

#10

# 量子回路の最適化

Quantum Circuit Optimization

2025/09/04

Translated and modified by Daiki Murata

Created by Toshinari Itoko

# 今日のテーマ

## チャプター（リンクはYouTubeのReplay）

1. はじめ
2. 量子ビット・量子ゲート・量子回路
3. 量子テレポーテーション
4. グローバーのアルゴリズム
5. 量子位相推定
6. 量子変分アルゴリズム
7. 量子系のシミュレーション
8. 古典計算によるシミュレーション
9. 量子ハードウェア
10. 量子回路の最適化
11. 量子エラー緩和
12. 量子ユーティリティの実験Ⅰ
13. 量子ユーティリティの実験Ⅱ
14. 量子ユーティリティの実験Ⅲ

} ②  
}  
} ①  
}  
} ③

① アプリケーション



② プログラム



③ ハードウェア



# 今日のゴール

量子プログラム（回路）が量子コンピューターで実行される前に、  
**どのように「コンパイル」（変換・最適化）されるか**を理解しましょう！

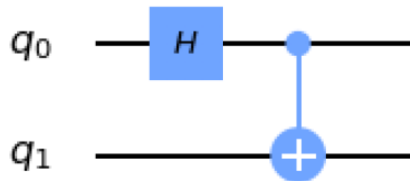
古典

```
int a = 2;  
int b = 3;  
int c = a + b;
```

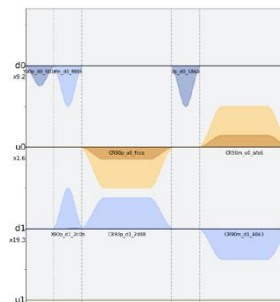
コンパイル

```
10110000 00000010 ;  
10110011 00000011 ;  
00000001 ;
```

量子



プログラム（量子回路）



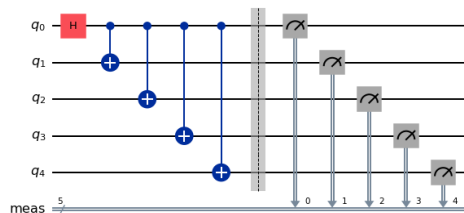
制御指示(スケジュール)

# 量子コンパイラの目標

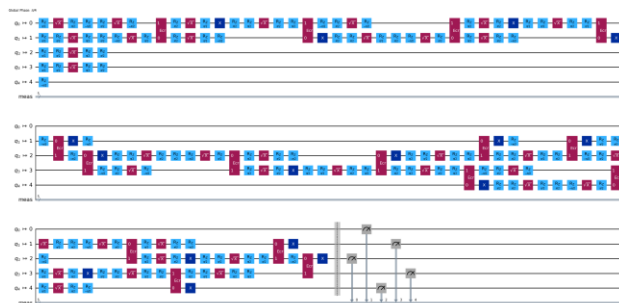
- ・ 利用可能な基底ゲートセットの制約
- ・ 量子ビットの接続性の制約

1. 使用する量子プロセッサの**制約を満たして**量子プログラムを実行可能にする
2. 量子プログラムを**最適化して**より速く、より正確に実行可能にする

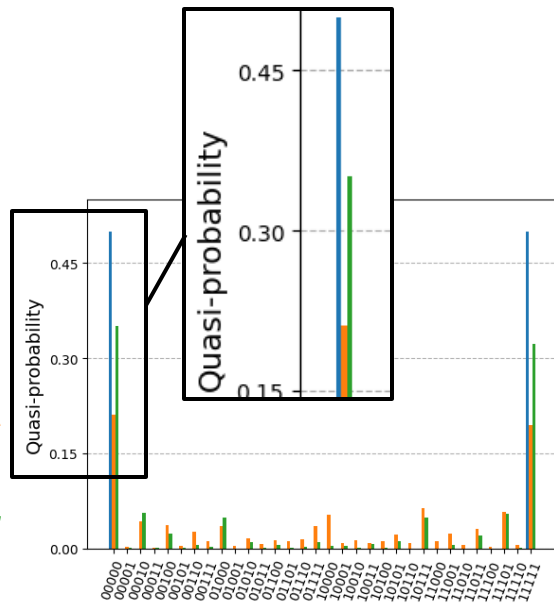
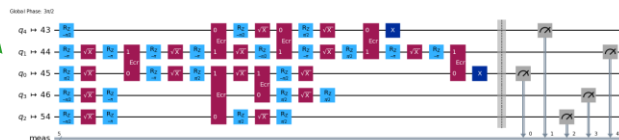
コンパイル前の回路



最適化なし



最適化あり

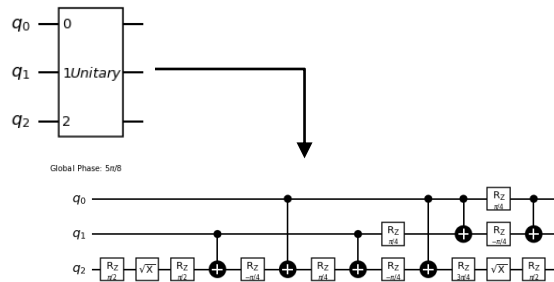


- 理論値
- 最適化なし
- 最適化あり

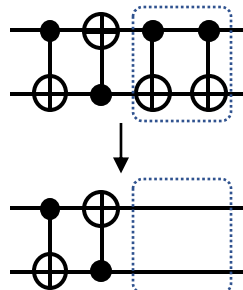
Qt

# コンパイルの流れ

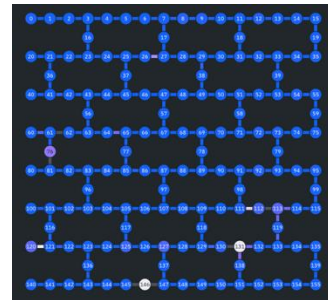
## ①量子回路の合成



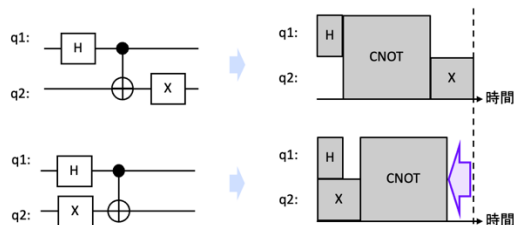
## ②量子回路の最適化



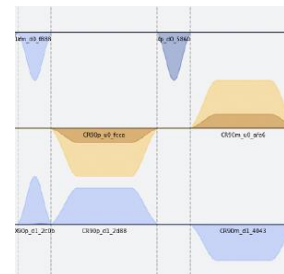
## ③量子ビットマッピング



## ④実行スケジューリング



## ⑤低水準化



# 量子回路の合成（ゲートの分解）

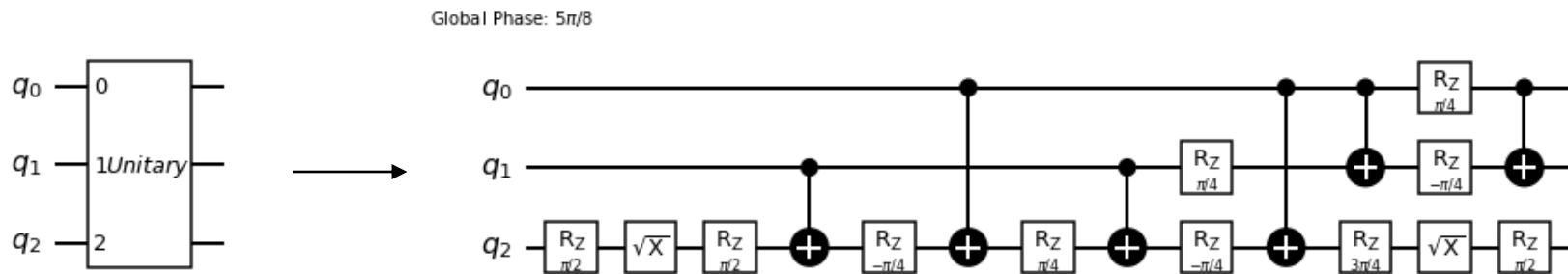
合成

最適化

マッピング

スケ  
ジュール

使用する量子プロセッサで利用可能な基底ゲートセットの制約を前提として、ハイレベルな量子ゲートをハードウェアがサポートする基底ゲートを使って合成する



## 基底ゲートセットの例

- 通常、ユニバーサルゲートセット  
任意のゲートを任意の精度に近似できるゲートセット
- 通常、1 量子ビットおよび 2 量子ビットのゲート

Z回転  
(位相シフト)



SX



CNOT



# 補足：1量子ビットゲートの分解

合成

最適化

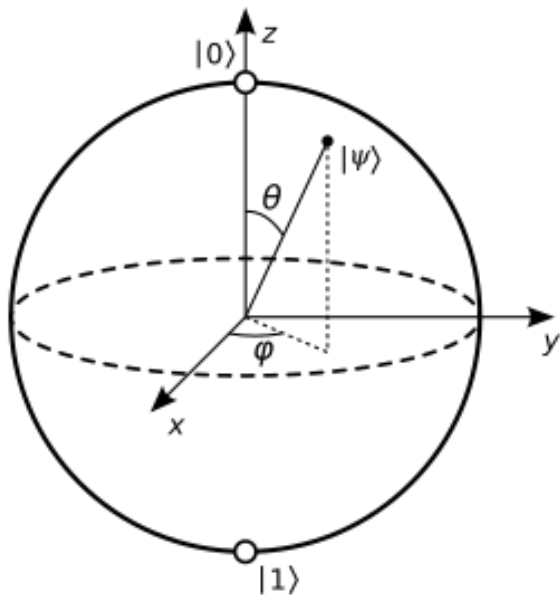
マッピング

スケ  
ジュール

全ての1量子ビットゲートの操作はBloch球上の操作であり、U3ゲートの1表現とみなすことができる

→回転ゲートの組み合わせに分解することが可能（Euler分解）

→ $R_z$  と  $SX$  ( $R_x$ ) を用いて1量子ビットを表現することが可能



$$U3(\theta, \phi, \lambda)$$

球面上の操作はZ軸回転→Y軸回転→Z軸回転  
の3回で表現できるはず

$$R_z(\phi)R_y(\theta)R_z(\lambda)$$

ハードウェアの制約がある  
例) 任意角度の $R_y$ や $R_x$ を直接サポートしないが、  
固定角度だけを基底にしている( $SX$ ゲート)

$$\underbrace{R_z(\phi)R_x\left(-\frac{\pi}{2}\right)}_{SX^\dagger} R_z(\theta) \underbrace{R_x\left(\frac{\pi}{2}\right)R_z(\lambda)}_{SX}$$

# IBM Quantumプロセッサの基底ゲート

合成

最適化

マッピング

スケ  
ジュール

- 一般的な 1 量子ビットゲート: Rz、SX、X
- 異なる2量子ビットゲート: CX、ECR、CZ

ibm\_torino

Details

Qubits  
**133**

Status  
● Online

Basis gates  
cz, id, rx, rz, rzz, sx, x

Median readout error  
3.003E-2

ibm\_brisbane

Details

Qubits  
**127**

Status  
● Online

Basis gates  
ecr, id, rz, sx, x

Median readout error  
2.112E-2

ibm\_kingston

Details

Qubits  
**156**

Status  
● Online

Basis gates  
cz, id, rx, rz, rzz, sx, x

Median readout error  
7.996E-3



# 回路の最適化

合成

最適化

マッピング

スケ  
ジュール

与えられた回路をよりシンプルな回路に変換する

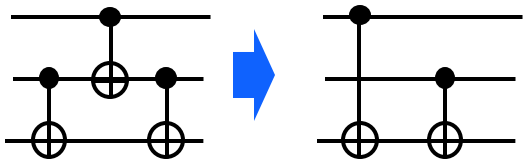
## 作戦① パターンに当てはめる

例: CNOT ゲートを減らす

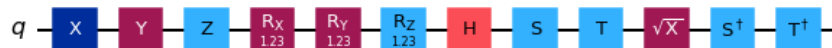
例①



例②



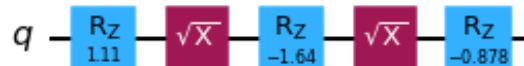
## 作戦② 1量子ビットゲートをまとめる



Euler分解の考え方を逆転

1. 1量子ビットの操作はどんなに数が多くても Bloch球面上の操作
2. 回転ゲートの組で表現できるはず

Global Phase:  $3\pi/4$



# 回路のマッピング (①レイアウト)

合成

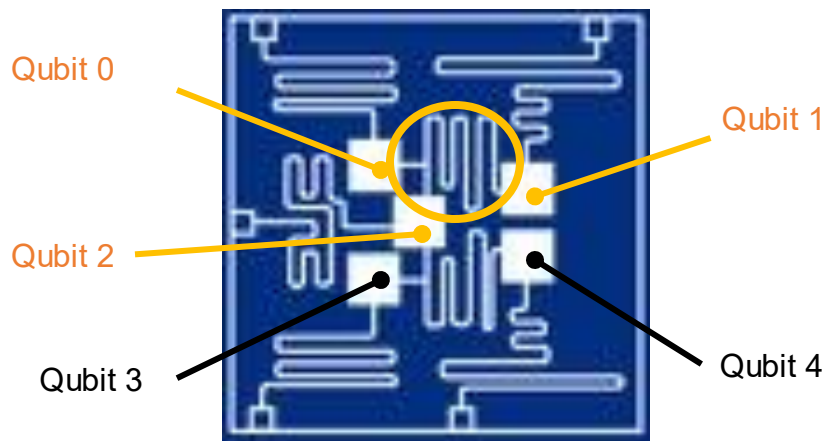
最適化

マッピング

スケ  
ジュール

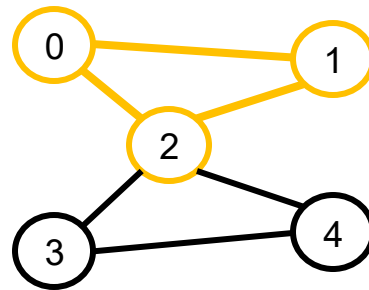
**量子コンピュータの制約：2 量子ビットゲートは、「結合された」 量子ビットでのみ実装可能  
コンパイルの作戦：**

- ① 量子回路中で頻繁に相互作用する論理量子ビット同士を、隣り合う物理量子ビットに配置 (レイアウト)
- ② 回路の途中で発生する「隣接していない量子ビット同士のCNOT」に対応できるように、SWAPゲートを挿入して論理量子ビットを物理的に移動させ、必要な位置関係を一時的に作り出す (ルーティング)



Device image of IBM Q 5 Tenerife [ibmqx4]

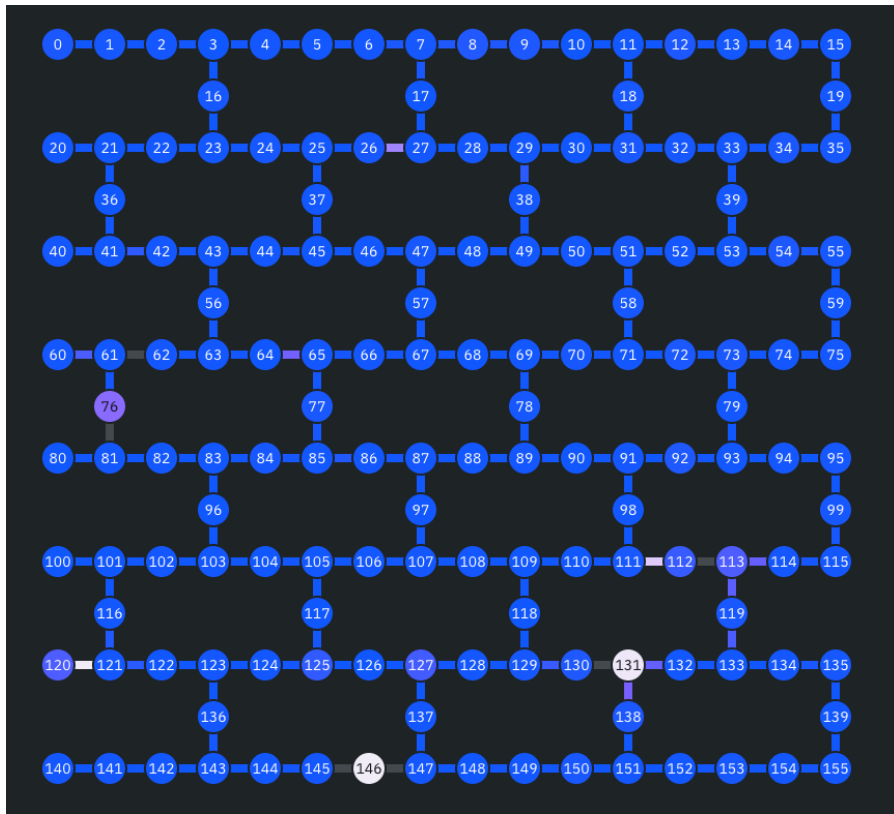
モデル化  
(抽象化)



カップリンググラフ (マップ)  
ノード ⇔ 量子ビット  
エッジ ⇔ カプラー

# 補足：IBM Quantumプロセッサの結合マップ

例：IBM\_KINGSTON 156量子ビット



# 回路のマッピング (②ルーティング)

合成

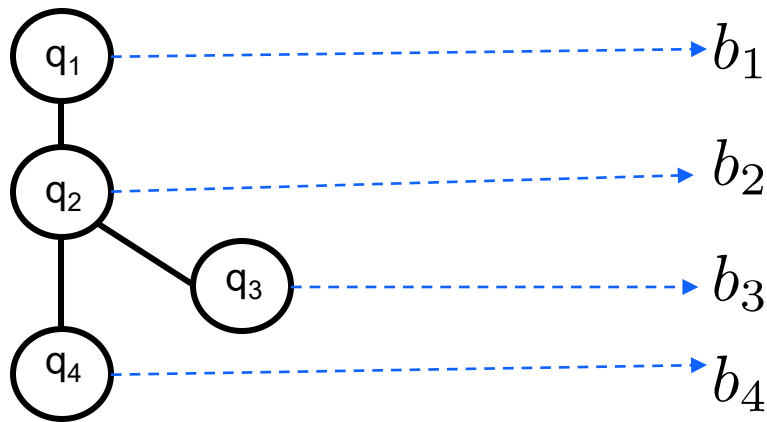
最適化

マッピング

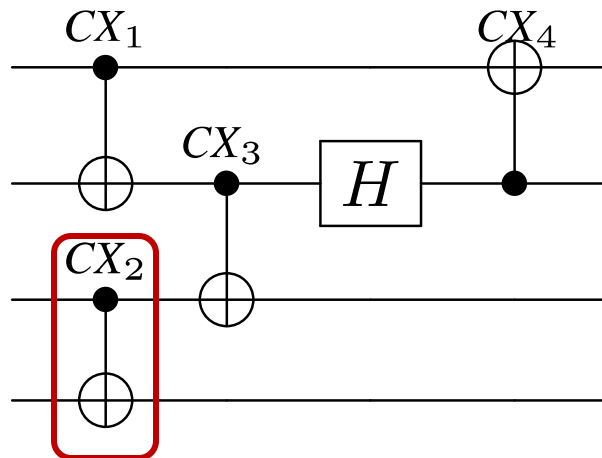
スケ  
ジュール

そうはいつでも、制約を完璧に満たす初期配置が見つからない場合がある

カップリングマップ



実行したい回路



制約違反

# 回路のマッピング (②ルーティング)

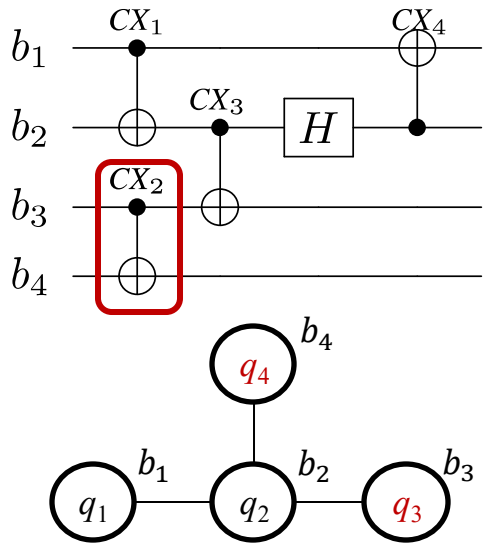
合成

最適化

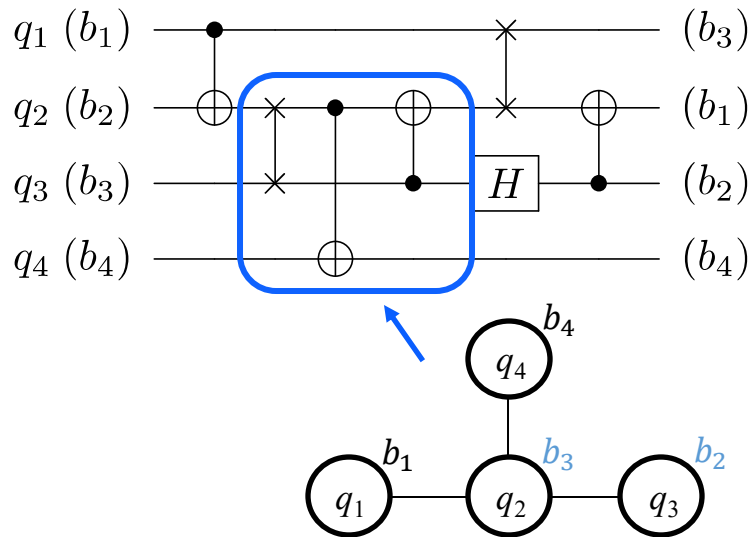
マッピング

スケ  
ジュール

2つの量子ビットの状態を入れ替えるSWAPゲートを活用することで、論理的な量子ビットの隣接関係を一時的に作り出し、制約を満たしながらアルゴリズムを実行させる



ルーティング



ただしSWAPゲートはエラー率の高いCNOTゲートの集合  
→ SWAPゲート数の最適化は必須

$$\begin{array}{c} b_1 \times b_2 \\ b_2 \times b_1 \end{array} = \begin{array}{c} \bullet \oplus \bullet \\ \oplus \bullet \oplus \end{array}$$

# 実行スケジューリング

合成

最適化

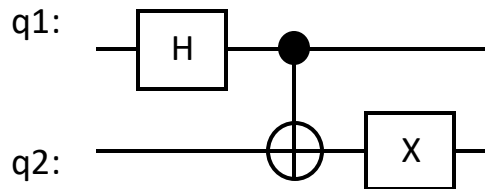
マッピング

スケ  
ジュール

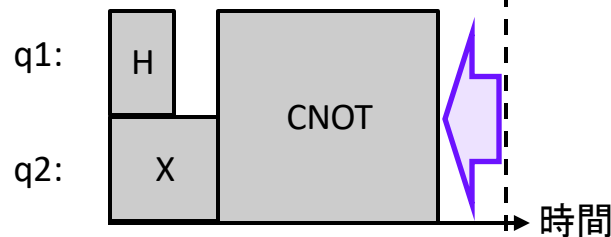
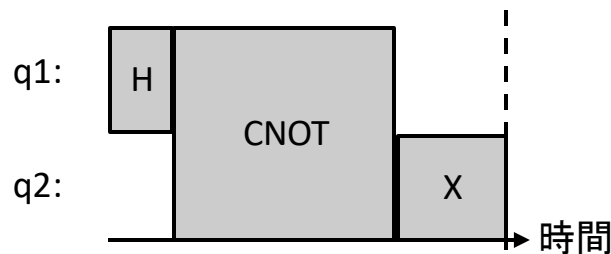
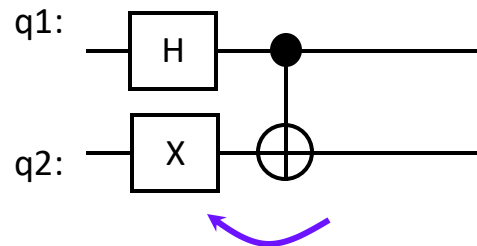
マッピングされた回路に対し、アルゴリズムの意味を変えない範囲でゲートの実行順やタイミングを最適化

1. 回路の実行時間を短縮
2. 並列実行できるゲートを同時に走らせる
3. ハードウェアの制約を守る

マッピング後



スケジューリング後

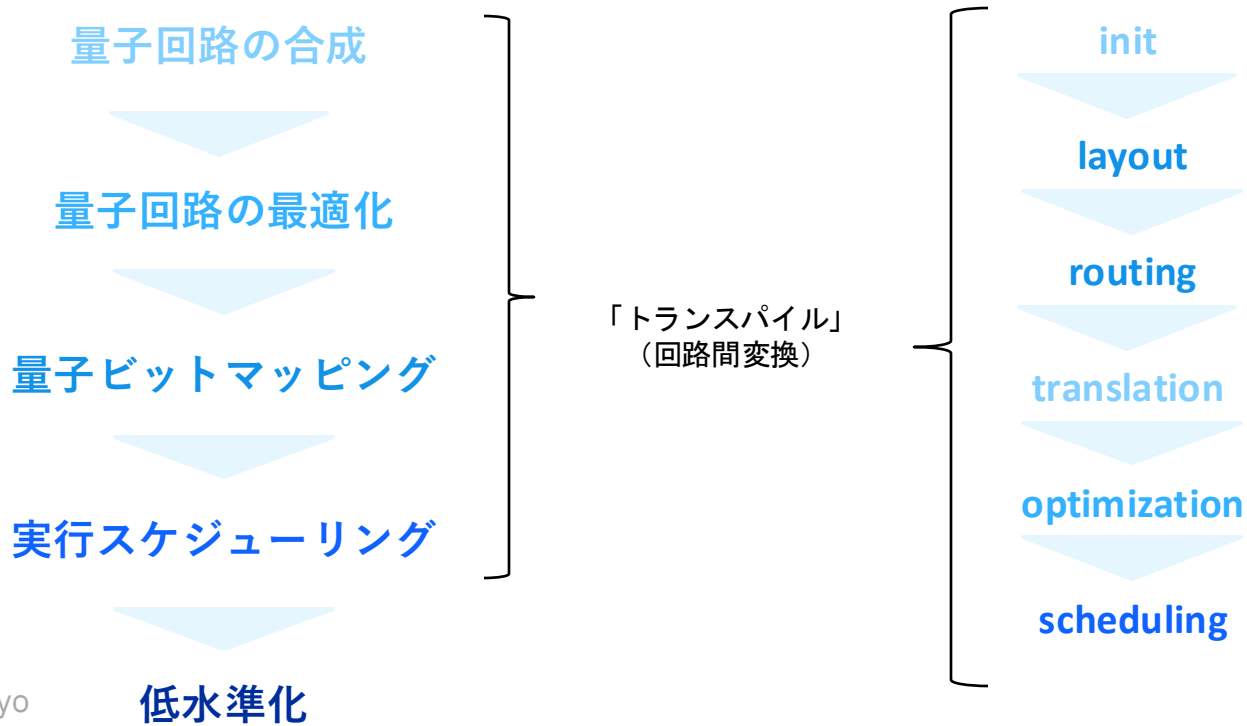


より短い=より良いスケジュール

# まとめ

量子コンパイラとは

1. 使用する量子プロセッサの制約を満たして量子プログラムを実行可能にする
2. 量子プログラムを最適化してより速く、より正確に実行可能にする



# 最後に：もっと勉強したい方へ



Quantum Tokyo へようこそ

学習コンテンツ

- Qiskit の始め方
- IBM Quantum Plaform 教材 日本語訳
- IBM Research Blog 日本語版
- (旧) Qiskitテキストブック 日本語版
- (旧)Qiskitテキストブック (Qiskitコース) 日本語版
- (旧) Qiskitドキュメント・チュートリアル 日本語版リンク集
- IBM Quantum Challenge
- Qiskit Global Summer School (Qiskit夏の学校) 資料 日本語版

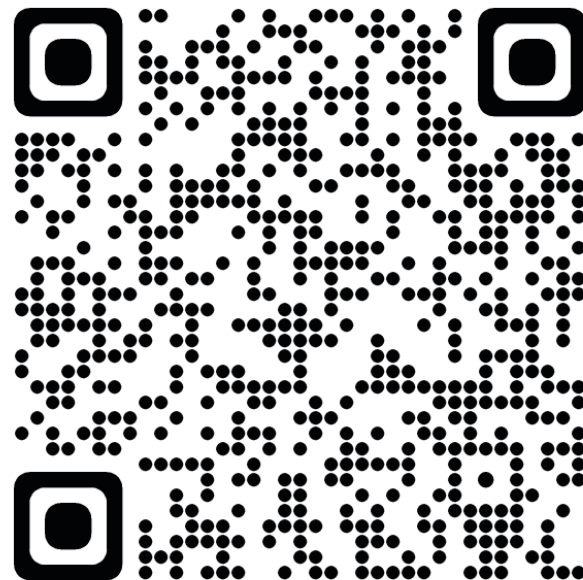
メニュー

## ユーティリティー・スケール量子コンピューティング

概要

このイベント・リプレイ・コースは、IBM Quantum®が東京大学と共同で開発し実施した14のLessonとLabで構成されています。このコースでは、量子コンピューティングにおける幅広い重要なトピックを網羅しつつ、実用規模（ユーティリティー・スケール）の量子計算を構築することに重点を置いています。最終的な結果として、2023年6月にNature誌の表紙を飾った論文と非常によく似た課題を扱います。

1. はじめに
2. [量子ビット・量子ゲート・量子回路](#)
3. [量子テレポーテーション](#)
4. グローバーのアルゴリズム
5. [量子位相推定](#)
6. 量子変分アルゴリズム
7. [量子系のシミュレーション](#)
8. 古典計算によるシミュレーション
9. 量子ハードウェア
10. 量子回路の最適化
11. 量子エラー緩和
12. 量子ユーティリティーの実験 I
13. 量子ユーティリティーの実験 II
14. 量子ユーティリティーの実験 III



<https://quantum-tokyo.github.io/introduction/courses/utility-scale-quantum-computing/overview-ja.html>





# IBM Quantum Learning

## 「Utility-scale quantum computing」の日本語解説

チャプター	日程
1. はじめ	-
2. 量子ビット・量子ゲート・量子回路	7/7(月)
3. 量子テレポーテーション	7/16(水)
4. グローバーのアルゴリズム	7/16(水)
5. 量子位相推定	7/28(月)
6. 量子変分アルゴリズム	7/28(月)
7. 量子系のシミュレーション	8/22(金)
8. 古典計算によるシミュレーション	skip
9. 量子ハードウェア	9/4(木)
10. 量子回路の最適化	9/4(木)
11. 量子エラー緩和	9/24(水)
12. 量子ユーティリティの実験 I	9/24(水)
13. 量子ユーティリティの実験 II	TBD
14. 量子ユーティリティの実験 III	TBD

Replay (YouTube) :

<https://www.youtube.com/playlist?list=PLA-UlvpIBvpuzFXRPNTqIK94kfRgYCBMs>

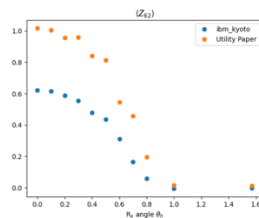
Jupyter notebookの和訳 :

<https://quantum-tokyo.github.io/introduction/courses/utility-scale-quantum-computing/overview-ja.html>

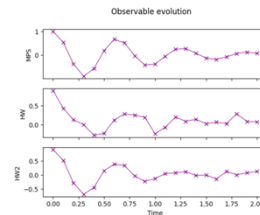
English version (IBM Quantum Platform) :

<https://quantum.cloud.ibm.com/learning/en/courses/utility-scale-quantum-computing>

I. Nature paper  
(127 qubits x 60 entangling gates)



II. 1D Transverse Ising model  
(70 qubits x 80 entangling gates)



III. The largest GHZ state challenge

