2024 年 8 月 5 日
情報システム工学実験レポート

# 量子コンピュータプログラミング

情報経営システム工学分野 B3
**学籍番号** ： 24336488
**氏名** ： 本間三暉

**共同実験者**

学籍番号:22100289　氏名: 浅野　繭
学籍番号:22105590　氏名: 筒井　翼
学籍番号:22100986　氏名: 板山修大

# 1　目的

この実験の目的は，量子コンピュータの基本原理とプログラミング技術を習得することにある．この実験では，量子ビット（キュービット）の基礎概念や，重ね合わせ，もつれといった量子特有の性質を理解し，古典的なビットとの違いを明確にする．また，基本的な量子ゲート（Hadamard ゲート，Pauli ゲート，CNOT ゲートなど）の操作方法を学び，これらを用いた量子回路の設計と実装の方法を習得することを目指す．さらに，ショアのアルゴリズムやグローバーのアルゴリズムなどの代表的な量子アルゴリズムを理解し，これらを実際にプログラムとして実装する．

プログラミングの実践においては，Qiskit や Cirq などの量子コンピュータプログラミングフレームワークを用いたプログラミングスキルを習得し，実際の量子コンピュータ上でプログラムを実行する経験を得ることを目的とする．これにより，量子計算が解決できる問題の種類や応用分野についても検討し，従来のコンピュータでは実現困難な計算の可能性を理解する．

この実験を通じて，参加者は量子コンピュータの基礎から応用までの広範な知識を習得し，将来的に量子技術の研究や実務に貢献できる能力を身につけることを目指す．

# 2　原理

## 2.1　量子論理回路

### 概要

量子論理回路は，量子ビット（キュービット）を操作するための基本単位である．従来の論理回路が古典ビットを用いるのに対し，量子論理回路は量子力学の原理に基づいて動作する．これにより，並列処理能力や量子重ね合わせ，量子もつれといった特性を活用することが可能である．

### 量子ビット

量子ビットは，0 と 1 の状態を同時にとることができる量子重ね合わせの性質を持つ．これは，$|0\rangle$ と $|1\rangle$ の状態の重ね合わせとして表現される．量子ビットの状態は式 (1) のように表される．

$$|\,\psi\rangle = \alpha|\,0\rangle + \beta|\,1\rangle \tag{1}$$

ここで，$\alpha$ と $\beta$ は複素数であり，$|\alpha|^2 + |\beta|^2 = 1$ である．

### 量子ゲート

量子論理回路は，量子ゲートと呼ばれる基本的な演算を組み合わせて構成される．代表的な量子ゲートには以下のようなものがある．

#### Hadamard ゲート (H ゲート)

量子ビットを重ね合わせ状態に変換する．

$$H|\,0\rangle = \frac{1}{\sqrt{2}}\left(|\,0\rangle + |\,1\rangle\right) \tag{2}$$

$$H|\,1\rangle = \frac{1}{\sqrt{2}}\left(|\,0\rangle - |\,1\rangle\right) \tag{3}$$

#### Pauli-X ゲート

古典論理回路の NOT ゲートに相当する．

$$X|0\rangle = X|1\rangle \tag{4}$$
$$X|1\rangle = X|0\rangle \tag{5}$$

#### CNOT ゲート (制御 NOT ゲート)

2 量子ビットゲートで，制御ビットが 1 のときにターゲットビットを反転させる．

$$CNOT(|a\rangle \otimes |b\rangle) = |a\rangle \otimes (a \oplus b) \tag{6}$$

## 量子回路の設計

量子回路は，量子ビットを初期化し，量子ゲートを適用して計算を行い，最終的に測定するという一連のステップで設計される．これにより，量子ビットの状態を古典ビットに変換し，結果を得ることができる．

## 2.2 グローバーのアルゴリズム

グローバーのアルゴリズムは，無条件検索問題（非構造的データベースの検索）に対する量子アルゴリズムである．古典的なアルゴリズムでは $O(N)$ の時間がかかる検索問題を，グローバーのアルゴリズムでは $O(\sqrt{N})$ の時間で解くことができる．

#### アルゴリズムのステップ

### 2.2.1 初期化

すべての量子ビットを $|0\rangle$ に初期化し，Hadamard ゲートを適用してすべての状態の重ね合わせを生成する．

$$H^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \tag{7}$$

### 2.2.2 オラクル (Oracle) の適用

オラクルは，解となる状態を反転させる役割を果たす．状態 $|x\rangle$ が解である場合，$-1$ の位相が付加される．

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle \tag{8}$$

### 2.2.3 拡散変換 (Amplitude Amplification)

拡散変換は，解の確率振幅を増幅させる操作である．これは，状態ベクトル全体を平均値に対して反転させることによって実現される．

$$U_s = 2|\psi\rangle\langle\psi| - I \tag{9}$$

#### 2.2.4 反復

オラクルと拡散変換を $\sqrt{N}$ 回繰り返すことで，解の確率振幅を最大化する．

#### 2.2.5 測定

最後に量子ビットを測定することで，高確率で解が得られる．

## 3 論理パズル

グループメンバー全員で相談して決めたので，誰がどこをやるかみたいな明確な割振りはしていません．

### 3.1 2 人用

殺人事件が起きました．容疑者は二人おり，それぞれ次のように述べています．

エリカ「アリスが犯人です」
アリス「私たちのどちらも犯人です」

犯人は一人です．一人が本当のことを言っていますが，もう一人は嘘をついています．犯人は誰でしょう．

### 3.2 3 人用

コナン君の目の前で歩美ちゃん，元太君，光彦君が突然次のようなことを述べ始めました．

歩美「元太君は黒幕じゃないよ」
元太「俺は黒幕じゃない」
光彦「元太君は黒幕です」

黒幕を特定してください．

### 3.3 4 人用

黒の組織でスパイが紛れ込んでいることが分かりました．そこで，スパイ疑惑をかけられているキール，ジン，ライ，バーボンは次のように主張しています．

キール「ライがスパイよ」
ジン「俺はスパイじゃねぇ」
ライ「バーボンはスパイだ」
バーボン「ライはうそつきだ」

彼らのうち 3 人が嘘をついています．スパイを特定してください．

# 4　パズルの論理式

節 3 と同様に，グループメンバー全員で相談しながら考えたので，明確な割振りはしていません．基本的に犯人である事象を $X$，嘘をついている事象を $Y$ とする．

## 4.1　2 人用

エリカが犯人である ： $X_1$
アリスが犯人である ： $X_2$
エリカが正直である ： $Y_1$
アリスが正直である ： $Y_2$
エリカ　「犯人はアリスである」： $Y_1 \cdot X_2 \cdot (\overline{\overline{Y_1}} \cdot \overline{\overline{X_2}}) = Y_1 \cdot X_2$
アリス　「二人とも犯人である」： $Y_1 \cdot (X_1 \cdot X_2) \cdot \overline{X_2} = X_1 \cdot Y_1 \cdot \phi$(恒偽命題)

## 4.2　3 人用

歩美が黒幕である ： $X_1$
元太が黒幕である ： $X_2$
光彦が黒幕である ： $X_3$
歩美が正直である ： $Y_1$
元太が正直である ： $Y_2$
光彦が正直である ： $Y_3$
歩美　「元太君は黒幕ではない」： $Y_1 \cdot \overline{X_2} \cdot \overline{\overline{X_2}} \cdot \overline{X_2} = Y_2 \cdot \phi$(恒偽命題)
元太　「俺は黒幕じゃない」　　： $Y_2 \cdot \overline{X_2} \cdot \overline{\overline{X_2}} = Y_2 \cdot \phi$(恒偽命題)
光彦　「元太君は黒幕である」　： $Y_3 \cdot X_2 \cdot \overline{\overline{X_2}} = Y_3 \cdot X_2$

## 4.3　4 人用

キールがスパイである　： $X_1$
ジンがスパイである　　： $X_2$
ライがスパイである　　： $X_3$
バーボンがスパイである ： $X_4$
キールが正直である　　： $Y_1$
ジンが正直である　　： $Y_2$
ライが正直である　　： $Y_3$
バーボンが正直である　： $Y_4$
キール　「ライがスパイよ」　　： $Y_1 \cdot X_3 \cdot \overline{\overline{X_2}} \cdot \overline{X_4} \cdot \overline{\overline{Y_3}}$
ジン　「俺はスパイじゃねぇ」： $Y_2 \cdot \overline{X_2} \cdot \overline{X_3} \cdot \overline{X_4} \cdot \overline{\overline{Y_3}}$
ライ　「バーボンはスパイ」　： $Y_3 \cdot X_4 \cdot \overline{X_3} \cdot \overline{\overline{X_2}} \cdot \overline{\overline{Y_3}}$
バーボン　「ライは嘘つきさ」　： $Y_4 \cdot \overline{Y_3} \cdot \overline{X_3} \cdot \overline{\overline{X_2}} \cdot \overline{X_4}$

# 5　量子回路

2 人，3 人及び，4 人の場合の論理パズルを解く量子論理回路について説明を行う．

## 5.1　2 人用

2 人用の論理パズルの真理値表を表 1 に，量子論理回路を図 1 に，ソースコードをソースコード 1 に示す.

**表 1:** 2 人用真理値表

| | | 犯人は？ | | 正直者は？ | |
|---|---|---|---|---|---|
| | | エリカ | アリス | エリカ | アリス |
| 誰の証言が正しい | エリカ | | ○ | ○ | |
| | アリス | ○ | ○ | | ○ |

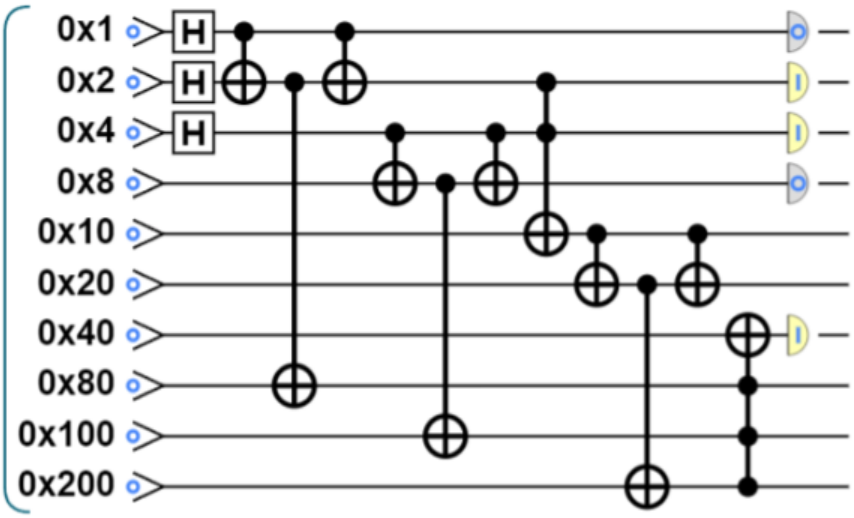


**図 1:** 2 人用量子論理回路

**ソースコード 1:** 2 人用 QCEngin

```
qc.reset(10);
qc.write(0b0000000000);
\\ 定義
var x1=0x1;
var x2=0x2;
var y1=0x4;
var y2=0x8;
var comment1=0x10;
var comment2=0x20;
var result=0x40;
var x1_xor_x2=0x80;
var y1_xor_y2=0x100;
var comment1_xor_comment2=0x200;

```

```
15  qc.had(x1|x2|y1);
16
17  xor2(x1,x2,x1_xor_x2);
18
19  xor2(y1,y2,y1_xor_y2);
20
21  and2(y1,x2,comment1);
22
23  xor2(comment1,comment2,comment1_xor_comment2);
24
25  qc.cnot(result,x1_xor_x2|y1_xor_y2|comment1_xor_comment2);
26
27  qc.read(x1|x2|y1|y2|result);
28
29  \\--------------------------
30  \\ 関数
31  \\--------------------------
32  function and2(x,y,result){
33      qc.cnot(result,x|y);
34  }
35
36  function or2(x,y,result){
37      qc.not(x|y);
38      qc.cnot(result,x|y);
39      qc.not(x|y|result);
40  }
41
42  function xor2(x,y,result){
43      qc.cnot(y,x);
44      qc.cnot(result,y);
45      qc.cnot(y,x);
46  }
```

まず，量子コンピュータの 10 量子ビットをリセットし，すべての量子ビットを初期状態 (0) に設定する．各変数は異なる量子ビットを表しており，x1 や x2 などがそれに該当する．次に，x1，x2，y1 のビットにアダマール変換を適用し，これらのビットを重ね合わせ状態にする．ただし，y2 については，問題の設定上アリスが正直であることがないため，アダマール変換は適用しなかった．

続いて，x1 と x2 の XOR 結果を x1_xor_x2 に，y1 と y2 の XOR 結果を y1_xor_y2 に保存する．これは，それぞれ犯人が一人であることと正直者が一人であることを示している．次に，y1 と x2 の AND 結果を comment1 に保存し，エリカの証言「アリスが犯人です」を適応する．さらに，エリカの証言とアリスの証言の XOR 結果を comment1_xor_comment2 に保存し，アリスの証言「私たちのどちらも犯人です」を適応する．

次に，XOR された証言の結果が嘘であることを示すために result に保存する．最後に，x1，x2，y1，y2，および result のビットを読み取り，最終結果を得る．このようにして，特定の条件を満たすかどうかを検証し，エリカとアリスの証言の整合性を確認し，犯人を判定する．

## 5.2　3 人用

人用の論理パズルの真理値表を表 2 に，量子論理回路を図 2 に，ソースコードをソースコード 2 に示す.

**表 2:** 3 人用真理値表

| | | 黒幕は？ | | | 正直者は？ | | |
|---|---|---|---|---|---|---|---|
| | | あゆみ | げんた | みつひこ | あゆみ | げんた | みつひこ |
| 誰の証言が正しい | あゆみ | | × | | ○ | × | ○ |
| | げんた | | × | | × | ○ | ○ |
| | みつひこ | | ○ | | × | × | ○ |

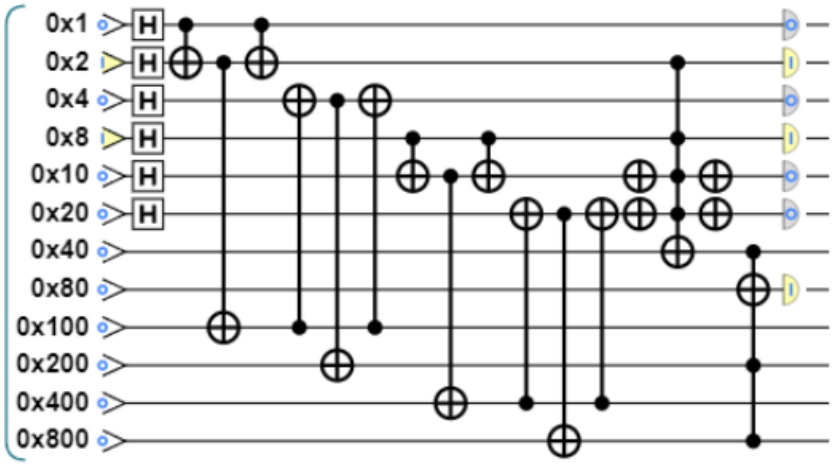


**図 2:** 3 人用量子論理回路

**ソースコード 2:** 3 人用 QCEngin

```
1   qc.reset(12);
2   qc.write(0b0000001010);
3
4   var x1=0x1;
5   var x2=0x2;
6   var x3=0x4;
7   var y1=0x8;
8   var y2=0x10;
9   var y3=0x20;
10  var comment=0x40;
11  var result=0x80;
12  var x1_xor_x2=0x100;
13  var x12_xor_x3=0x200;
14  var y1_xor_y2=0x400;
15  var y12_xor_y3=0x800;
```

```
16
17
18  qc.had(x1|x2|x3|y1|y2|y3);
19
20  xor2(x1,x2,x1_xor_x2);
21  xor2(x1_xor_x2,x3,x12_xor_x3);
22  xor2(y1,y2,y1_xor_y2);
23  xor2(y1_xor_y2,y3,y12_xor_y3);
24
25  qc.not(y2|y3);
26  qc.cnot(comment,y1|x2|y2|y3);
27  qc.not(y2|y3);
28
29  qc.cnot(result,x12_xor_x3|y12_xor_y3|comment);
30  qc.read(x1|x2|x3|y1|y2|y3|result);
31
32  function and2(x,y,result){
33      qc.cnot(result,x|y);
34  }
35
36  function or2(x,y,result){
37      qc.not(x|y);
38      qc.cnot(result,x|y);
39      qc.not(x|y|result);
40  }
41
42  function xor2(x,y,result){
43      qc.cnot(y,x);
44      qc.cnot(result,y);
45      qc.cnot(y,x);
46  }
```

まず，量子コンピュータの 12 量子ビットを初期化し，特定のビット（2 番目と 4 番目）を 1 に設定する．これにより，初期状態がセットされる．各変数は異なる量子ビットを表しており，x1 や x2 などがそれに対応する．次に，x1，x2，x3，y1，y2，y3 のビットにアダマール変換を適用して，これらのビットを重ね合わせ状態にする．続いて，x1 と x2 の XOR 結果を x1_xor_x2 に，x1_xor_x2 と x3 の XOR 結果を x12_xor_x3 に保存することで特定の条件を表現し，y1 と y2 の XOR 結果を y1_xor_y2 に，y1&_xor_&y2 と y3 の XOR 結果を y12_xor_y3 に保存することで別の条件を表現する．さらに，y2 と y3 のビットを反転させ，反転させた y2 と y3 を含めた AND 結果を comment に保存し，再度 y2 と y3 のビットを元に戻す．そして，XOR された結果と comment を組み合わせ，その結果を result に保存することで特定の条件が満たされていることを示し，最後に x1，x2，x3，y1，y2，y3，および result のビットを読み取り最終結果を得る．また，AND，OR，XOR の操作を行う関数を定義し，and2 は x と y の AND 結果を result に，or2 は x と y の OR 結果を result に，xor2 は x と y の XOR 結果を result に保存する．このようにして，特定の条件を検証するために量子ビットを操作し，設定された条件の下で証言の整合性を確認し，特定の結論に達する．

## 5.3　4 人用

4 人用の論理パズルの真理値表を表 3 に，量子論理回路を図 3 に，ソースコードをソースコード 3 に示す.

**表 3:** 4 人用真理値表

| | | スパイは？ | | | | 正直者は？ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ジン | ライ | バーボン | キール | ジン | ライ | バーボン | キール |
| 証言の正しさ | ジン | × | × | ○× | | ○ | ○ | | |
| | ライ | ○ | × | ○ | | | ○ | | |
| | バーボン | ○ | × | × | | | × | ○ | |
| | キール | ○ | ○ | × | | | ○ | | ○ |

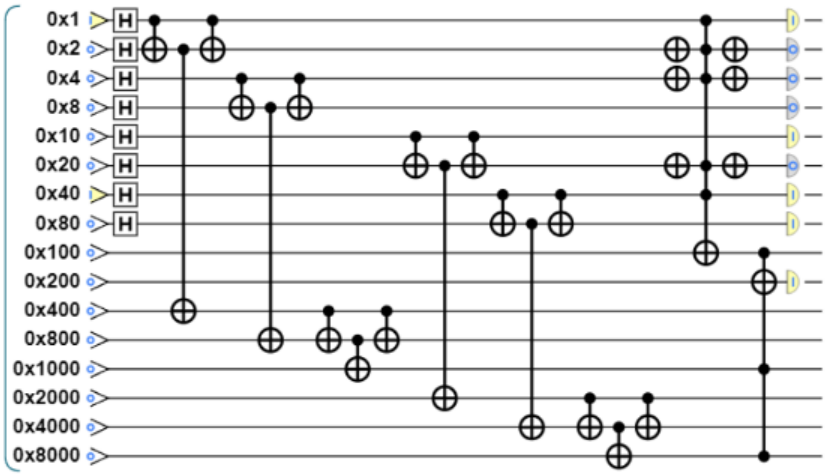


**図 3:** 4 人用量子論理回路

**ソースコード 3:** 2 人用 QCEngin

```
  qc.reset(16);
qc.write(0b0000000001000001);

var x1=0x1;
var x2=0x2;
var x3=0x4;
var x4=0x8;
var y1=0x10;
var y2=0x20;
var y3=0x40;
var y4=0x80;
var comment=0x100;
var result=0x200;
```

```
14 var x1_xor_x2=0x400;
15 var x3_xor_x4=0x800;
16 var x12_xor_x34=0x1000;
17 var y1_xor_y2=0x2000;
18 var y3_xor_y4=0x4000;
19 var y12_xor_y34=0x8000;
20
21 qc.had(x1|x2|x3|x4|y1|y2|y3|y4);
22
23 xor2(x1,x2,x1_xor_x2);
24 xor2(x3,x4,x3_xor_x4);
25 xor2(x1_xor_x2,x3_xor_x4,x12_xor_x34);
26 xor2(y1,y2,y1_xor_y2);
27 xor2(y3,y4,y3_xor_y4);
28 xor2(y1_xor_y2,y3_xor_y4,y12_xor_y34);
29
30 qc.not(x2|x3|y2);
31 qc.cnot(comment,y3|x2|y2|x3|x1);
32 qc.not(x2|x3|y2);
33
34 qc.cnot(result,x12_xor_x34|y12_xor_y34|comment);
35 qc.read(x1|x2|x3|x4|y1|y2|y3|y4|result);
36
37 function and2(x,y,result){
38     qc.cnot(result,x|y);
39 }
40
41 function or2(x,y,result){
42     qc.not(x|y);
43     qc.cnot(result,x|y);
44     qc.not(x|y|result);
45 }
46
47 function xor2(x,y,result){
48     qc.cnot(y,x);
49     qc.cnot(result,y);
50     qc.cnot(y,x);
51 }
```

最初に，量子コンピュータの 16 量子ビットをリセットし，特定のビット（最下位ビットと 7 番目のビット）を 1 に設定して初期状態を確立する（正解の確認のため）．各変数は異なる量子ビットを表し，x1 から x4，および y1 から y4 がそれに該当する．次に，x1，x2，x3，x4，y1，y2，y3，y4 のビットにアダマール変換を適用し，それらを重ね合わせ状態にする．

その後，x1 と x2 の XOR 結果を x1_xor_x2 に，x3 と x4 の XOR 結果を x3_xor_x4 に保存し，さらにこれらの結果を XOR して x12_xor_x34 に保存することで，特定の条件を表現する．同様に，y1 と y2 の XOR 結果を y1_xor_y2 に，y3 と y4 の XOR 結果を y3_xor_y4 に保存し，さらにこれらの結果を XOR して y12_xor_y34 に保存することで，別の条件を示す．

続いて，x2，x3，y2 のビットを反転させ，反転させた状態で y3，x2，y2，x3，x1 の AND 結果を comment に保存し，再度ビットを反転して元に戻す．次に，XOR された結果と comment を組み合わせ，その結果を result に保存することで特定の条件が満たされていることを示す．最後に，x1，x2，x3，x4，y1，y2，y3，y4，および result のビットから最終結果を得る．

プログラム上で，AND，OR，XOR の操作を行う関数も定義されており，and2 は x と y の AND 結果を result に，or2 は x と y の OR 結果を result に，xor2 は x と y の XOR 結果を result に保存する．このように，特定の条件を検証するために量子ビットを操作し，特定の前提の下で証言の整合性を確認し結論を導き出す．

# 6　実行結果

## 6.1　2 人用

図 4 に 2 人用の QCEngin の Circle notation を，図 5 に 2 人用の IBMQ で作成したグラフを示す．

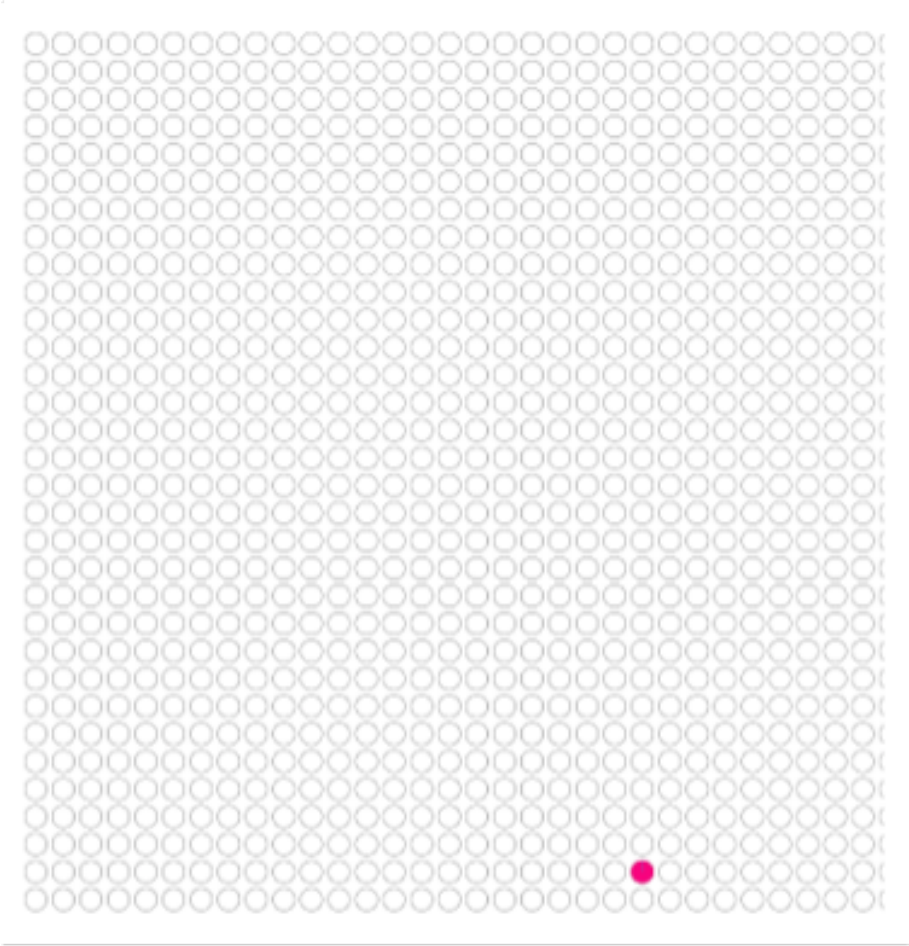

**図 4:** 2 人用 QCEngin Circle notation

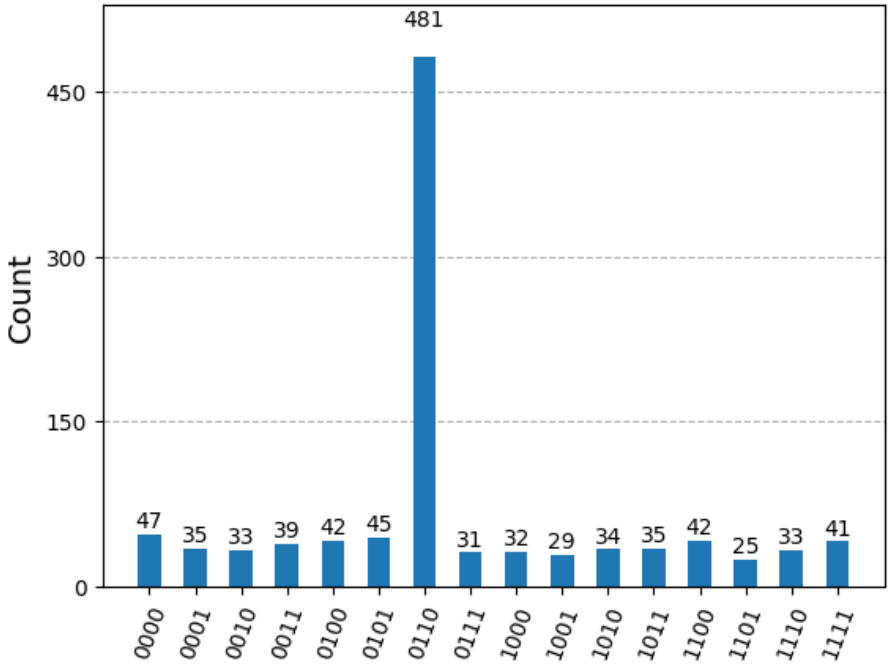


**図 5:** 2 人用 IBMQ 結果

## 6.2 3 人用

図 6 に 3 人用の QCEngin の Circle notation を，図 7 に 3 人用の IBMQ で作成したグラフを示す.

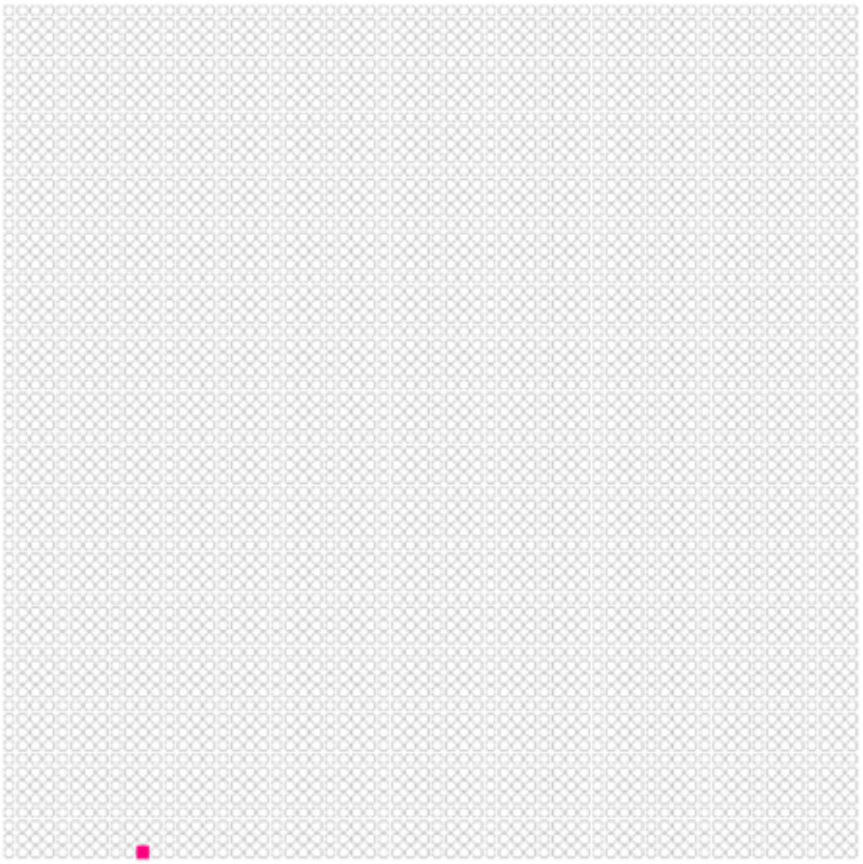

**図 6:** 3 人用 QCEngin Circle notation

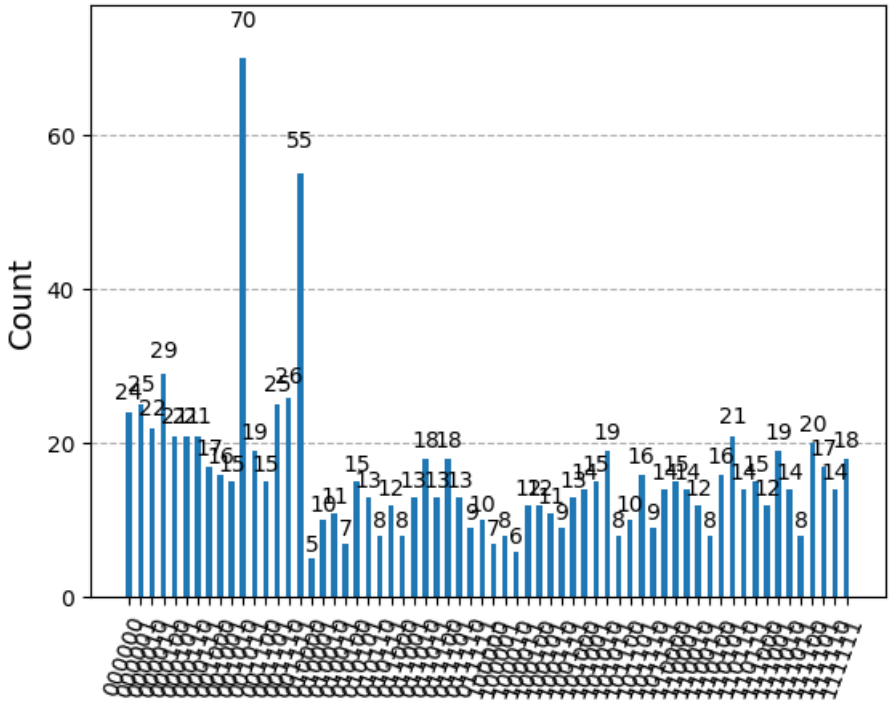


**図 7:** 3 人用 IBMQ 結果

## 6.3 4 人用

図 8 に 4 人用の QCEngin の Circle notation を，図 9 に 4 人用の IBMQ で作成したグラフを示す.

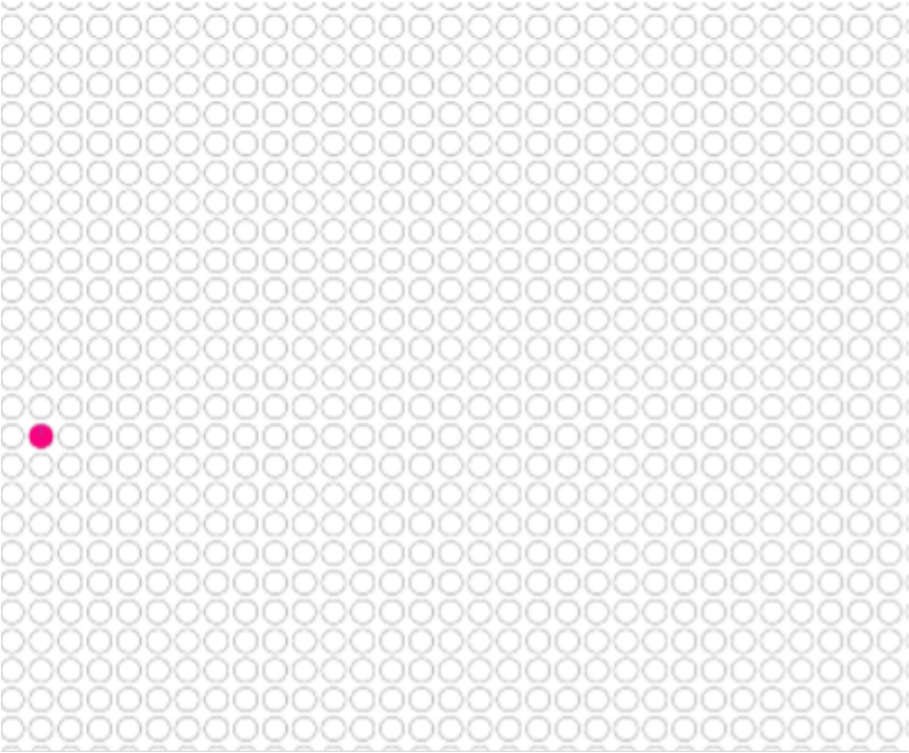

**図 8:** 4 人用 QCEngin Circle notation

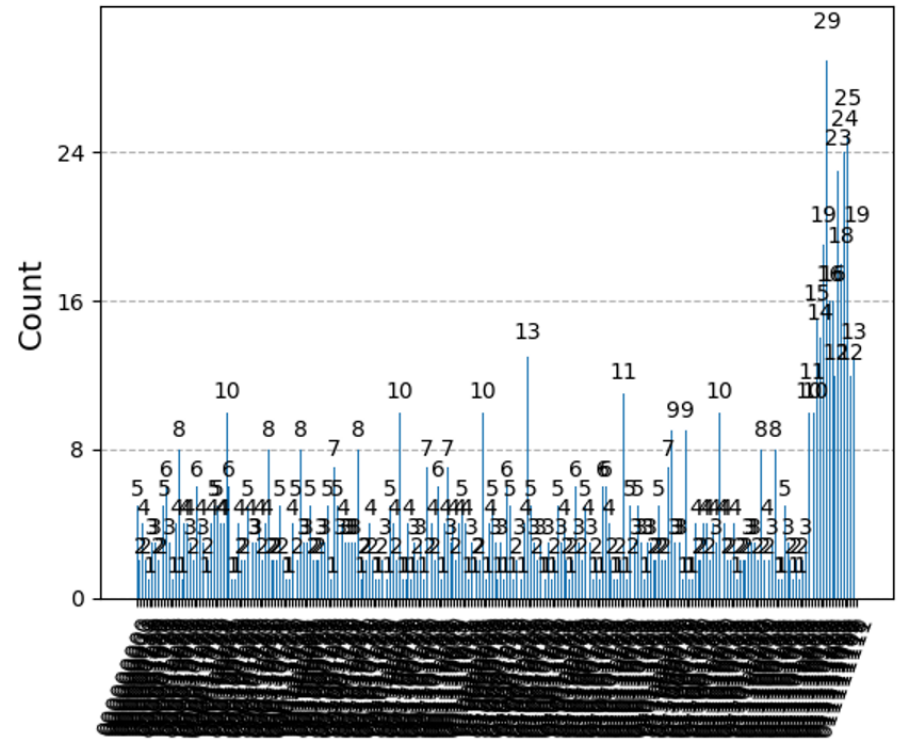


**図 9:** 4 人用 IBMQ 結果

## 7　考察

2 人用 3 人用 4 人用のすべてで circle notation は最終的に 1 つのみ量子ビットが 1 になった．これはただ一つの解を持つことにほかならない．私の班は全ての論理パズルで解が 1 つは 1 つなので，論理パズルの解と circle notation を照らし合わせると結果は正しいと言える．

また，IBMQ シミュレーターの結果について考察する．2 人用のパズルでは，図 5 に示す通り，0110 だけ抜き出ていて，論理パズルの正解の組み合わせと合致するので精度の高い結果になっていると言える．

3 人用のパズルでは，図 7 に示す通り，正解の値とは別にもう一つ値が抜き出てしまっている．そのため，論理パズルの内容を正しく論理式に反映できなかった可能性やそもそも論理パズルに複数の解がある可能性，論理式は正しいがそれを IBMQ のソースコードにする過程で間違ってしまった可能性が考えられる．図 6 に示す通り，circle notation では解が 1 つに求まっているので，IBMQ のソースコードにする過程で間違ってしまった可能性が高いと考える．この場合，比較的容易に改善できると考える．

4 人用のパズルでは，図 9 に示す通り，おおよその解の方向性はわかるが，ここから 1 つや 2 つに解を絞ることは難しい事がわかる．考えられる可能性は 3 人用のときと同様であるが，図 8 の結果も踏まえると，3 人用のときに比べパズルやその論理式が複雑であるため，IBMQ シミュレーターに起こす過程でミスがあったのではないかと考える．また，今回は IBMQ シミュレーターのサンプル数が選択肢に対して多いわけではないため，その分シミュレーション結果が分散してしまった可能性も考えられる．この場合，サンプル数を増やせば解決するのではないかと考える．

## 8　感想

レポートを書く際に結果を使ったりするのならば，ソースコードを保存した方が良いなど実験中に言ってくれると嬉しかった．

それはそれとして，実験内容自体は面白かった．また，高専でやった論理回路の良い復習になったと思う．