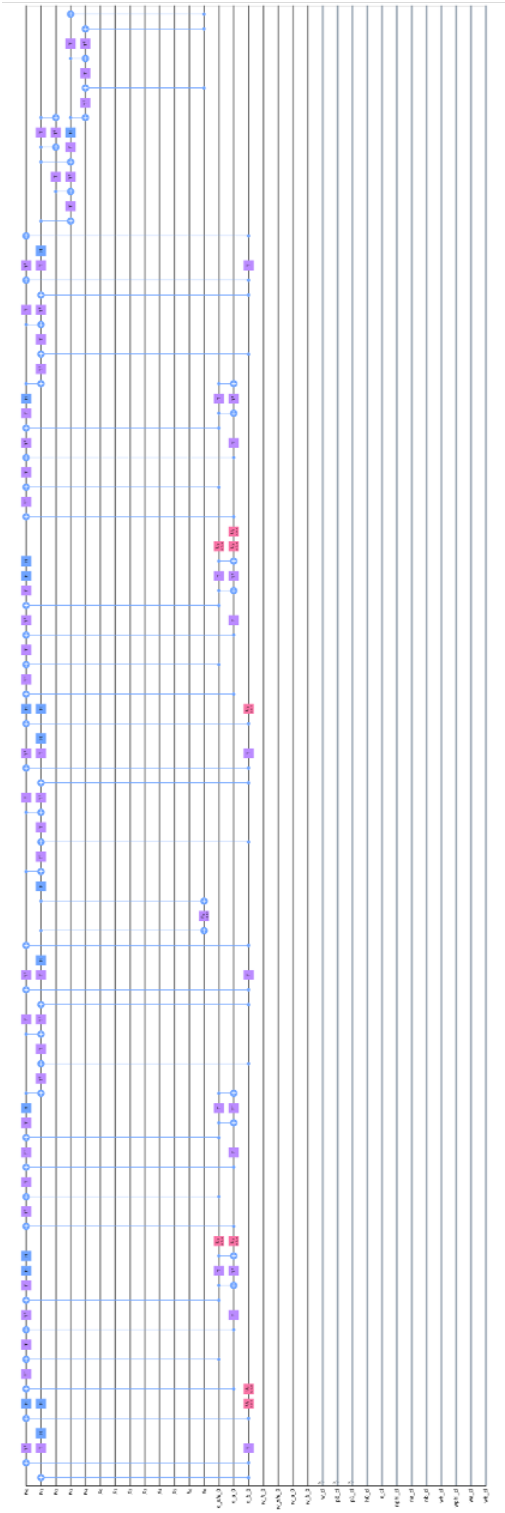

図 A.10　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (3/15)。decompose は Qiskit 標準の decomposer で行った。

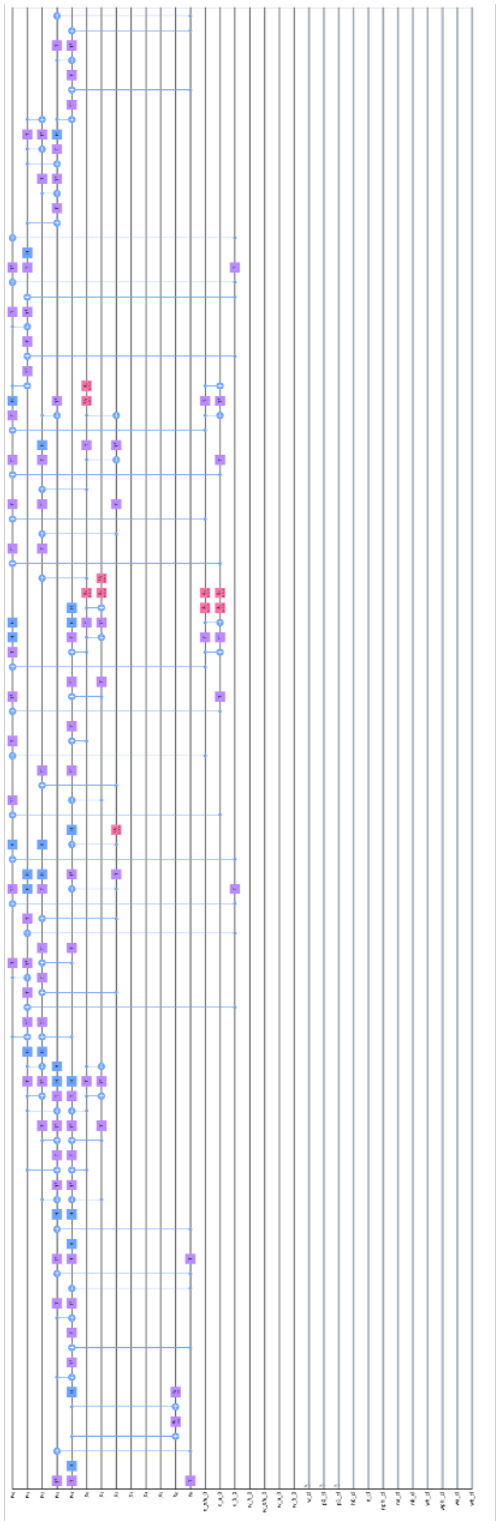

図 A.11　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (4/15)。decompose は Qiskit 標準の decomposer で行った。

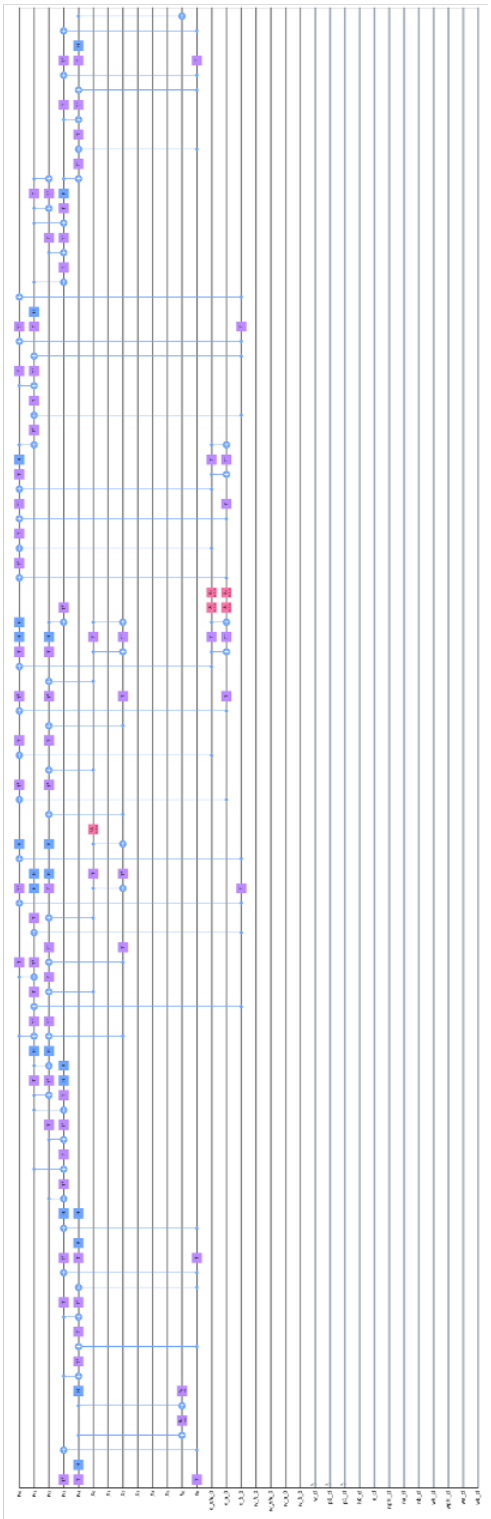

図 A.12　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (5/15)。decompose は Qiskit 標準の decomposer で行った。

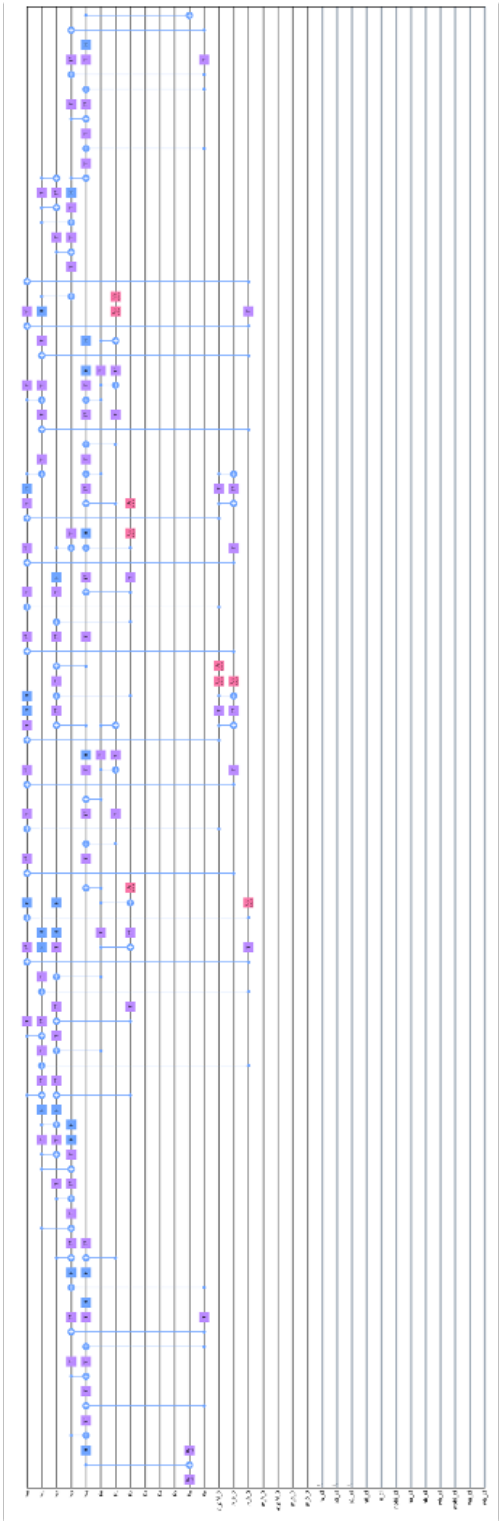

図 A.13　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (6/15)。decompose は Qiskit 標準の decomposer で行った。

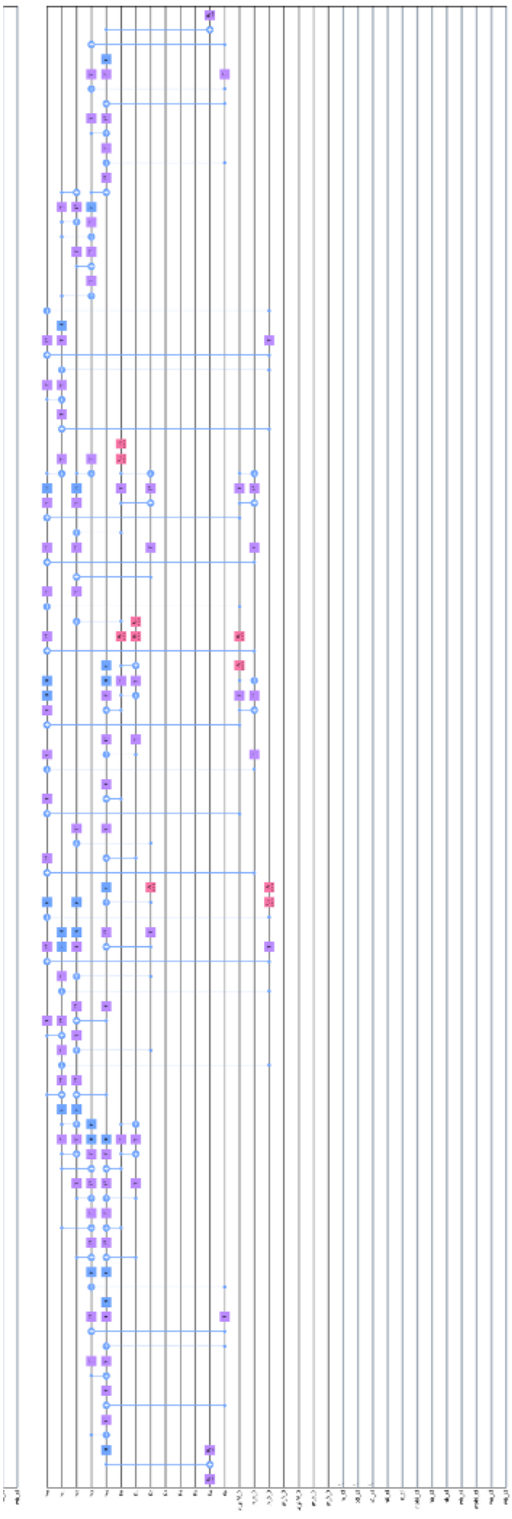

図 A.14　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (7/15)。decompose は Qiskit 標準の decomposer で行った。

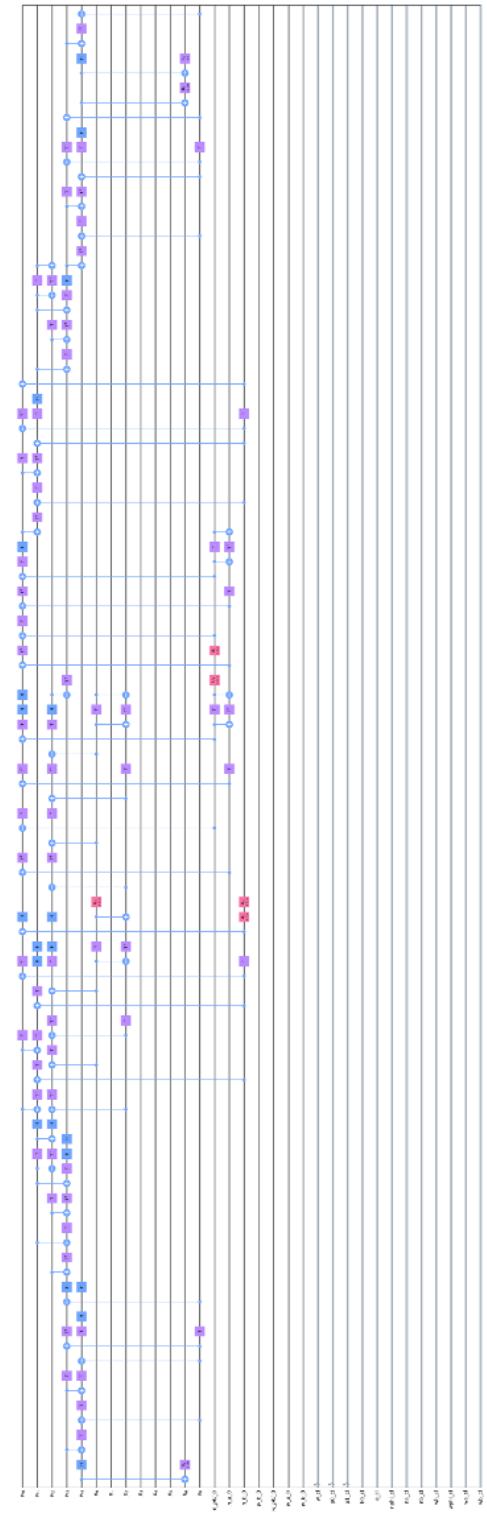

図 A.15　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (8/15)。decompose は Qiskit 標準の decomposer で行った。

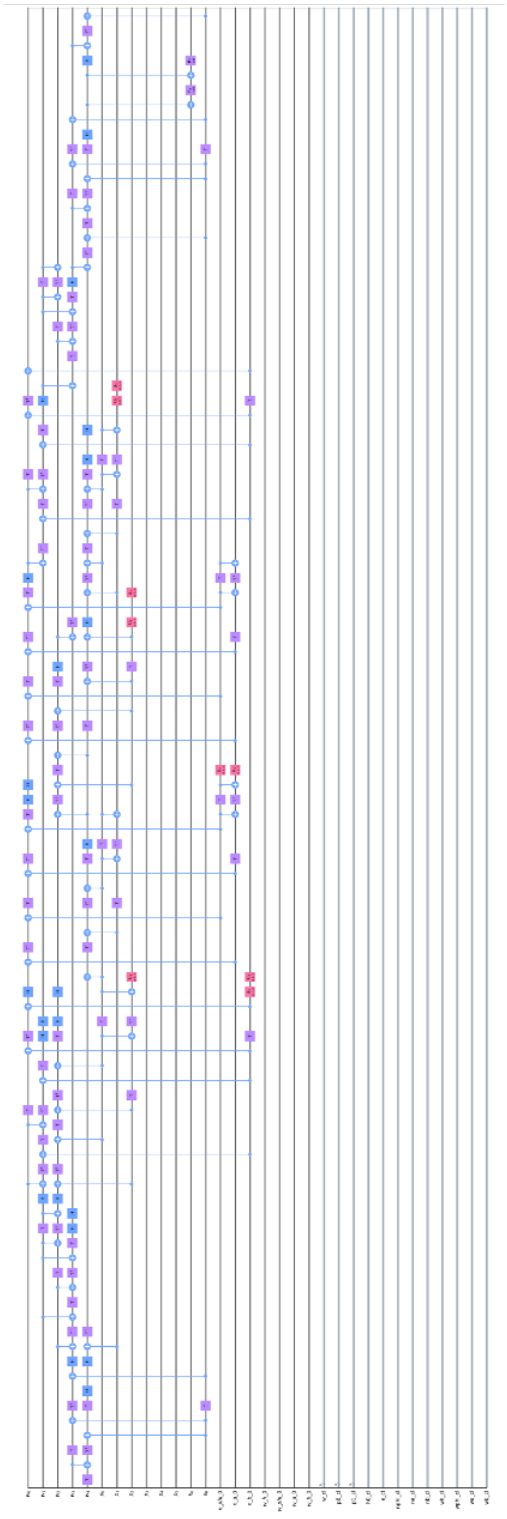

図 A.16　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (9/15)。decompose は Qiskit 標準の decomposer で行った。

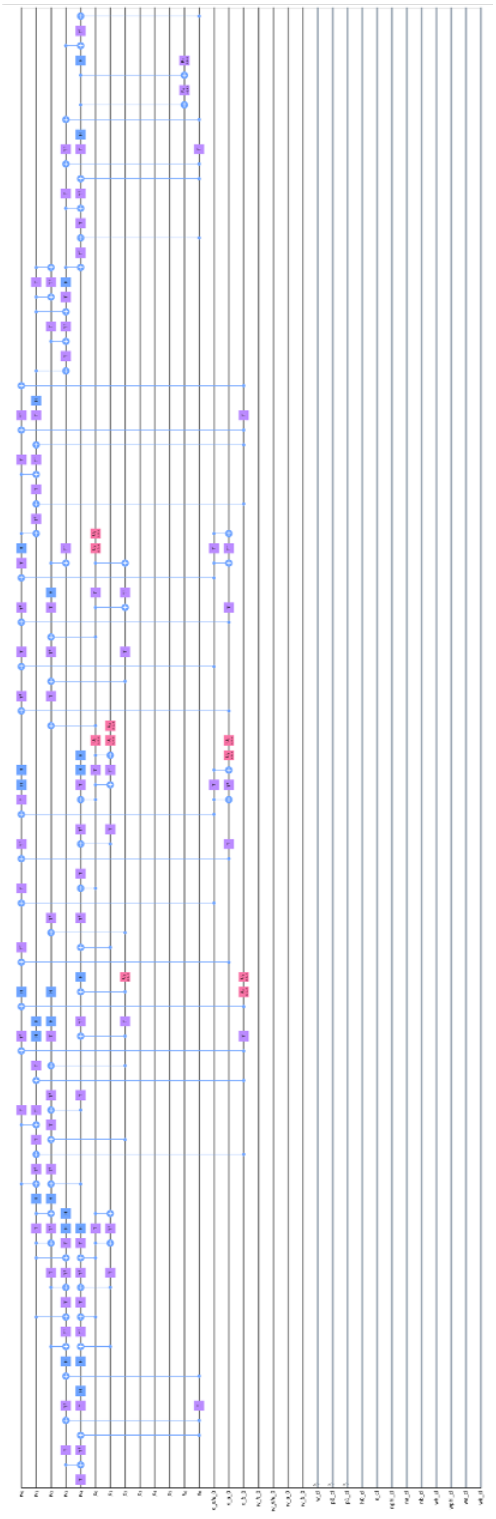

図 A.17　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (10/15)。decompose は Qiskit 標準の decomposer で行った。

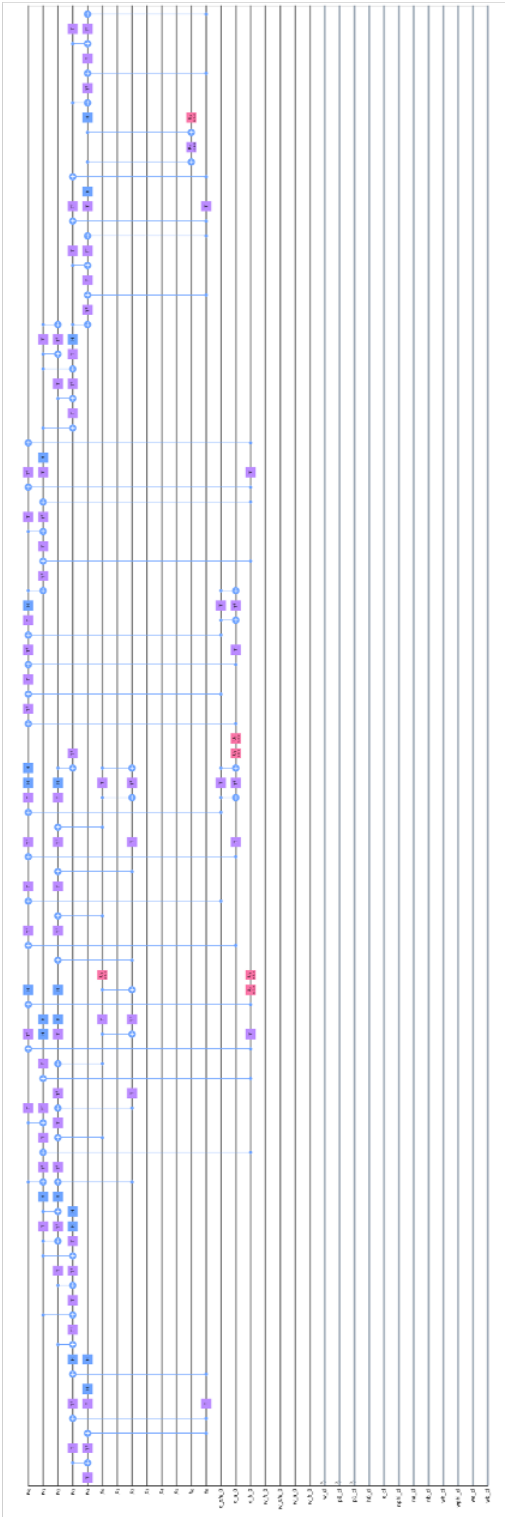

図 A.18　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (11/15)。decompose は Qiskit 標準の decomposer で行った。

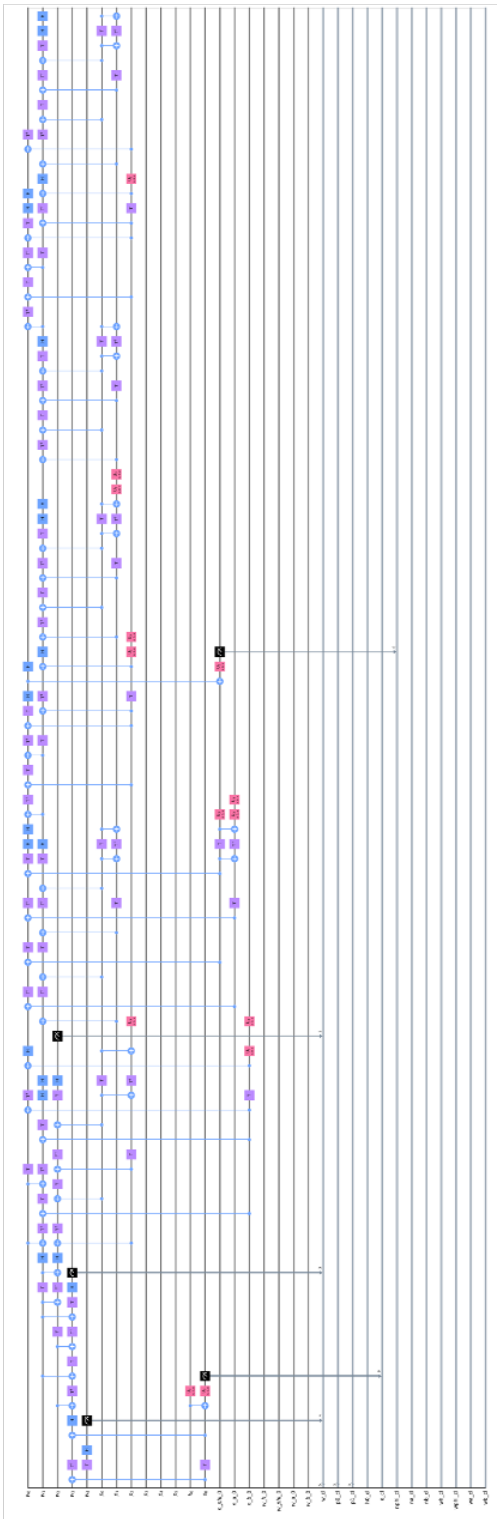

図 A.19　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (12/15)。decompose は Qiskit 標準の decomposer で行った。

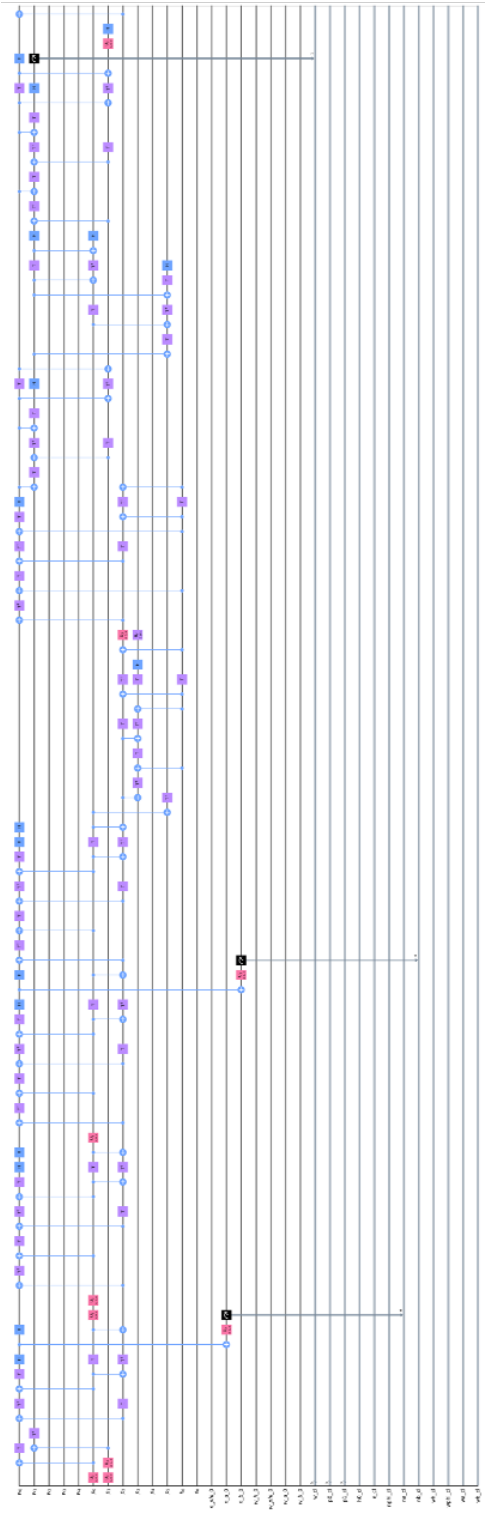

図 A.20　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (13/15)。decompose は Qiskit 標準の decomposer で行った。

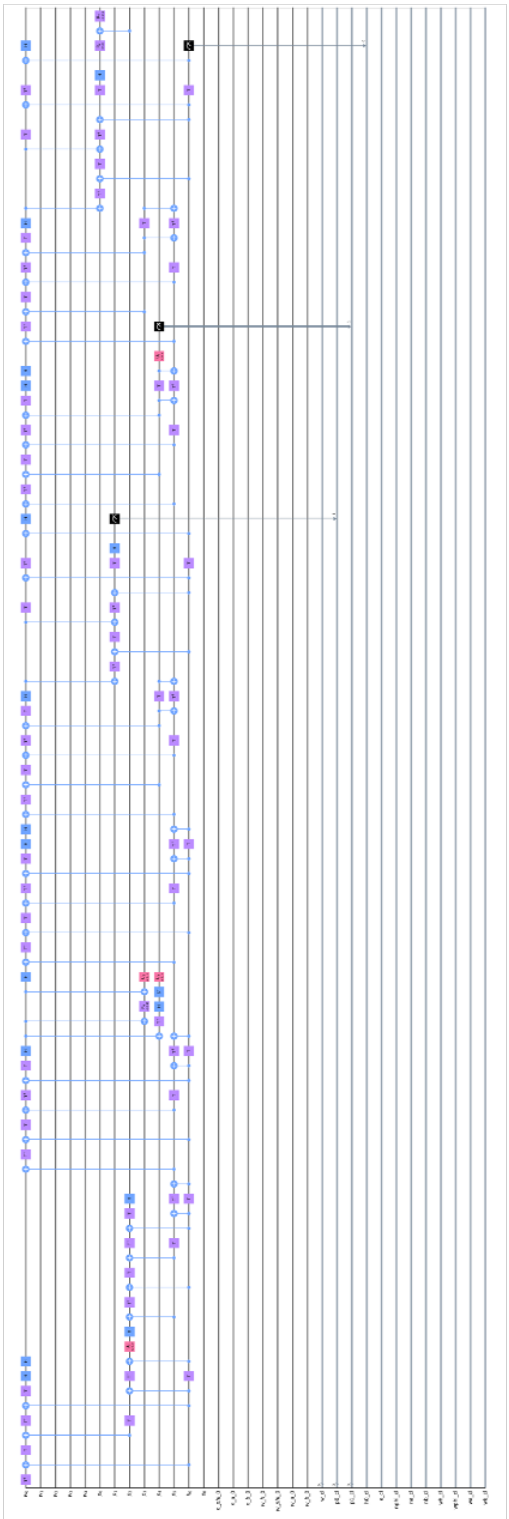

図 A.21　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (14/15)。decompose は Qiskit 標準の decomposer で行った。

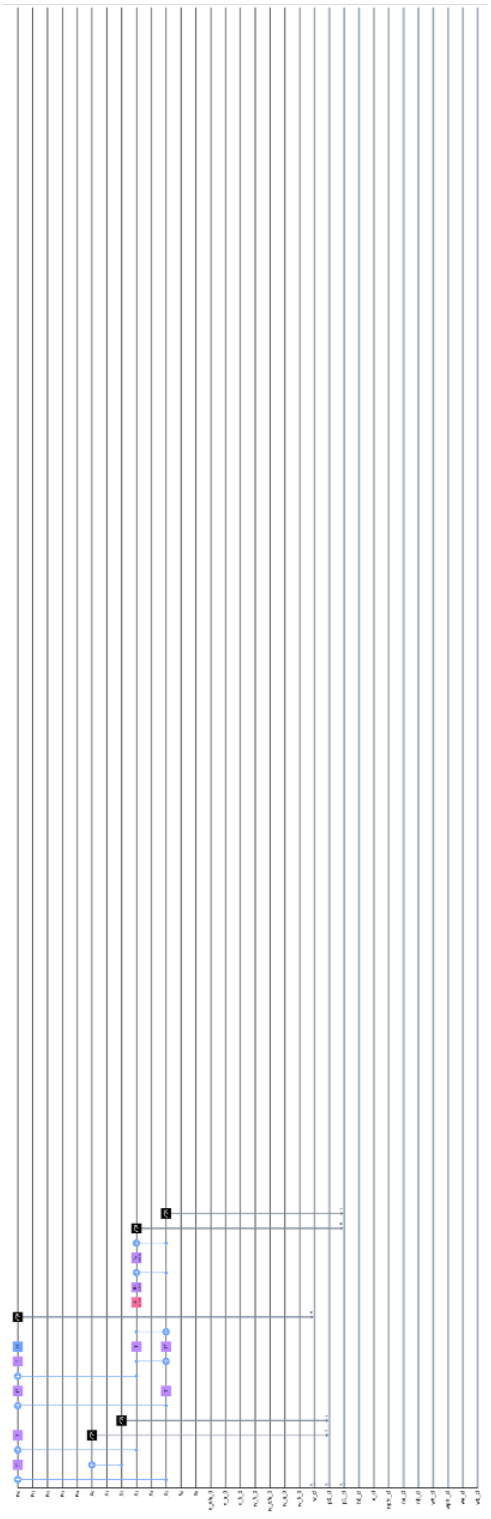

図 A.22　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (15/15)。decompose は Qiskit 標準の decomposer で行った。

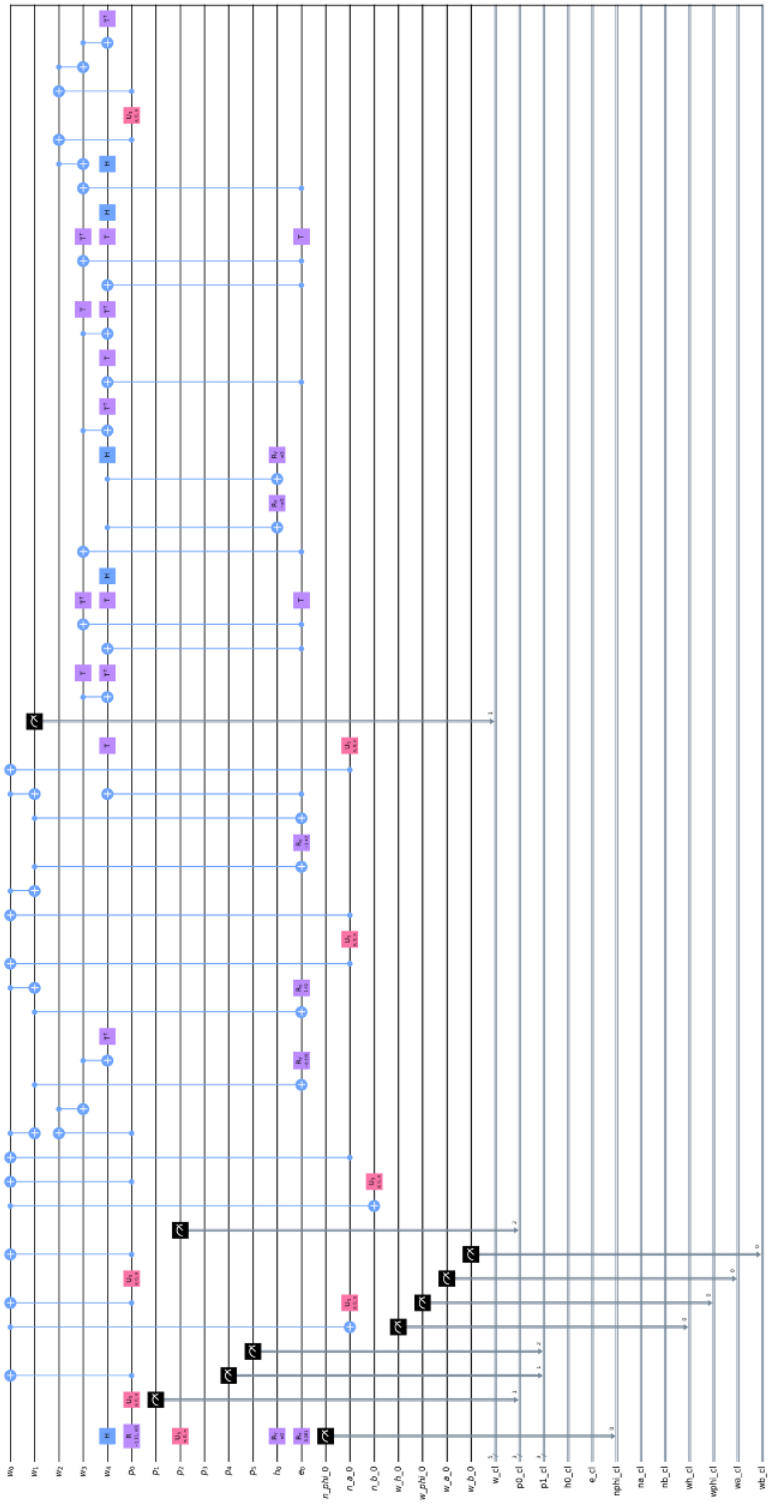

図 A.23　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用後の回路 (1/2)。decompose は Qiskit 標準の decomposer で行った。

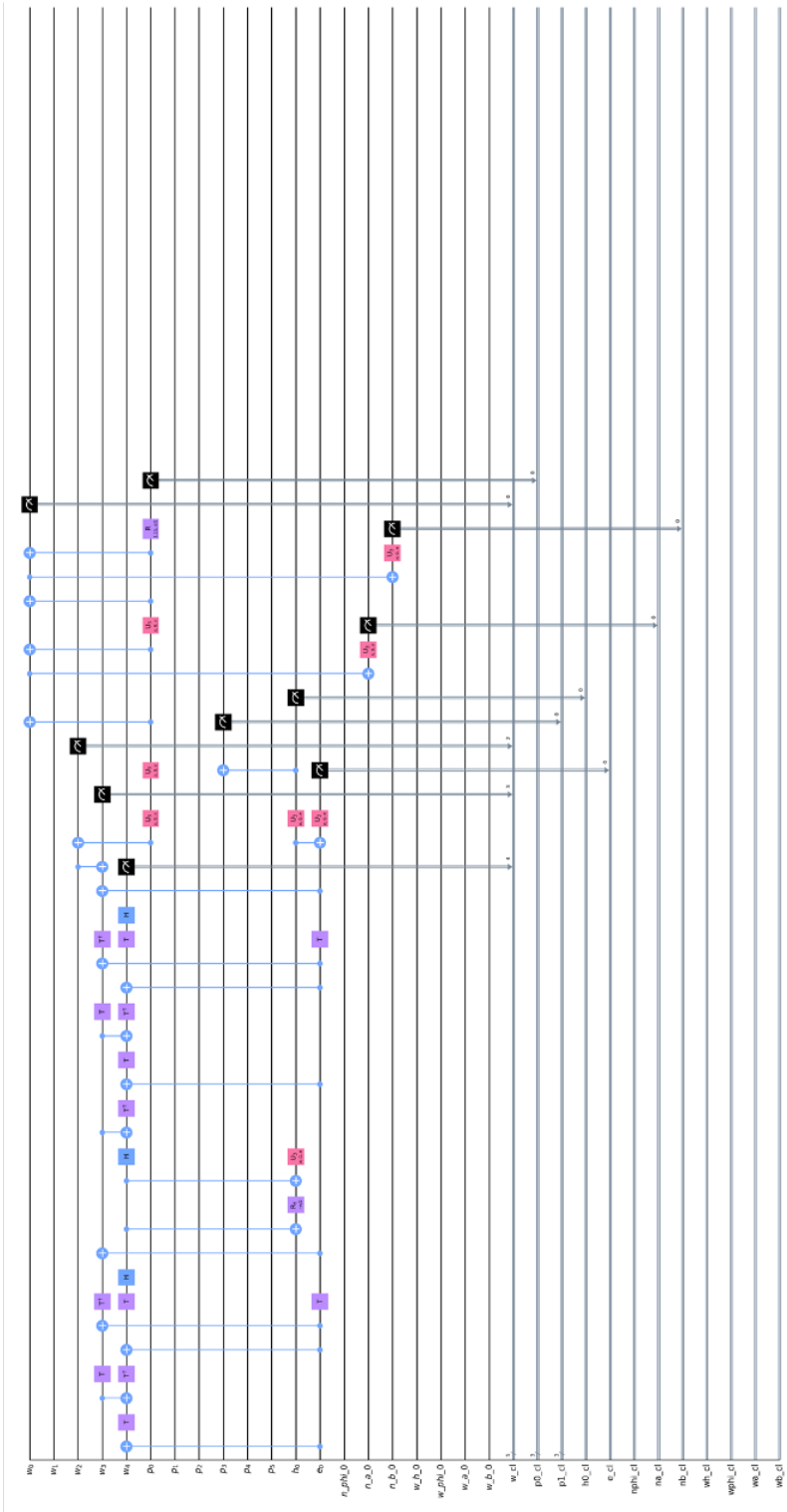

図 A.24　NQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用後の回路 (2/2)。decompose は Qiskit 標準の decomposer で行った。

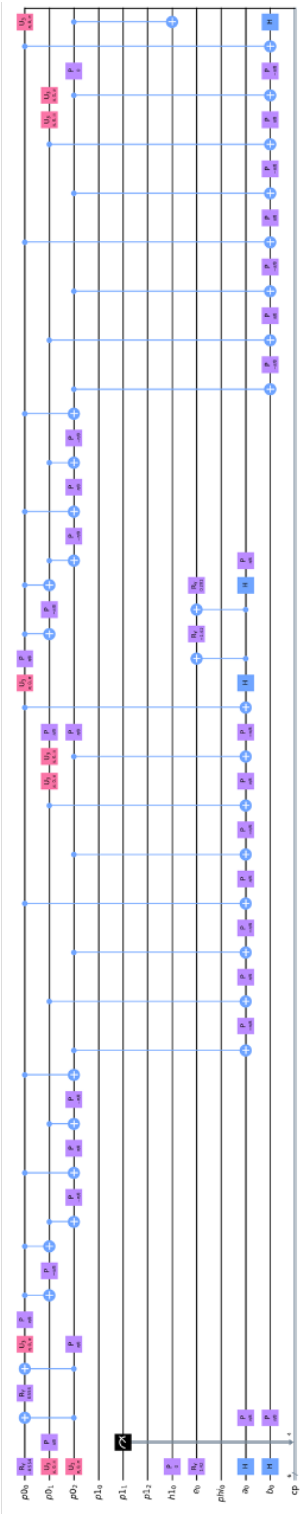

図 A.25　SQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (1/6)。decompose は Qiskit 標準の decomposer で行った。

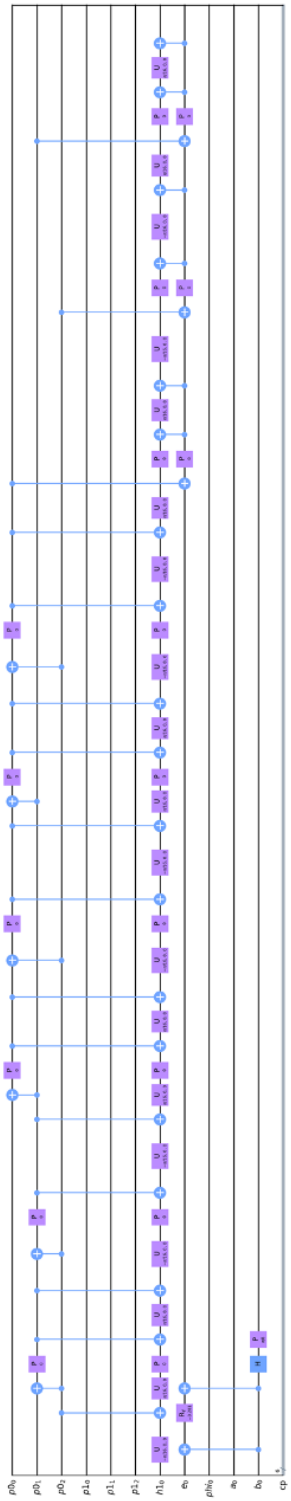

図 A.26　SQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (2/6)。decompose は Qiskit 標準の decomposer で行った。

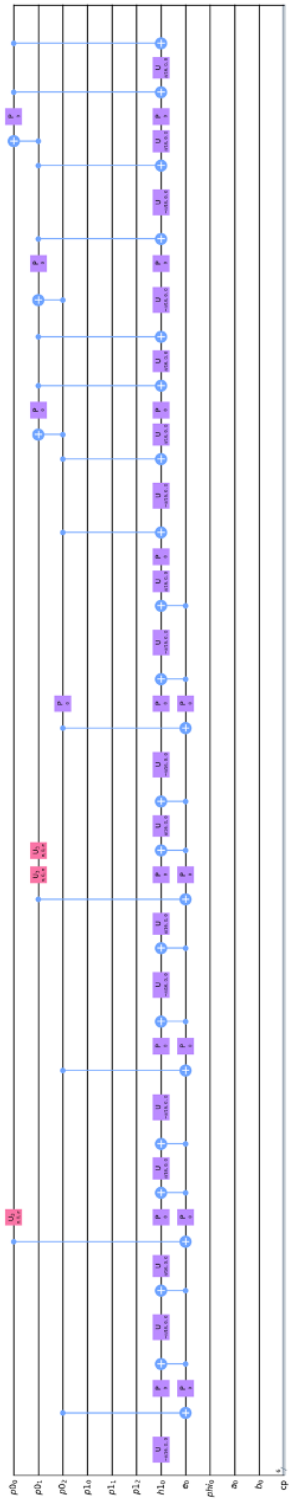

図 A.27　SQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (3/6)。decompose は Qiskit 標準の decomposer で行った。

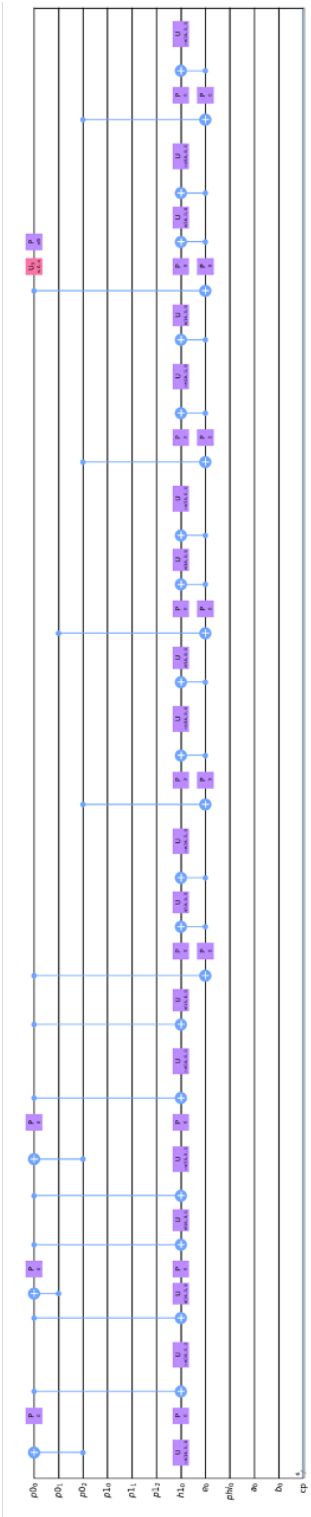

図 A.28　SQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (4/6)。decompose は Qiskit 標準の decomposer で行った。

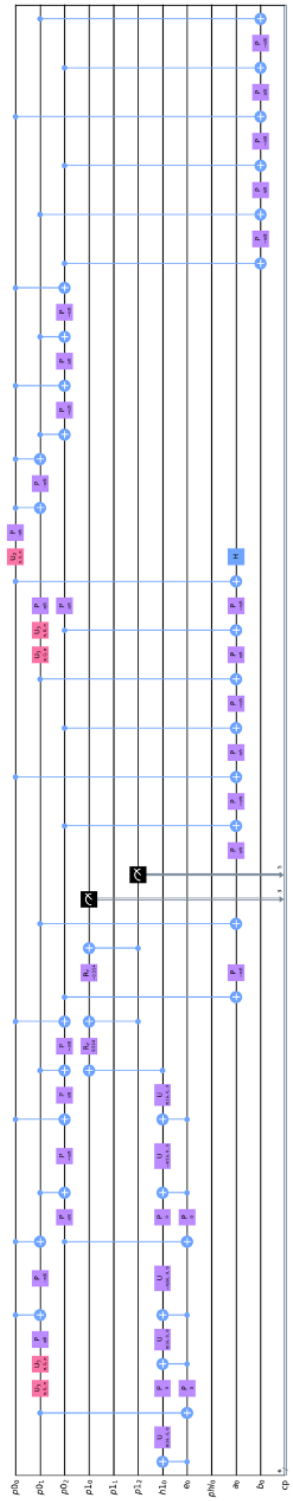

図 A.29　SQPS1 ステップシミュレーション回路 (docomposed) の AQCEL適用前の回路 (5/6)。decompose は Qiskit 標準の decomposer で行った。

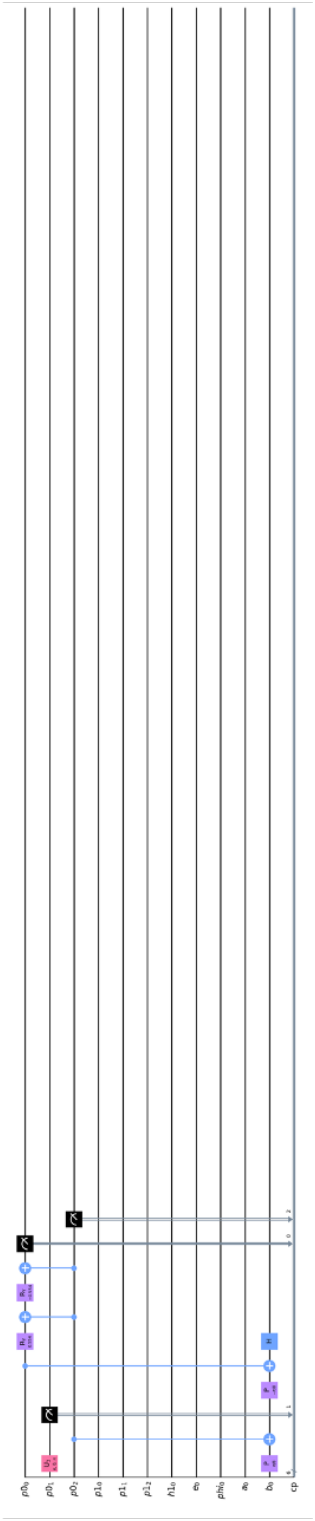

図 A.30　SQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用前の回路 (6/6)。decompose は Qiskit 標準の decomposer で行った。

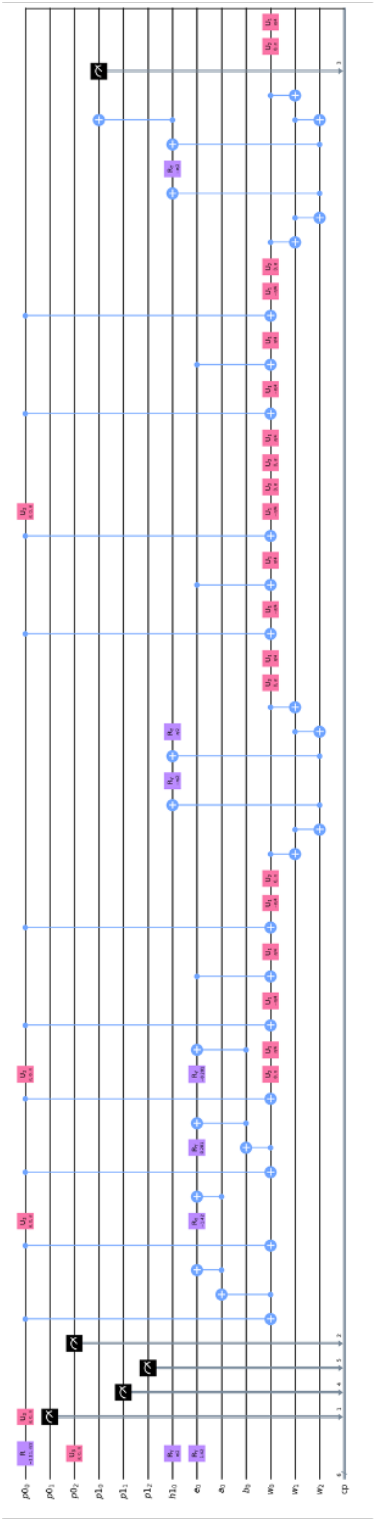

図 A.31　SQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用後の回路 (1/2)。decompose は Qiskit 標準の decomposer で行った。

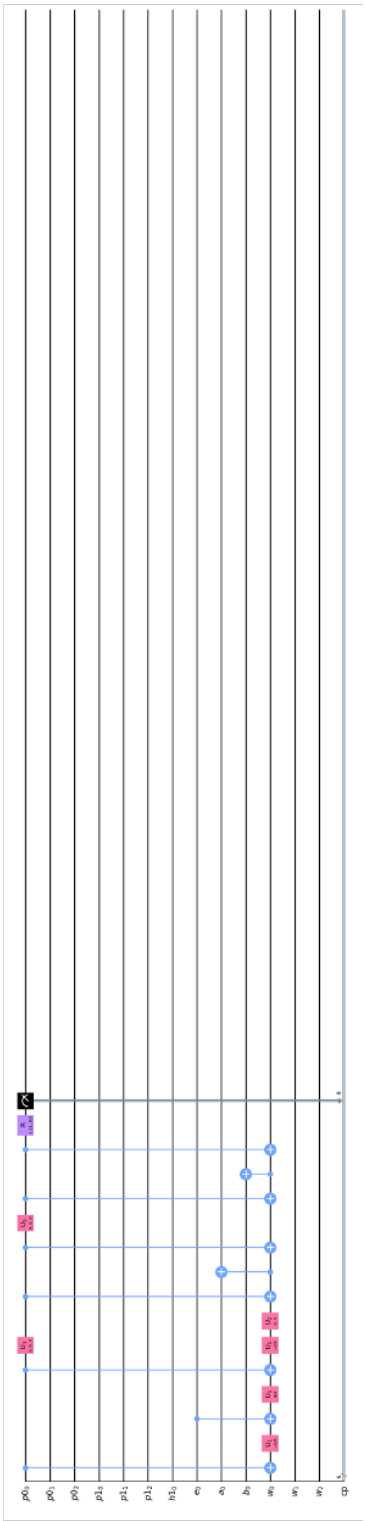

図 A.32　SQPS1 ステップシミュレーション回路 (docomposed) の AQCEL 適用後の回路 (2/2)。decompose は Qiskit 標準の decomposer で行った。