6J-01

高水準言語を用いた量子回路生成手法の提案

小松　拓夢　　Takumu Komatsu
金沢大学大学院　集積回路工学研究室

1. はじめに

現在，量子コンピュータに関する様々な研究開発が行われている．今までの汎用コンピュータと同じように，量子コンピュータでも動かすためのアルゴリズムを考え，設計する必要がある．また，量子コンピュータにおいてスケーラブルな量子計算を行うにあたり，新たな設計フローを考える必要がある[1].

現在，IBM Quantum Experience[2](以下 IBM Q と表記)と呼ばれる，量子コンピュータを実際にクラウド上で動作させることが出来るようなサービスが存在している．アルゴリズムの組み方としては，量子回路と呼ばれる 1 量子ビット毎の操作を表すものに直接量子ゲートを置く方式と，または QASM と呼ばれる言語で記述する 2 通りの組み方がある．配置や使用できるゲートには制約が発生するため，設計する際には考慮する必要が出てくる．

その手間を出来るだけ省き，また，可読性の向上も目指し，さらに，将来的には簡単に量子回路を生成出来るようにするために，ここでは C 言語による QASM で記述された回路生成を提案する．

2. 量子回路

現在開発されている量子コンピュータにはいくつかの方式が存在するが，ここでは量子ゲート方式を元に話を進める．量子回路の例を図 1 に示す．

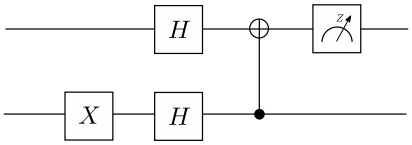

図 1　量子回路の例

それぞれの線が 1 量子ビットを表しており，ユニタリ行列で表される量子ゲートと呼ばれる論理ゲートを各量子ビットに配置している．

量子コンピュータでは，量子ビットと呼ばれる重ね合わせを表現したビットで表される．量子状態として 0 と 1 の重ね合わせ状態を表すのに，ブラケット記法を用いて式 1 のように表す．

$$|x_n\rangle = \alpha|0\rangle + \beta|1\rangle \quad \cdots(1)$$
$$(-1 \leq |\alpha| \leq 1, -1 \leq |\beta| \leq 1, \alpha, \beta \in \mathrm{C})$$

量子ゲートは，量子ビットに作用させる演算子である．例えるなら，汎用コンピュータにおける回路素子にあたるものである．

3. QASM

QASM は，量子回路のアセンブリ言語である．前述したように，IBM Q はこの言語を読む事で回路を構成し実行する事が出来る．

ソースコード 1　QASM の一例

```
x q[1];
h q[0];
h q[1];
cx q[1],q[0];
measure q[0] -> c[0];
```

C 言語で IBM Q にて QASM を構成するために，ライブラリを作成した．IBM Q で読むことが可能となるように，特に制御 NOT の制約内で収められるように QASM を構成する．例えば，標的ビット q[4]，制御ビット q[0]の制御 NOT は直接配置することが出来ない．この場合，配置可能な制御 NOT，アダマール変換と量子ビット交換回路を組み合わせて配置することにより，同様の演算が可能なように設計する．量子ビット交換回路は，アダマール変換と制御 NOT の組み合わせで実現することが出来る．このような配置制約がある中でうまく配置が可能なように設計出来るようにするため，自動で調整を行うようにする．

4. 結果の取り込み

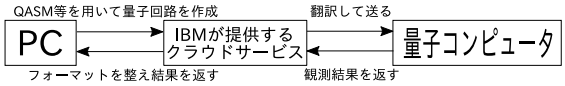

図 2　計算結果が返されるまでの流れ

IBM Q では，演算・観測結果は csv 形式で出力される．この中身は，それぞれの値が観測された個数である(つまり，観測確率が導き出せる)．観測結果を利用するためには，この結果を取り込む必要がある．そこで，出力されたファイルを読み込むための関数を用意した．それを利用することにより，汎用コンピュータ上で演算結果を用いて演算を行う事が可能になる．

5. 実装を行った量子ゲート

IBM Q であらかじめ定義されている量子ゲートの他に，以下の量子ゲートを，1 命令で実装できるようにライブラリとして組み込んだ．

・量子交換ゲート(スワップゲート)

量子ビットを交換するゲートである．

・トフォリゲート

制御制御 NOT ゲートとも言う．標的ビットが 2 ビットになった制御 NOT であり，制御 NOT と同じく可逆ゲートである．

・量子フーリエ変換

量子ビットに対してフーリエ変換を行う．式で表すと以下のようになる．

$$\mathrm{QFT}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi kxi/2^n}|k\rangle \quad \cdots(2)$$

量子フーリエ変換の回路は以下の図のように表される．

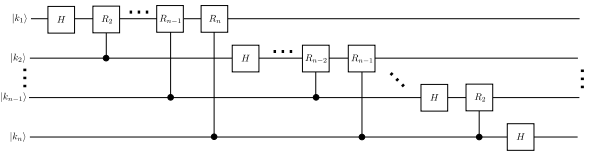

図 3　量子フーリエ変換の量子回路

・制御 Y ゲート，制御 Z ゲート

制御ユニタリゲートの 1 つであり，制御ビットが 1 ならば標的ビットにそれぞれ量子ゲート Y, Z を作用させるゲートである．

6. 実装例

このライブラリを用いて，例として Deutsch-Jozsa のアルゴリズム実装した．

Deutsch-Jozsa のアルゴリズムとは，関数値 f(x)がxに依存するかどうかを判定するアルゴリズムである[3]．

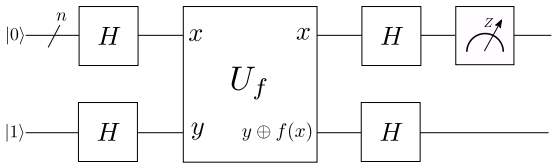

図 4　Deutsch-Jozsa のアルゴリズムの量子回路(U_fは関数によって異なる)

例えば，$f(0)=0, f(1)=1, f(2)=0, f(3)=1$ の値をとる時の場合，プログラムで書くと以下のようになる．

ソースコード 2　Deutsch-Jozsa アルゴリズムの回路生成の一例

```
int main(){
  int i;
  //StartHead();
  PauliX(2);
  for(i=0;i<3;i++){
    Hadamard(i);
  }
  CNot(0,2);
  for(i=0;i<3;i++){
    Hadamard(i);
  }
  Measure(0);
  Measure(1);
  QASM_Output("1.qasm");
  return 0;
}
```

ここから QASM を出力し，IBM Q でシミュレーションと実際に動作させた結果はそれぞれ表 1, 表 2 のようになった．

表 1：Deutsch-Jozsa のアルゴリズムのシミュレーション結果

観測値	00	01	10	11
観測確率	0.000	1.000	0.000	0.000

表 2：Deutsch-Jozsa のアルゴリズムの実際の動作結果

観測値	00	01	10	11
観測確率	0.162	0.822	0.005	0.011

7. 結論と今後

今回は，C 言語を用いて量子コンピュータの回路を比較的簡単に実装できるようにライブラリを組み，いくつかの回路を試作した．Deutsch-Jozsa のアルゴリズムや，Grover のアルゴリズムの回路の構成を行った．しかし，さらに簡潔に構成できるような余地はあると思われる．例えば，現状では回路記述をそのまま書いている部分が多いため，抽象化という点では弱いことが挙げられる．

今後は，より高次的に記述でき，よりスケーラブルな設計が出来るようなライブラリの設計が必要である．

参考文献

[1] Thomas Haner, Damian S. Steiger, Krysta Svore, and Matthias Troyer “A Software Methodology for Compiling Quantum Programs” , arXiv preprint arXiv:1614.01401(2016)

[2]IBM Quantum Experience (https://quantumexperience.ng.bluemix.net/qx/experience) 2017 年 12 月 27 日閲覧

[3]D.Deutsch, R.Jozsa ”Rapid solutions of problems by quantum computation” Proceedings if the royal society A, pp.553-558(1992)