
3時間で学ぶ
Shorのアルゴリズム入門

はじめに

- 6/3のマルレク「暗号技術の現在」では、暗号技術が現在の「公開キー暗号/RSA暗号」から「ポスト量子暗号」に大きく変わろうとしているという話をした。こうした変化を引き起こした最大の原因は、25年前に発見された「Shorの素因数分解アルゴリズム」である。
- RSA暗号は、大きな素数p,q 二つの積である大きな数Nが、たとえNを知っていても、現在のコンピュータでは、その素因数p,qを求めることがとても難しいという事実をその基礎にしている。公開キー暗号は、コンピュータでも分解できないこのNを、事実上、皆の前に公開するという暗号方式である。Shorは「量子コンピュータを使えば」、Nの素因数分解が極めて高速に可能になることを発見した。それは、「公開キー暗号/RSA暗号」が、簡単に破られるということを意味している。

はじめに

- ではなぜ、こうした発見が25年間も「暗号技術に対する脅威」とは見なさなかったのだろうか？　その理由は簡単なものだ。それは、「量子コンピュータ」が、すぐにも実現する技術とは見なされなかったからである。現時点でも、大きなNに対して、Shorのアルゴリズムでその素因数を求められる量子コンピュータは存在しない。
- ただ、20年後40年後は、どうなっているだろうか？
- 近年になって、量子コンピュータの「実現可能性」について、大きな認識の変化がある。基本的には、いままでより、かつてなく多くの人が「いつか、確実な時期はわからないが、量子コンピュータは実現するだろう」と考えるようになってきた。Shorのアルゴリズムに対する関心が、新たに高まっているのは、そうした背景がある。NSAやNISTが、「ポスト量子暗号」への動きを本格化しているのは、当然のことだと思う。

はじめに

- 小論は、先のセミナーでは十分に取り上げることのできなかったShorのアルゴリズムの概要をまとめたものである。
- 残念ながら、講演時間の制限もあって、重要ないくつかの論点が割愛されている。このセミナーを機会に、さらに量子アルゴリズムへの関心が高まることを期待している。

Agenda

Part I 量子計算の基礎

- 古典bitと量子bit
- 量子ゲート
- テンソル積
- 量子回路を構成する
- 任意のf(x)を計算する量子回路を考える

Part II Quantum Parallelism

- 量子コンピュータではなぜ高速な計算ができるのか？
- 量子コンピュータの出力は、どう取り出せるのか？

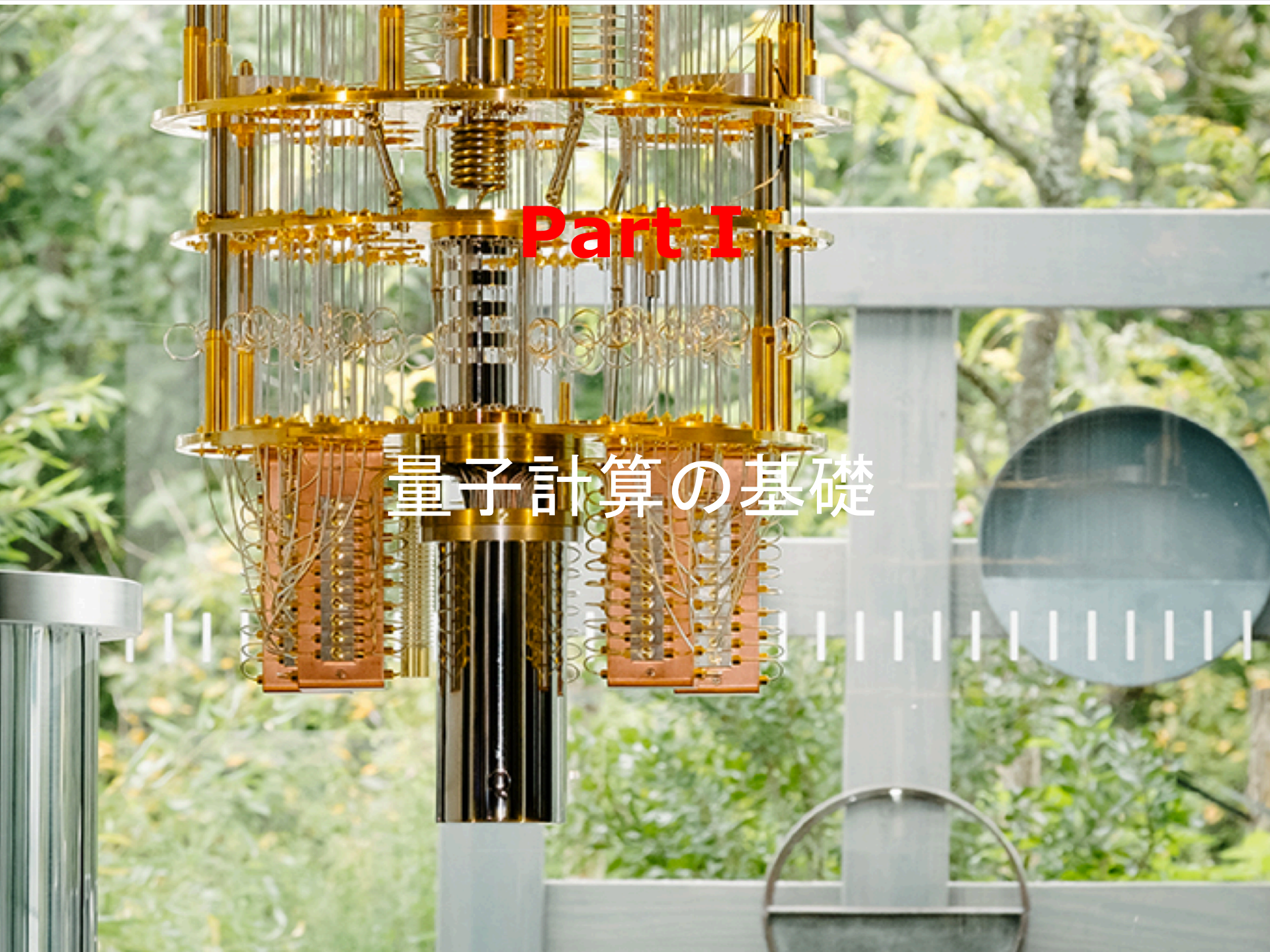
Agenda

Part III　Simonのアルゴリズム

- Simonの問題
- Simonの問題を解く回路　n=3の場合の実行例
- 一般的な解法

Part IV　Shorのアルゴリズム

- Shorの素因数分解アルゴリズム
  関数の「周期」から、素因数を求める
- どのように関数の「周期」を求めるのか？　概略編
- Phase Estimation
- Phase Estimationを利用して、「周期」を発見する


Part I
量子計算の基礎

古典bitと量子bit

一次元の点としての古典bit
二次元のベクトルとしての量子bit

- 古典bitは、0 または 1 の値しかとらない二つの離散的な点として表現される。
- 量子bit qubitは、|0>と|1>という二つのベクトルによって張られ、aとbという二つの数によって決定される二次元のベクトルとして表現される。この状態を「重ね合わせ」という。

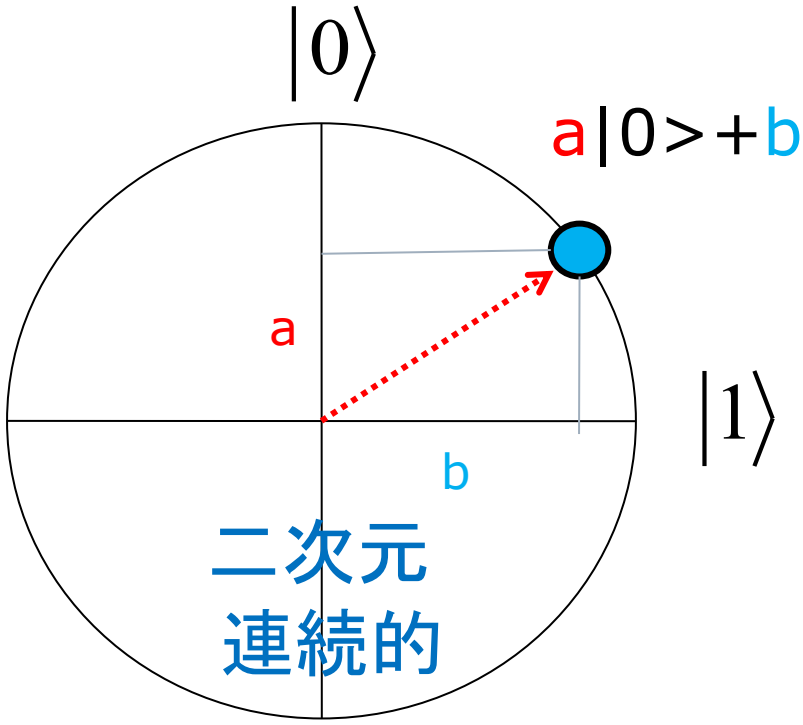
$$|Qubit> = a|0> + b|1>$$

この時、$|a|^2+|b|^2=1$ という条件がつく。
a, b は、複素数の値を取る。

- qubitは、二次元の複素ベクトルとして表現される。

二つのビットは、異なる性質を持つ

qubit



qubitは、|0>と|1>の「重ね合わせ」で、aとbの二つの数で決まる

$$\begin{pmatrix} a \\ b \end{pmatrix}$$

|0>の成分

|1>の成分

という二次元ベクトルである。

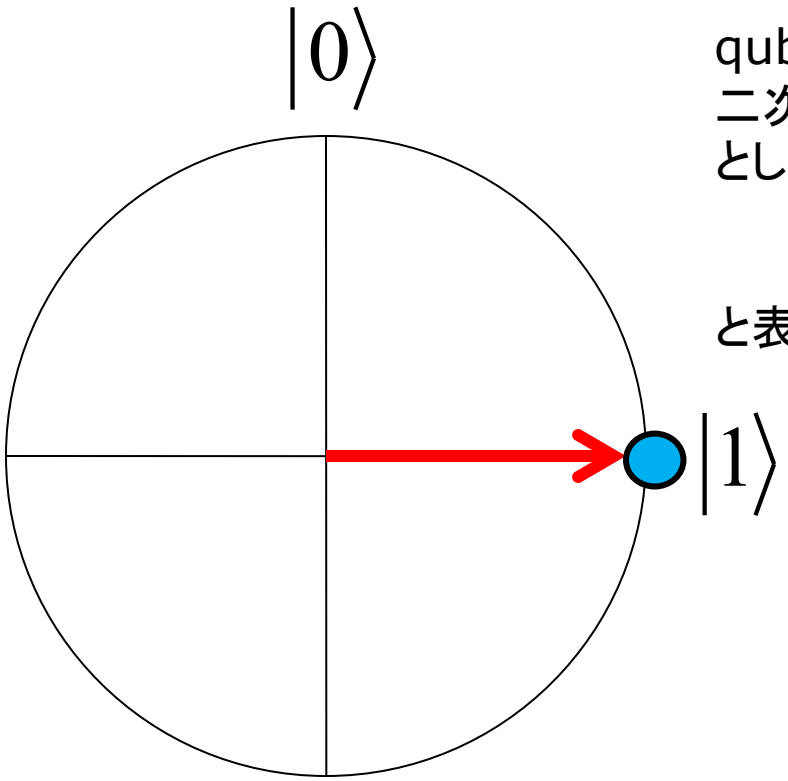
$|a|^2+|b|^2=1$

である。

bit

一次元
離散的

|Qubit> = a|0>+b|1>
で、a=0, b=1の時の状態

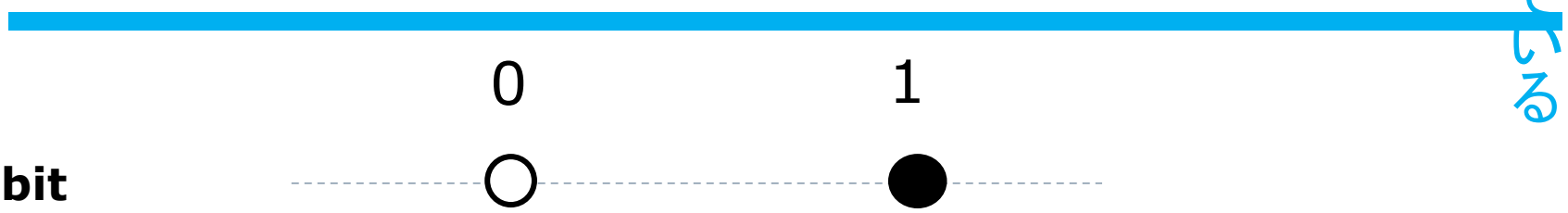


qubit |1>は、
二次元のベクトル
として、

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

と表される

qubit



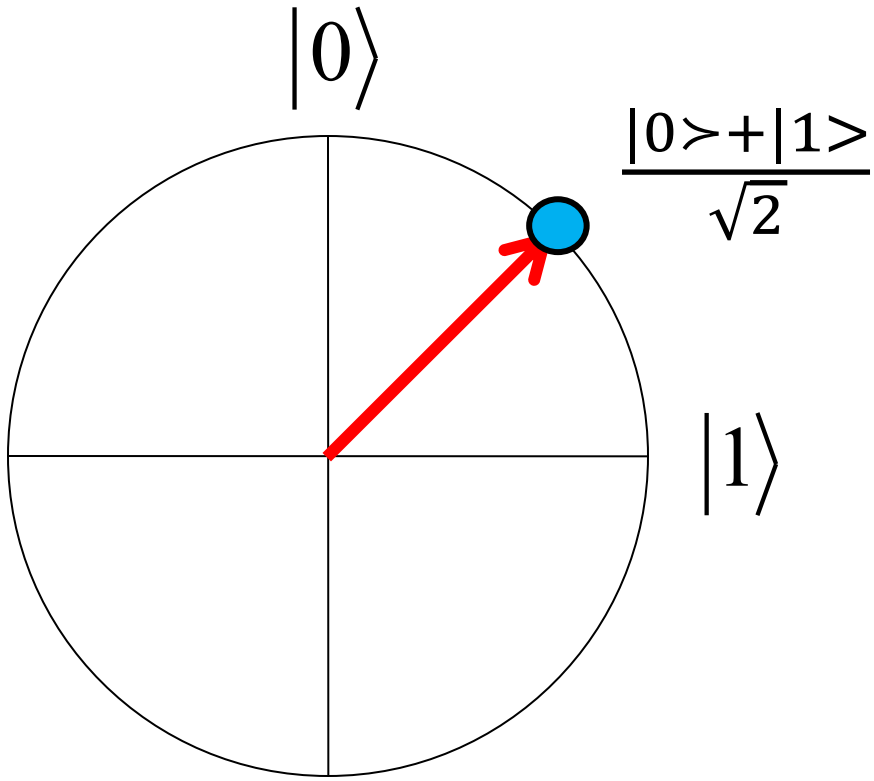
qubit $|1\rangle$ と bit 1 は、対応している

|Qubit> = a|0>+b|1>

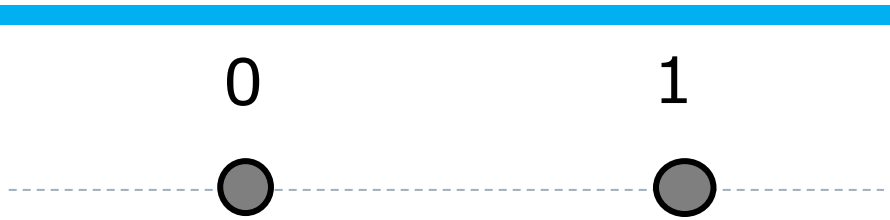
で、$a=\frac{1}{\sqrt{2}}$, $b=\frac{1}{\sqrt{2}}$ の時の状態

qubit と bit は、対応しない

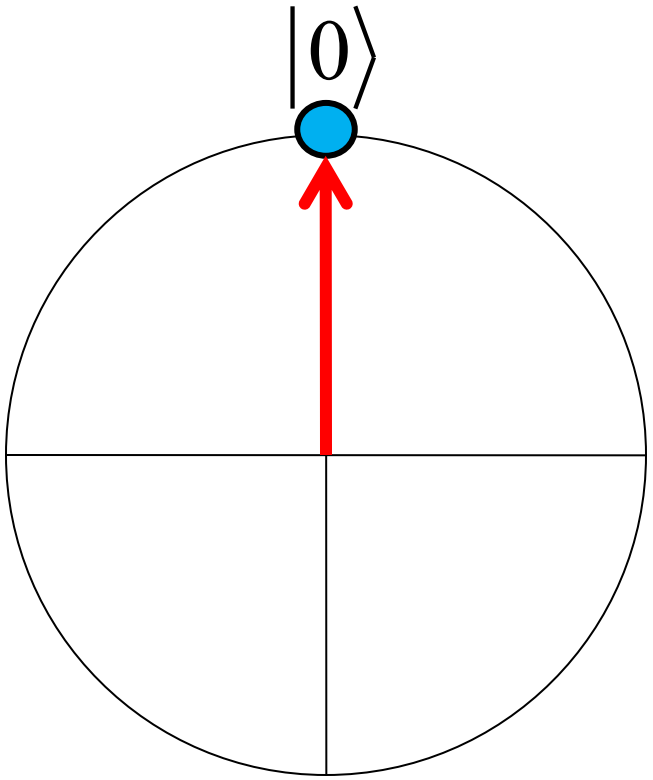
qubit



bit



|Qubit> = a|0>+b|1>
で、a=1, b= 0の時の状態



qubit |0>は、
二次元のベクトル
として、

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

と表される

qubit

qubit $|0\rangle$ と bit 0は、対応している

0 1

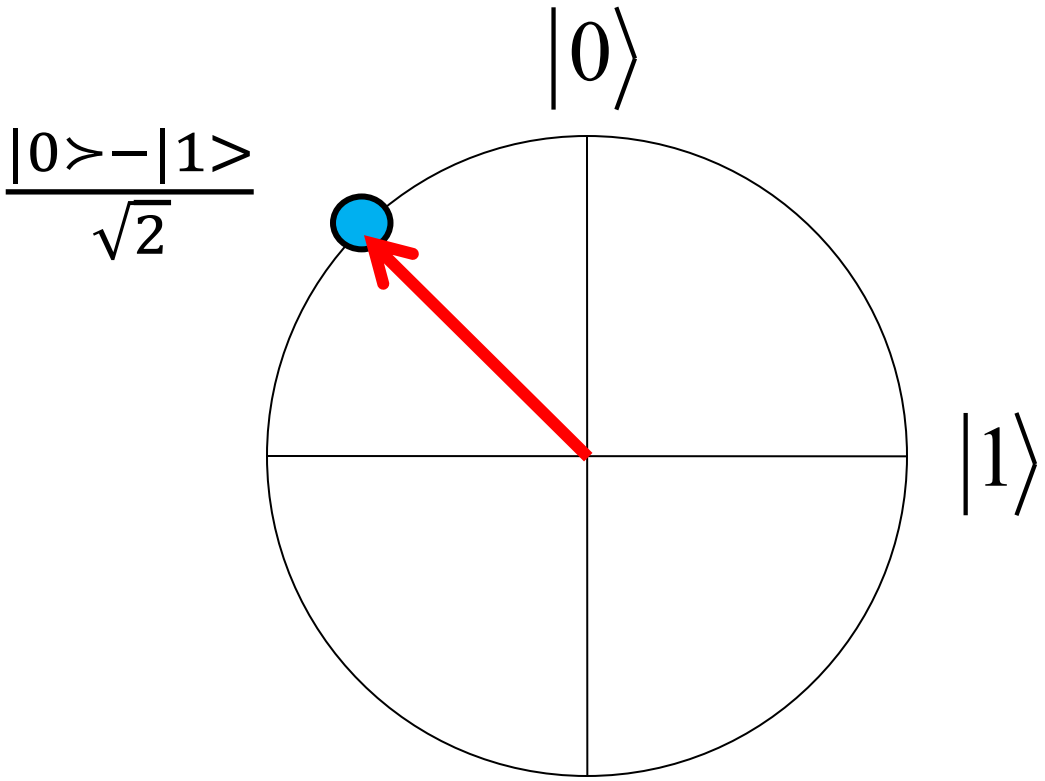
bit

|Qubit> = a|0>+b|1>

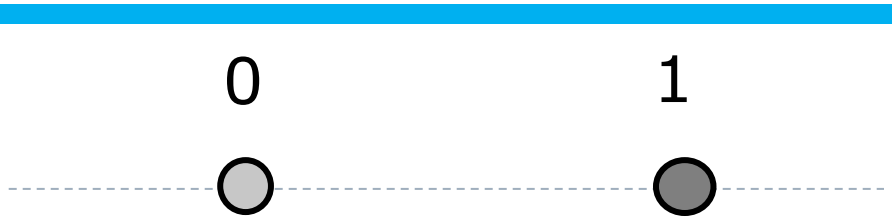
で、$a=\frac{1}{\sqrt{2}}$, $b=-\frac{1}{\sqrt{2}}$ の時の状態

qubit と bit は、対応しない

qubit



bit



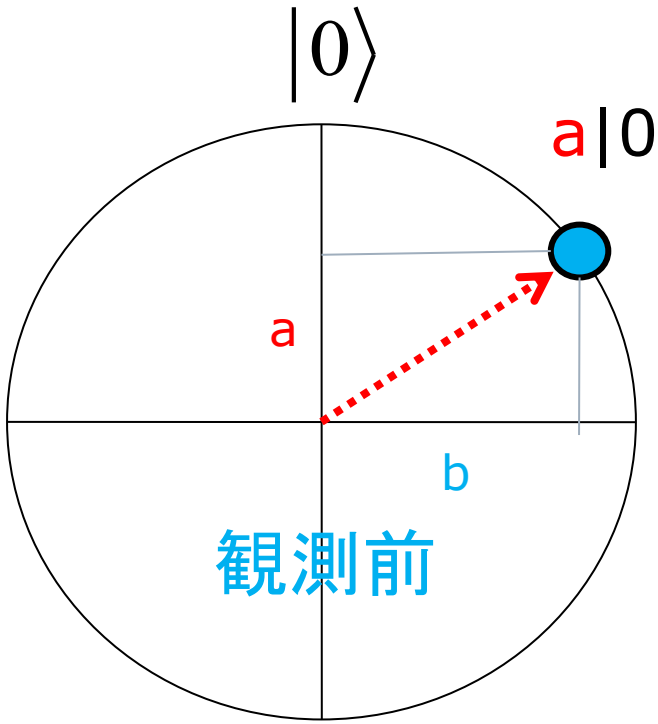
qubitの観測

- |Qubit> = a|0>+b|1> を観測すると、重ね合わせの状態は破れて失われ、|0>または|1>のいずれかの状態になる。

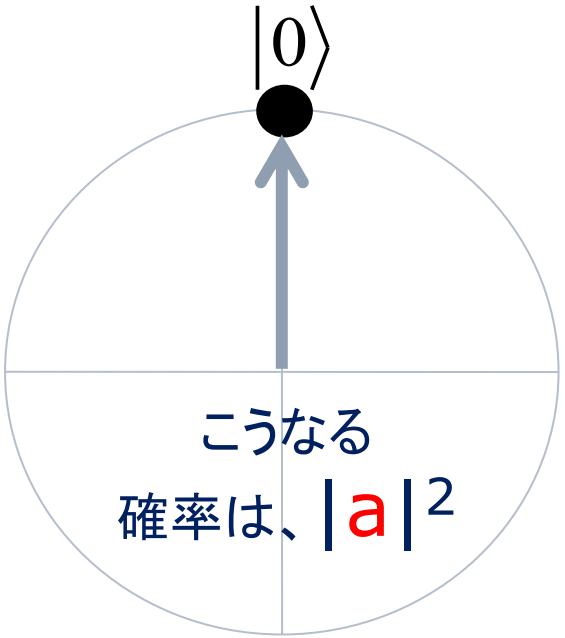
- この時、
  |0>を観測する確率は、$|a|^2$で与えられ、
  |1>を観測する確率は、$|b|^2$で与えられる。

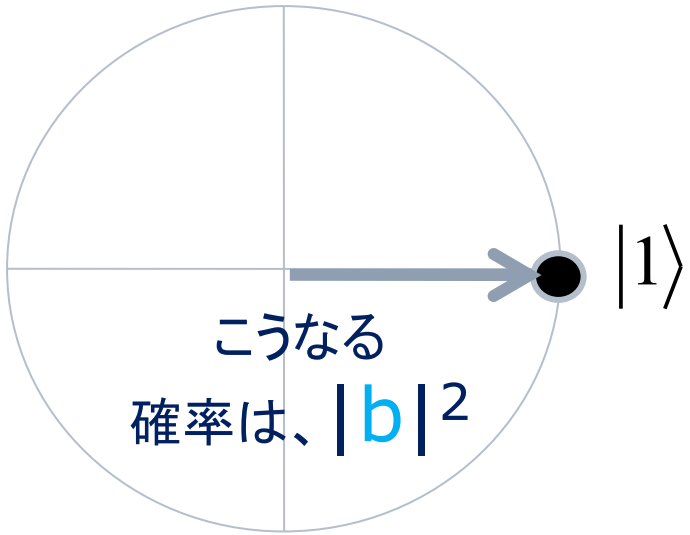
qubitの観測

観測によって、
「重ね合わせの状態」は
失われる



観測後





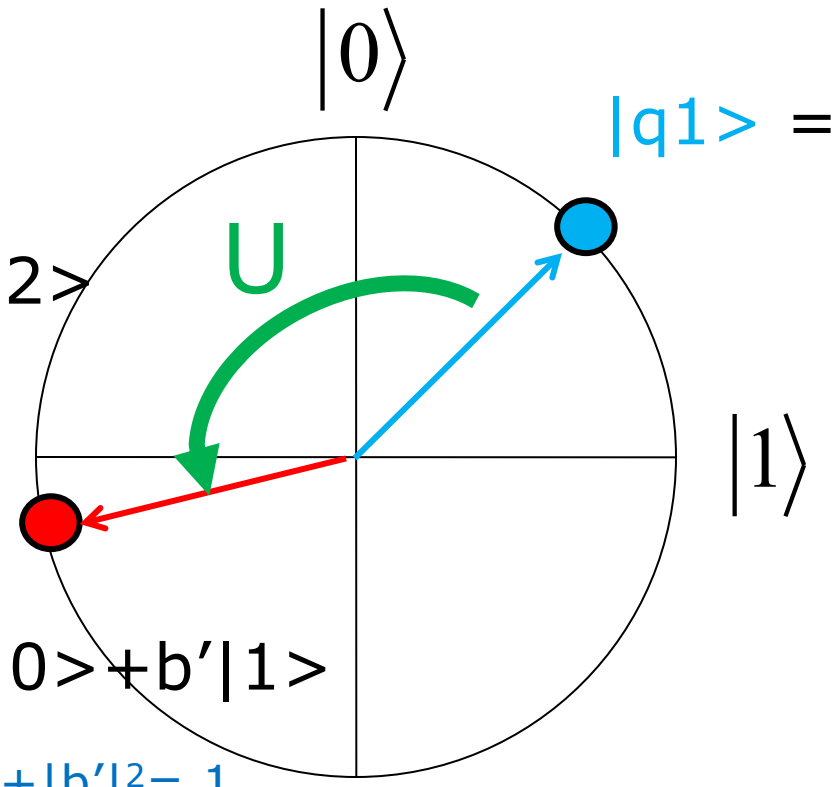
qubitの状態変化　ユニタリ変換

- qubitの状態 |q1> ＝ a|0>+b|1>が、
  別の状態 |q2> ＝ a′|0>+b′|1> に変化したとしよう
- それぞれのqubitの成分 a, b, a′, b′は、次の条件を満たす
  $|a|^2+|b|^2 = 1$
  $|a'|^2+|b'|^2 = 1$
- これは、すべてのqubitが、先の図では、原点から長さ 1の円周上の一点として表現できることを表している。（いくつか簡略化しているのだが）
- ベクトルの長さを変えない回転の変換を「ユニタリ変換」という。
  qubit |q1>から qubit |q2>への状態変化は、ユニタリ演算子 Uを用いて **U|q1> = |q2>** と表すことができる。
- Uの同じ角度での逆方向への回転を$U^{\dagger}$とすれば、
  $UU^{\dagger} = I$ (単位行列) となる。
  （回転して、今度は逆方向に回転すれば、元の位置に戻る）

|q1> = a|0>+b|1> から
|q2> = a’|0>+b’|1> への
状態変化

ユニタリ変換

ユニタリ変換Uは、
長さを変えない
回転である



|q1> = a|0>+b|1>

$|a|^2+|b|^2 = 1$

U|q1> = |q2>

|q2> = a’|0>+b’|1>

$|a'|^2+|b'|^2 = 1$

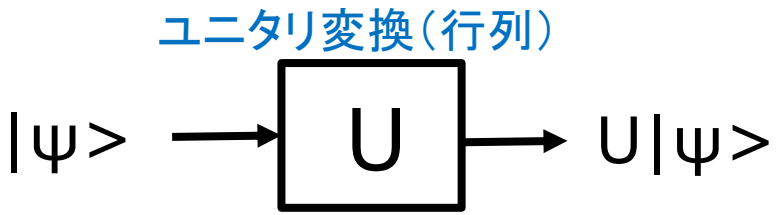
量子ゲート

量子ゲートはどのような働きをするのか？

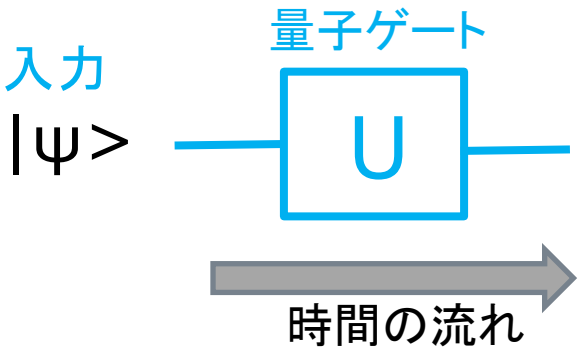
- 量子ゲートは、入力のqubitたちを出力のqubitたちに変換する。それは、古典的なゲートが、古典ビットの入力を古典ビットの出力に変換するのと同じである。
- 古典的なゲートは、基本的には、AND, OR, NOTといった「論理的」な演算で定義されるが、量子的なゲートは数学的には、「ユニタリ変換」として定義される。「ユニタリ変換」は、「ユニタリ行列」で定義される。
- 「ユニタリ行列」は、入力のqubitの「入力ベクトル」を出力のqubitの「出力ベクトル」に変換する。
- 一つの量子ゲートには、一つの「ユニタリ行列」が対応する。
- 古典的なゲートとは異なって、量子ゲートでは入力のqubitの数と出力のqubitの数は等しい。

量子ゲートとユニタリ行列

- 量子の状態 |ψ>は、ユニタリ変換U（ユニタリ行列）の作用を受けて、状態 U|ψ> に変化する。
- この変化 |ψ> → U|ψ>を、次のように表そう。

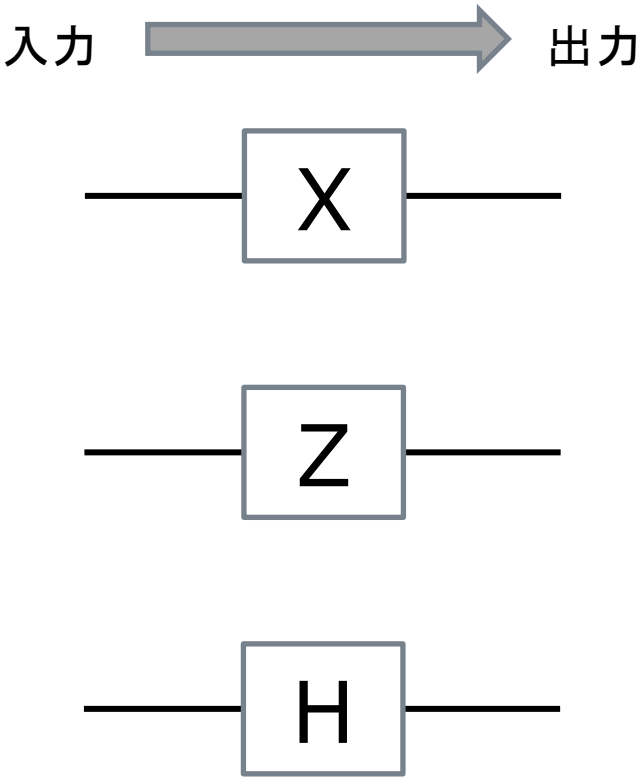


- この時、Uを、|ψ> を入力、U|ψ> を出力とする回路と考えることができる。これを、「量子ゲート」と呼ぶ。



量子ゲートは
ユニタリ行列と
一対一に対応する

代表的な1-qubitのゲート X, Z, H



Bit Flipper

$a|0\rangle+b|1\rangle \rightarrow b|0\rangle+a|1\rangle$

Phase Flipper

$a|0\rangle+b|1\rangle \rightarrow a|0\rangle-b|1\rangle$

Hadamard

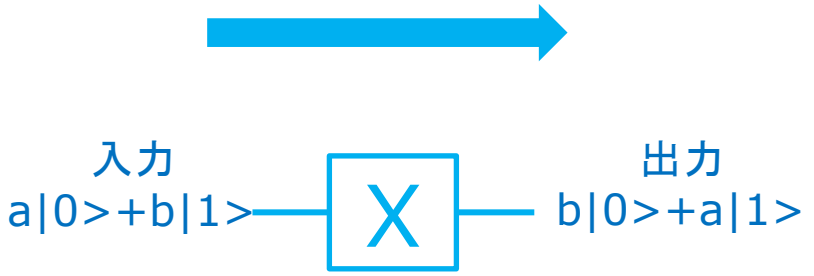
$a|0\rangle+b|1\rangle \rightarrow$

$\frac{1}{\sqrt{2}}(a+b)|0\rangle+\frac{1}{\sqrt{2}}(a-b)|1\rangle$

X, Z, H の行列と対応する量子ゲートの働き

$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ なので、

$$X\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix}$$

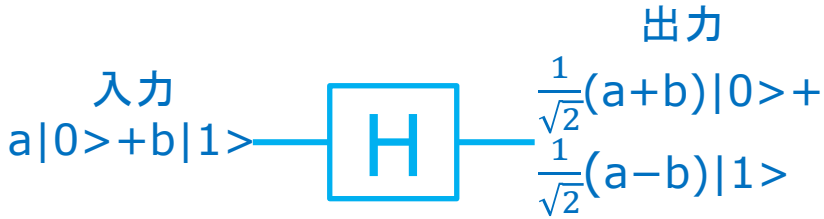


$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ なので、

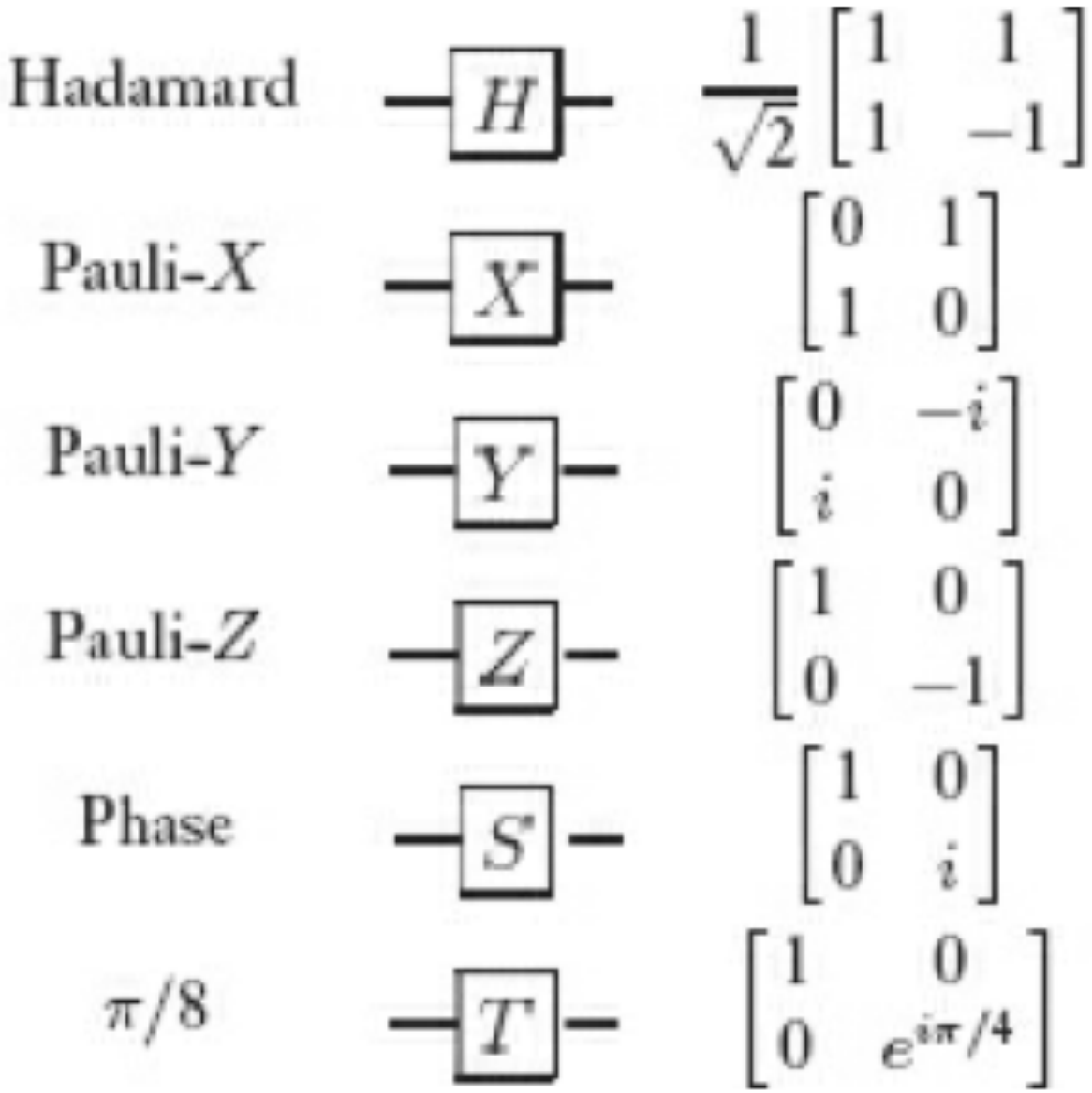
$$Z\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a \\ -b \end{pmatrix}$$

$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ なので、

$$H\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} a+b \\ a-b \end{pmatrix}$$



1-qubitの量子ゲートの例とそれに対応する行列



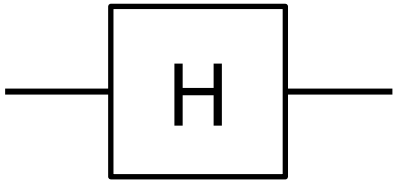
アダマール・ゲート

アダマール行列とアダマール・ゲート

$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

という行列を、アダマール（Hadamard）行列という。

この行列に対応するゲートを、アダマール・ゲートといい、次のように表す。



アダマール行列の性質

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

$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ とすると

1. $H|0\rangle = |+\rangle$
2. $H|1\rangle = |-\rangle$
3. $H|+\rangle = |0\rangle$
4. $H|-\rangle = |1\rangle$
5. $HH = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ 単位行列

|+>と|−>の観測　量子コイン

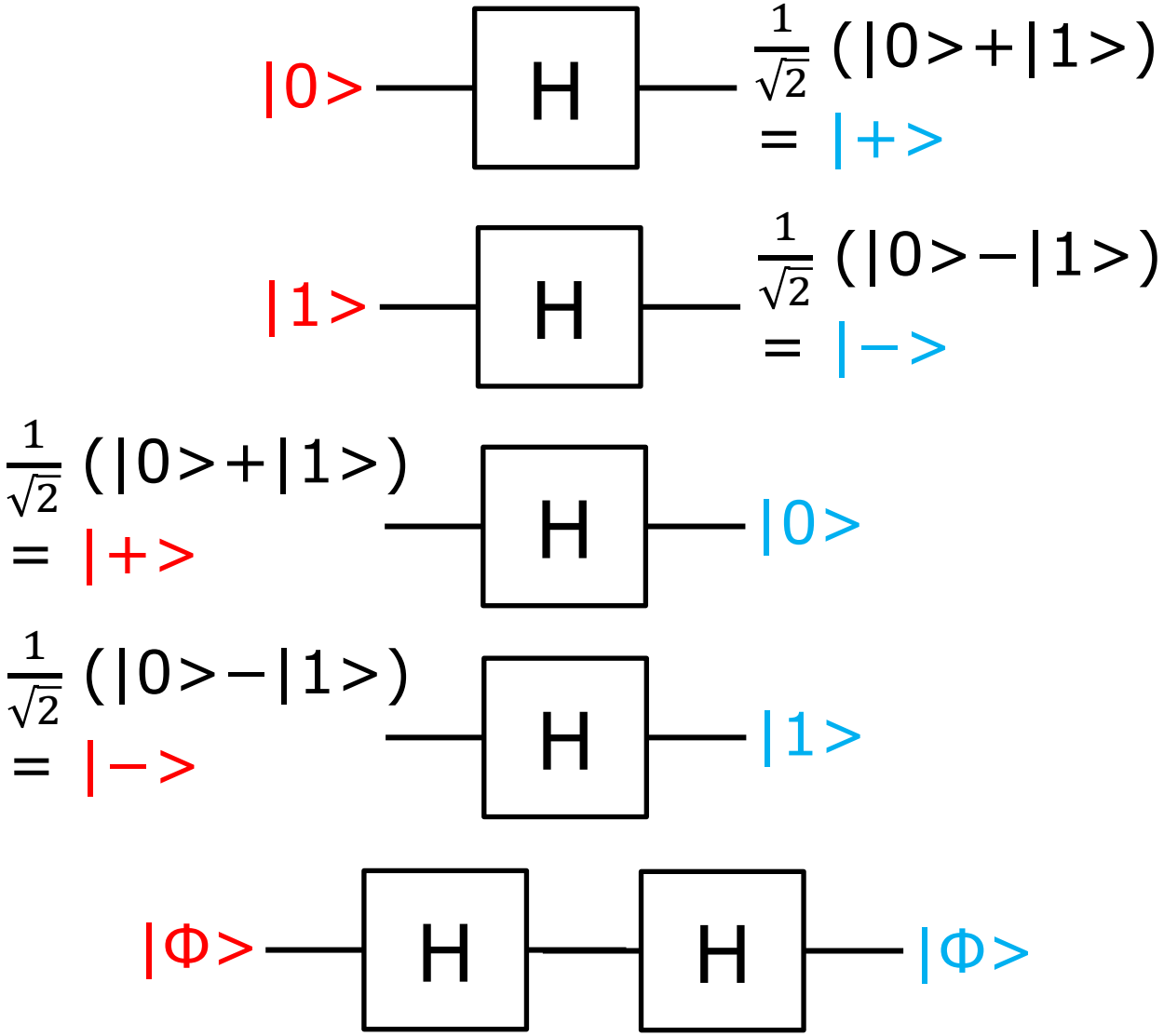
- |+>= $\frac{1}{\sqrt{2}}$ |0>+$\frac{1}{\sqrt{2}}$|1> を観測したとする。この時、
  |0>を観測する確率は、|$\frac{1}{\sqrt{2}}$ |^2=1/2
  |1>を観測する確率は、|$\frac{1}{\sqrt{2}}$ |^2=1/2
- |−>= $\frac{1}{\sqrt{2}}$ |0>−$\frac{1}{\sqrt{2}}$|1> を観測したとする。この時、
  |0>を観測する確率は、|$\frac{1}{\sqrt{2}}$ |^2=1/2
  |1>を観測する確率は、|−$\frac{1}{\sqrt{2}}$|^2=1/2
- |+>と|−>は、等しい確率 1/2で|0>と|1>を観測する。
  これは、コインを投げて裏・表が出る確率と等しい。
  |+>と|−>を、「量子コイン」と呼ぶことがある。
- |+>と|−>は、明らかに異なる状態だが、観測によっては区別することができない。

アダマール行列の性質(先の式の計算)

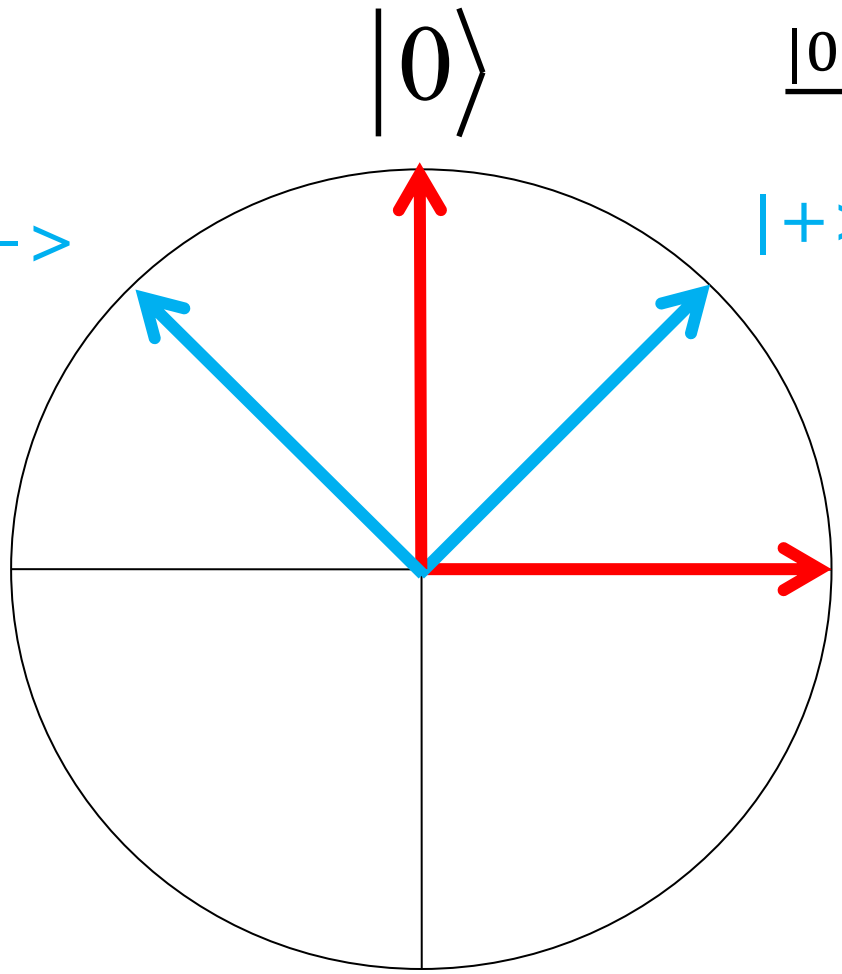
1. $H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$
2. $H|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$
3. $H|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}2\\ 0\end{pmatrix} = |0\rangle$
4. $H|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}0\\ 2\end{pmatrix} = |1\rangle$
5. $HH = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}2 & 0\\ 0 & 2\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$

単位行列

アダマール・ゲートの性質



標準基底とアダマール基底



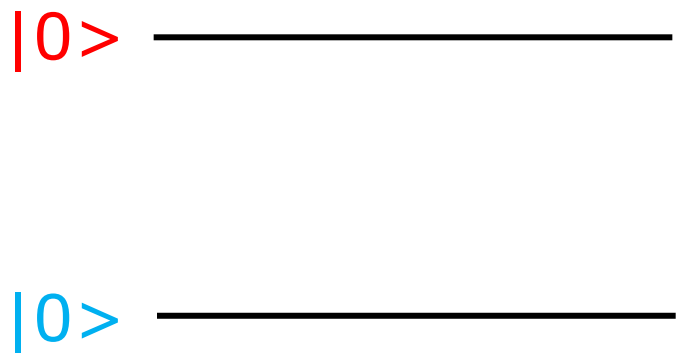
アダマール変換は、
基底 |0>, |1> で
張られた座標系を
基底 |+>, |−> で
張られた座標系に
変換する、

基底 |0>, |1> を
標準基底（計算基底）
基底 |+>, |−> を
アダマール基底
と呼ぶ

テンソル積

独立した二つのシステムを
一つのシステムとして、
その「状態」を考える

Two Qubitのテンソル積の計算例　単純な例 1



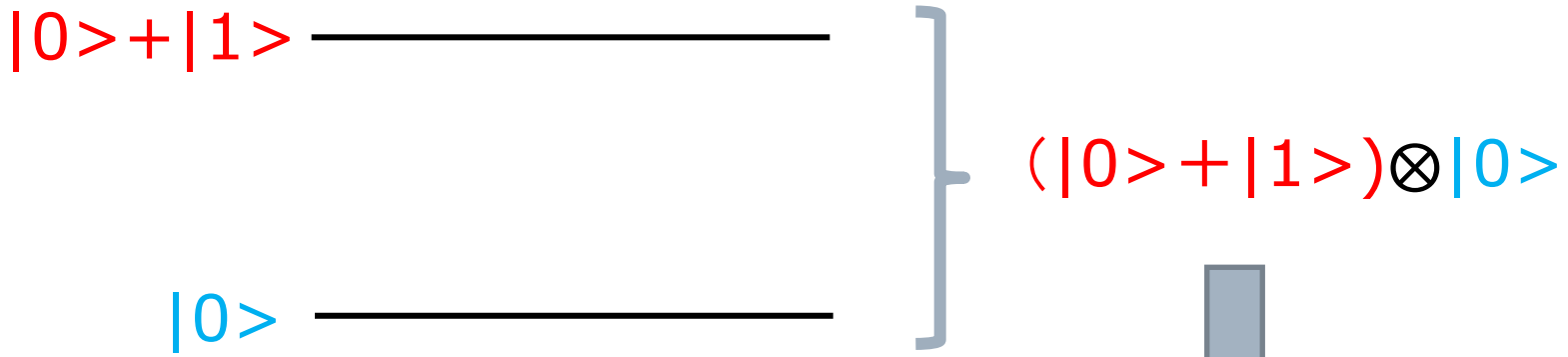
こういう表記をする

$|0\rangle \otimes |0\rangle = |00\rangle$

順序を持つ

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} \mathbf{1} \\ \mathbf{0} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{1}\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ \mathbf{0}\begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Two Qubitのテンソル積の計算例　単純な例 2

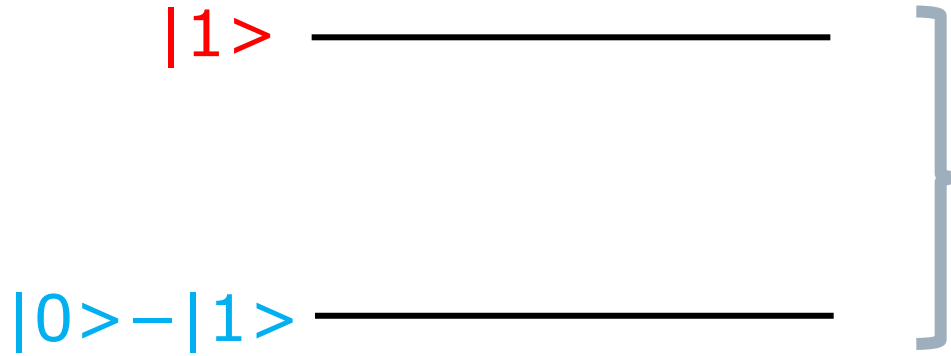


これが、系の状態

$$(|0\rangle+|1\rangle)\otimes|0\rangle$$
$$= |0\rangle\otimes|0\rangle + |1\rangle\otimes|0\rangle = |00\rangle+|10\rangle$$

|0>+|1>という形は本当はおかしい。$\alpha|0\rangle+\beta|1\rangle$の時、$|\alpha|^2+|\beta|^2=1$ という条件があるからだ。本当は、$\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle$なのだが、計算では無視している。

Two Qubitのテンソル積の計算例　単純な例 3



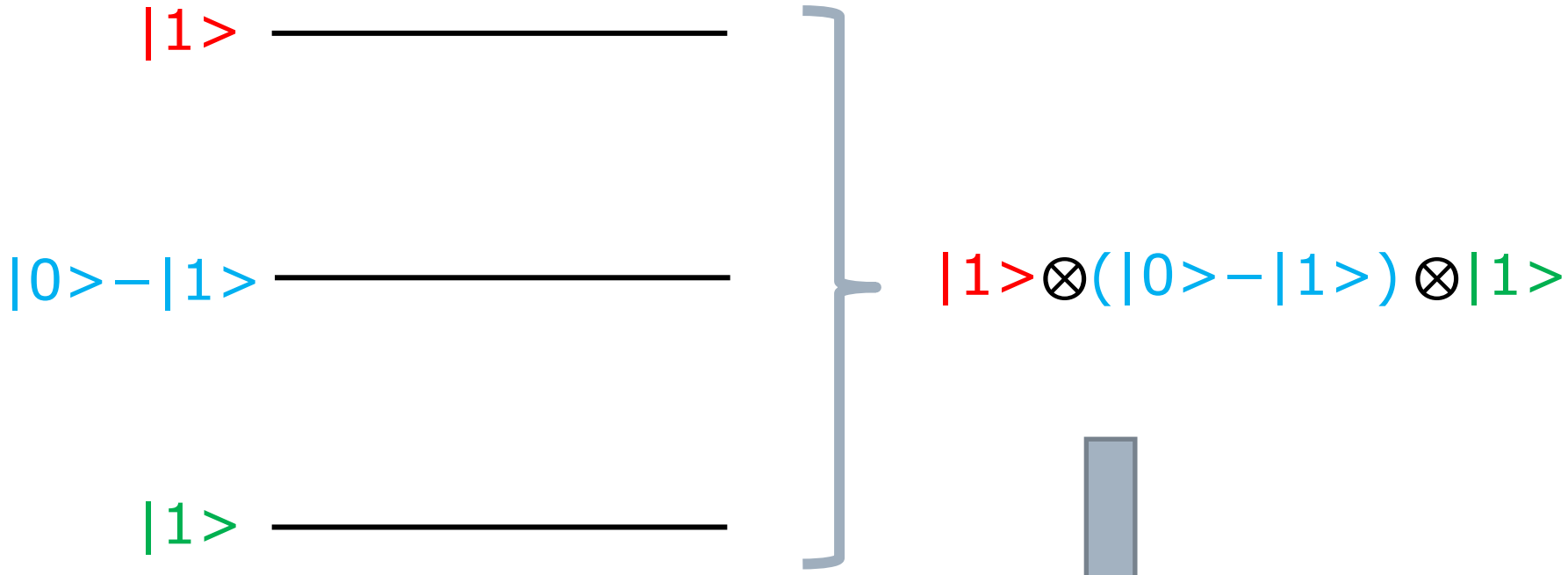
これが、系の状態

$|1\rangle \otimes (|0\rangle - |1\rangle) =$

$|1\rangle \otimes |0\rangle - |1\rangle \otimes |1\rangle = |10\rangle - |11\rangle$

$|0\rangle - |1\rangle$という形は本当はおかしい。$\alpha|0\rangle + \beta|1\rangle$の時、$|\alpha|^2 + |\beta|^2 = 1$ という条件があるからだ。本当は、$\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$なのだが、計算では無視している。

Three Qubitのテンソル積の計算例　単純な例 4



$$|1\rangle \otimes (|0\rangle - |1\rangle) \otimes |1\rangle$$
$$= (|1\rangle \otimes |0\rangle - |1\rangle \otimes |1\rangle) \otimes |1\rangle$$
$$= (|10\rangle - |11\rangle) \otimes |1\rangle = |101\rangle - |111\rangle$$

Two Qubitの計算例

- 第一のQubitが、3/5|0>+4/5|1> で、
  第二のQubitが、$1/\sqrt{2}|0>-1/\sqrt{2}|0>$としよう。

- この二つのQubitが結合したシステムの状態は、
  $(3/5|0>+4/5|1>) \otimes (1/\sqrt{2}|0>-1/\sqrt{2}|1>)$
  $= 3/5\sqrt{2}|00>-3/5\sqrt{2}|01>+4/5\sqrt{2}|10>-4/5\sqrt{2}|11>$
  のように計算される。(⊗ を、普通の掛け算のように計算した。)

- ところで、全ての Two Qubitsの状態は、二つのQubitの状態の組み合わせに、分解できるだろうか？
- それがテンソル積で分解できない状態を「エンタングルメント」という。

行列とテンソル積
例（1）

$A=\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix}$, $B=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$ としよう。

$$A\otimes B=\begin{pmatrix}\mathbf{1} & \mathbf{-1}\\ \mathbf{0} & \mathbf{2}\end{pmatrix}\otimes\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}=\begin{pmatrix}\mathbf{1}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix} & \mathbf{-1}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\\ \mathbf{0}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix} & \mathbf{2}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\end{pmatrix}$$

$$=\begin{pmatrix}1 & 2 & -1 & -2\\ 3 & 4 & -3 & -4\\ 0 & 0 & 2 & 4\\ 0 & 0 & 6 & 8\end{pmatrix}$$

行列のテンソル積の例 (2)

$A=\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix}$, $B=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$ としよう。

$$B\otimes A=\begin{pmatrix}\mathbf{1} & \mathbf{2}\\ \mathbf{3} & \mathbf{4}\end{pmatrix}\otimes\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix}=\begin{pmatrix}\mathbf{1}\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix} & \mathbf{2}\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix}\\ \mathbf{3}\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix} & \mathbf{4}\begin{pmatrix}1 & -1\\ 0 & 2\end{pmatrix}\end{pmatrix}$$

$$=\begin{pmatrix}1 & -1 & 2 & -2\\ 0 & 2 & 0 & 4\\ 3 & -3 & 4 & -4\\ 0 & 6 & 0 & 8\end{pmatrix}$$

テンソル積では、
$A\otimes B \neq B\otimes A$
である

ベクトルのテンソル積

ベクトルは、nx1の行列であるから、

$$\begin{pmatrix} \boldsymbol{a_1} \\ \boldsymbol{a_2} \end{pmatrix} \otimes \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{a_1} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \\ \boldsymbol{a_2} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ a_1 b_2 \\ a_2 b_1 \\ a_2 b_2 \end{pmatrix}$$

ベクトルのテンソル積の例（1）

$$\begin{pmatrix}\mathbf{1}\\\mathbf{2}\end{pmatrix}\otimes\begin{pmatrix}3\\4\end{pmatrix}=\begin{pmatrix}\mathbf{1}\begin{pmatrix}3\\4\end{pmatrix}\\\mathbf{2}\begin{pmatrix}3\\4\end{pmatrix}\end{pmatrix}=\begin{pmatrix}3\\4\\6\\8\end{pmatrix}$$

$$\begin{pmatrix}\mathbf{3}\\\mathbf{4}\end{pmatrix}\otimes\begin{pmatrix}1\\2\end{pmatrix}=\begin{pmatrix}\mathbf{3}\begin{pmatrix}1\\2\end{pmatrix}\\\mathbf{4}\begin{pmatrix}1\\2\end{pmatrix}\end{pmatrix}=\begin{pmatrix}3\\6\\4\\8\end{pmatrix}$$

ベクトルのテンソル積の例 (2)

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

量子回路を構成する

量子ゲートから量子回路を構成する

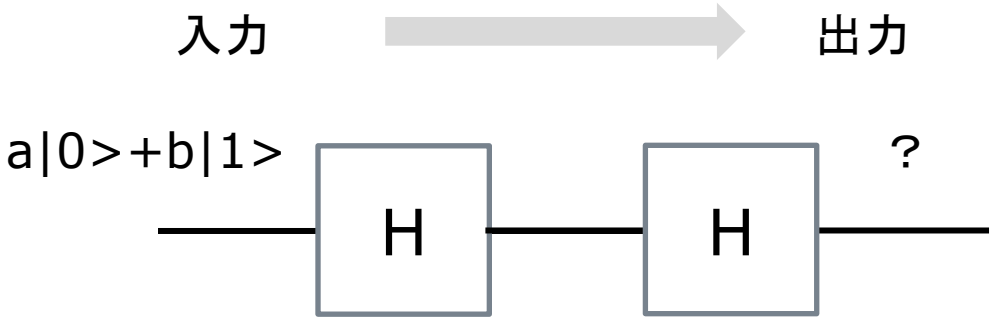
量子ゲートから回路を構成する方法は、基本的には、次の三つである。

1. Serialな構成　あるゲートの出力を次のゲートの入力にserialに接続する
2. Parallelな構成　あるゲートと別のゲートをparallelに構成する
3. コントロールの構成 あるゲートの出力を、parallelに走る別のゲートのコントロールに使用する¶

ゲートを直列に組み合わせる

ゲートを直列に組み合わせた回路の働きは、直列の回路を
構成するゲートの行列の積を計算することで、求めることができる。

二つのHゲートを直列につないだ、次のような回路を考えてみよう。

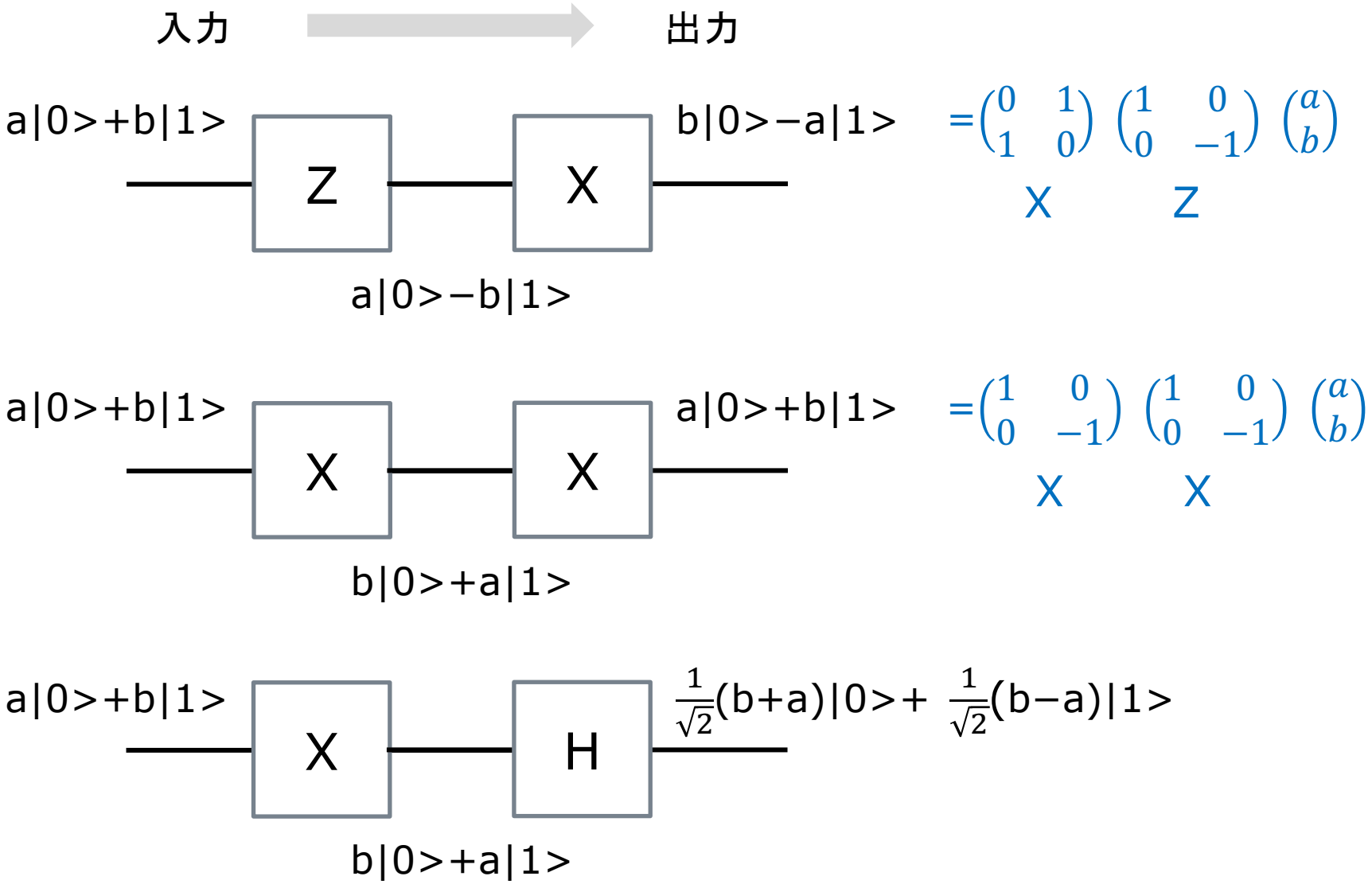


$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ なので、行列の積HHを計算する。

$$HH = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

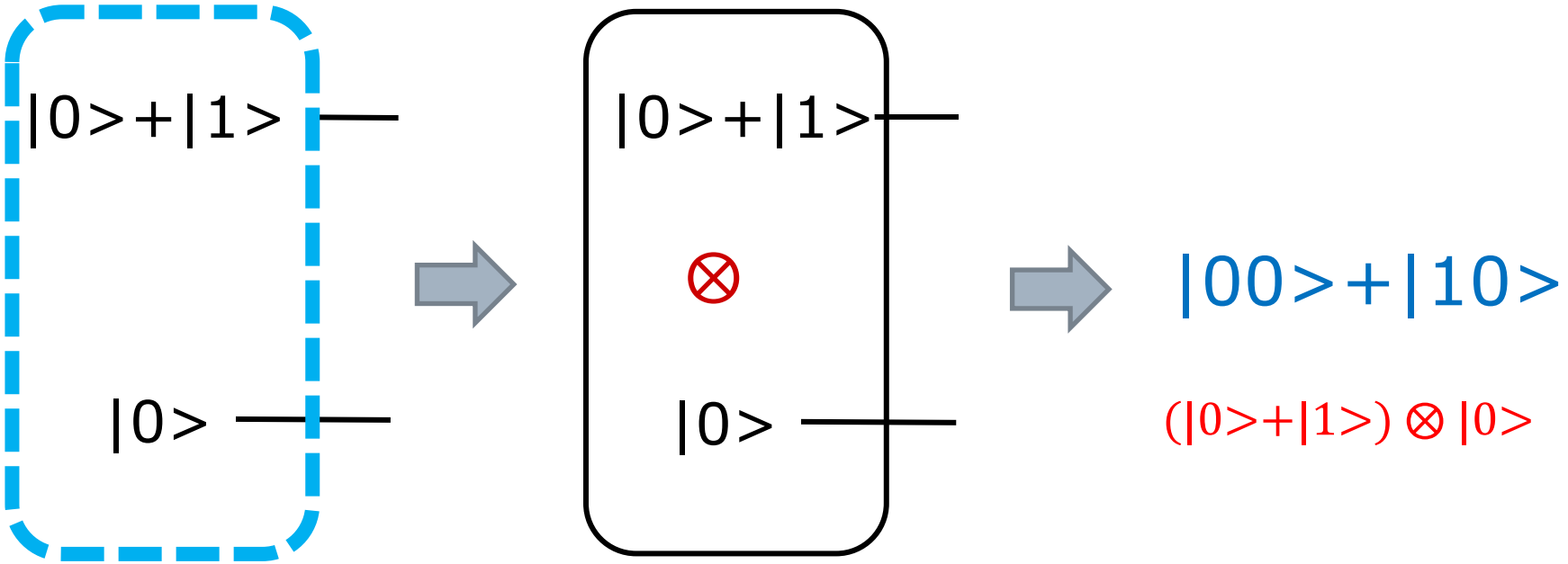
これは単位行列なので、先の回路の出力は、a|0>+b|1>

1-qubitのゲートを、直列に組み合わせる (1)



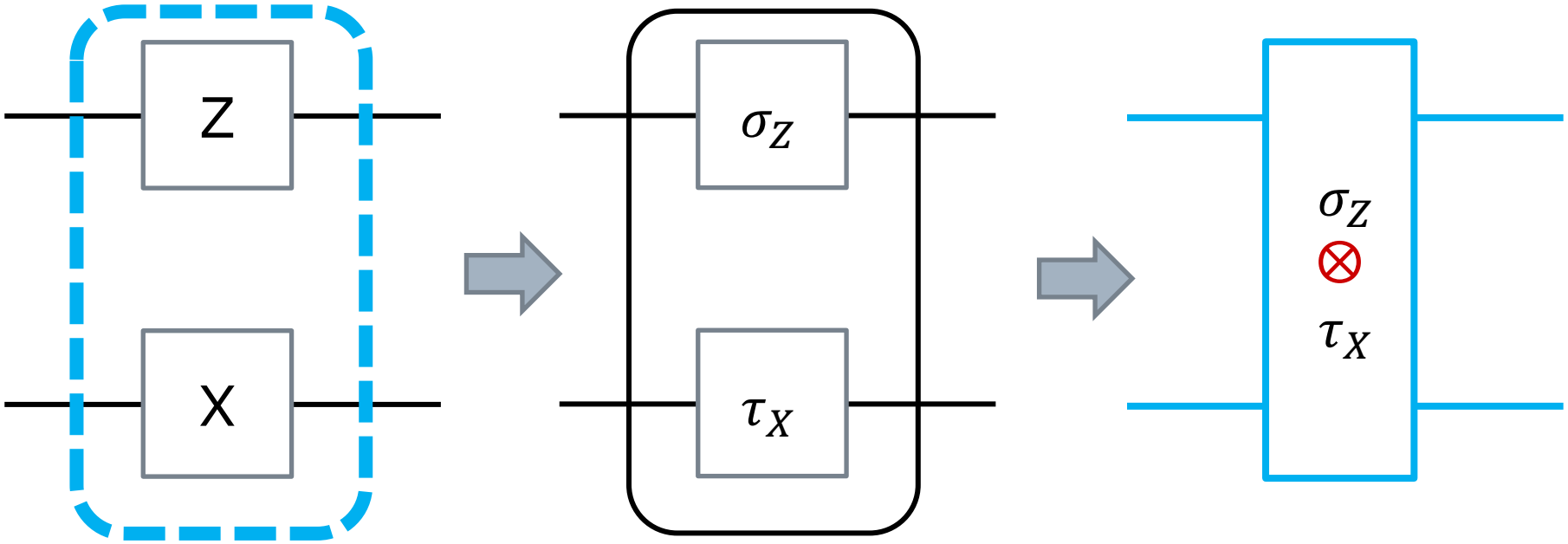
二つのqubitの状態を、並列に組み合わせる

二つの qubit がある時、それらを並列に組み合わせて、一つの状態とみなすことができる。この状態は、それぞれの状態のテンソル積である。



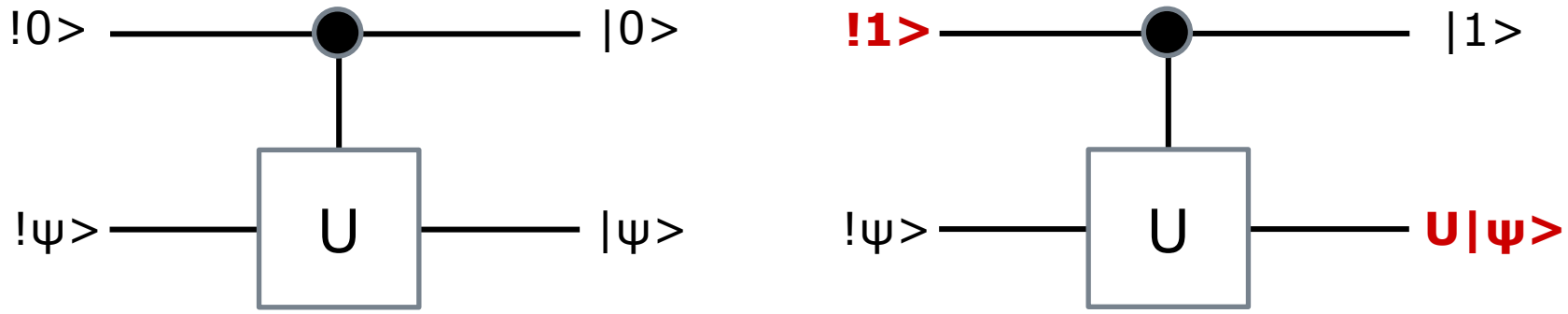
1-qubitのゲートを、並列に組み合わせる

二つの 1-qubit ゲートがある時、それらを並列に組み合わせて、2-qubitの量子ゲートを構成することができる。この量子ゲートは、それぞれの量子ゲートのユニタリ行列のテンソル積として作用する。



2-qubitのゲート **Control-U**

コントロール qubit が、|0> の時には、何もせず、（左図）
コントロール qubit が、|1> の時には、ターゲットのqubit |ψ> にユニタリ演算子 U を適用した U|ψ>を出力する（右図）
ゲートを、**Control-U ゲート**という。



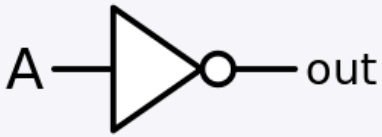
古典回路と量子回路

古典ゲートと量子ゲートの比較

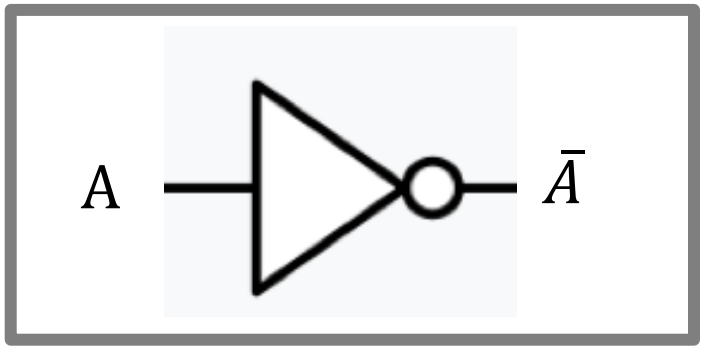
- ここでは、まず、古典的ゲートと量子ゲートの共通点を持て見よう。もちろん、入力があって出力があるのは同じである。
- qubitの|0>,|1> をそれぞれ古典ビットの 0, 1 に等しいとみなすと、入力と出力の値の対応を見れば、よく似た働きをするゲートを見つけることはできる。例えば、
  - 1bitのNOTゲートと1qubitのXゲート
  - 2bitのXORゲートと2qubitのCNOTゲート

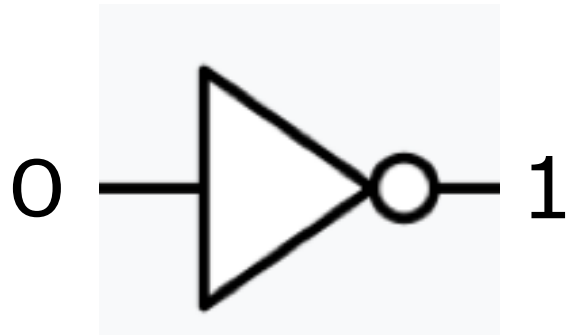
- ただ、この例でも、XORゲートの出力は一つだが、CNOTの出力は二つである。実は、量子ゲートでは、入力の数と出力の数は、つねに同じでなければならない。これについては後で述べる。

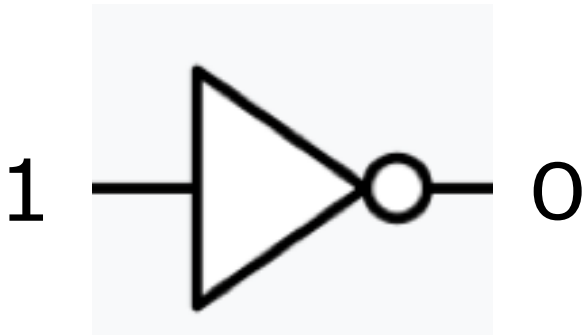
古典論理ゲートの例

論理	論理式	回路記号（MIL記号）
NOT	$\overline{A}$	
OR	$A + B$	
AND	$A \cdot B$	
XOR	$A \oplus B$	

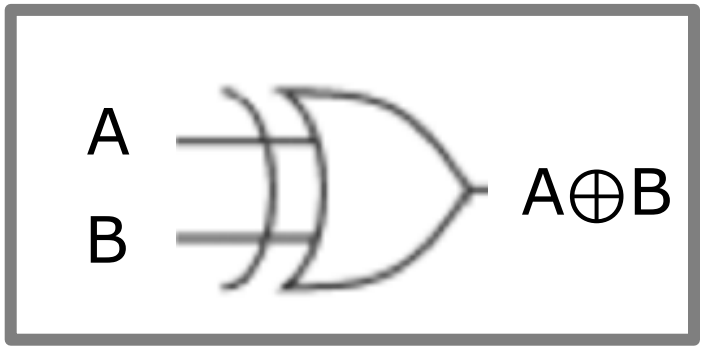
否定の論理ゲート（bit)



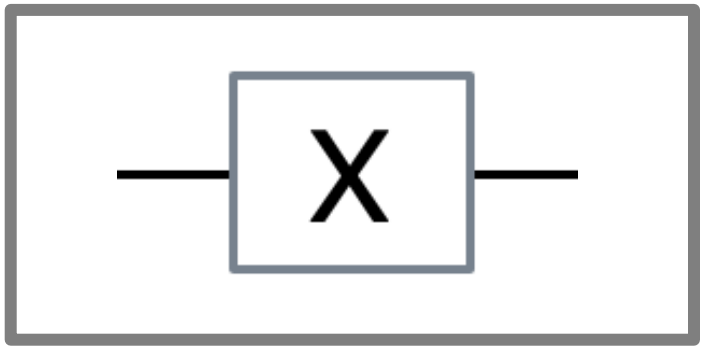




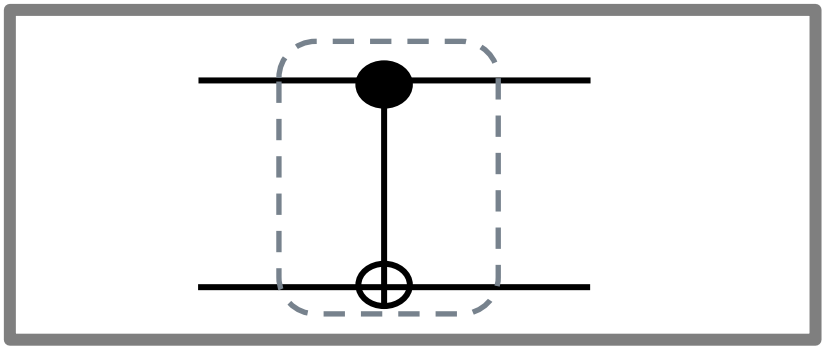
排他的論理和の論理ゲート（bit）

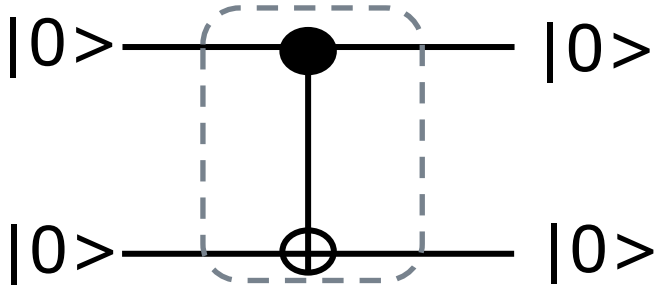


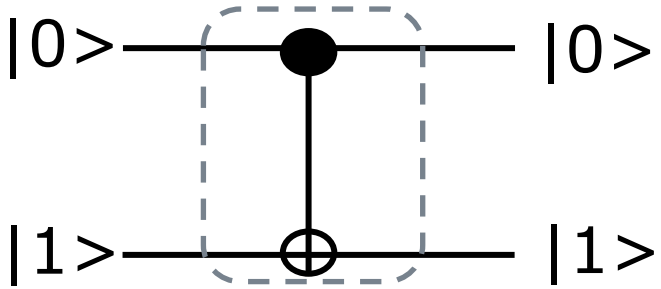
量子ゲート X（qubit)

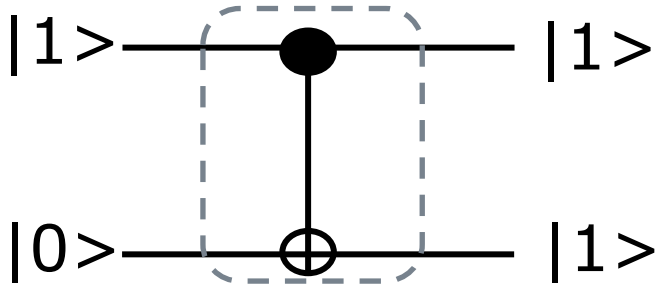


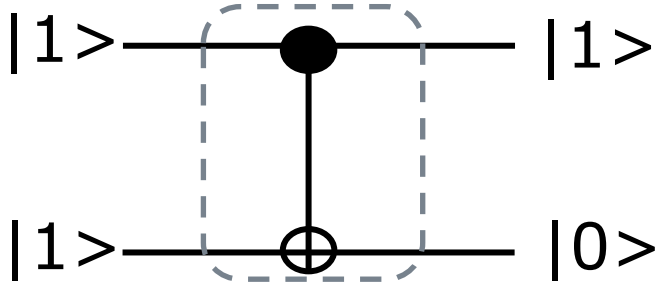
CNOT量子ゲート (qubit)



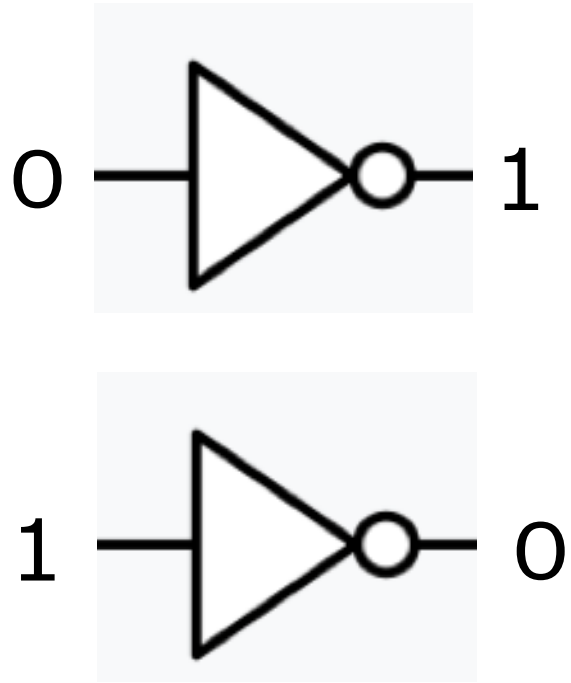






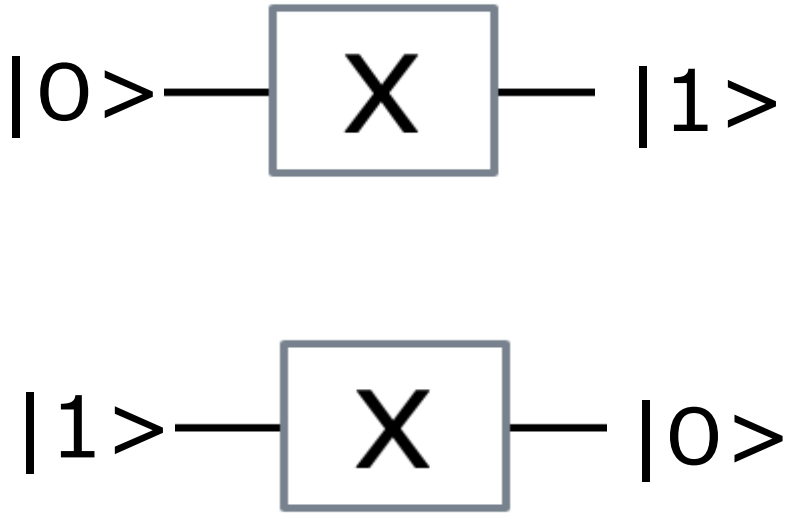


bit

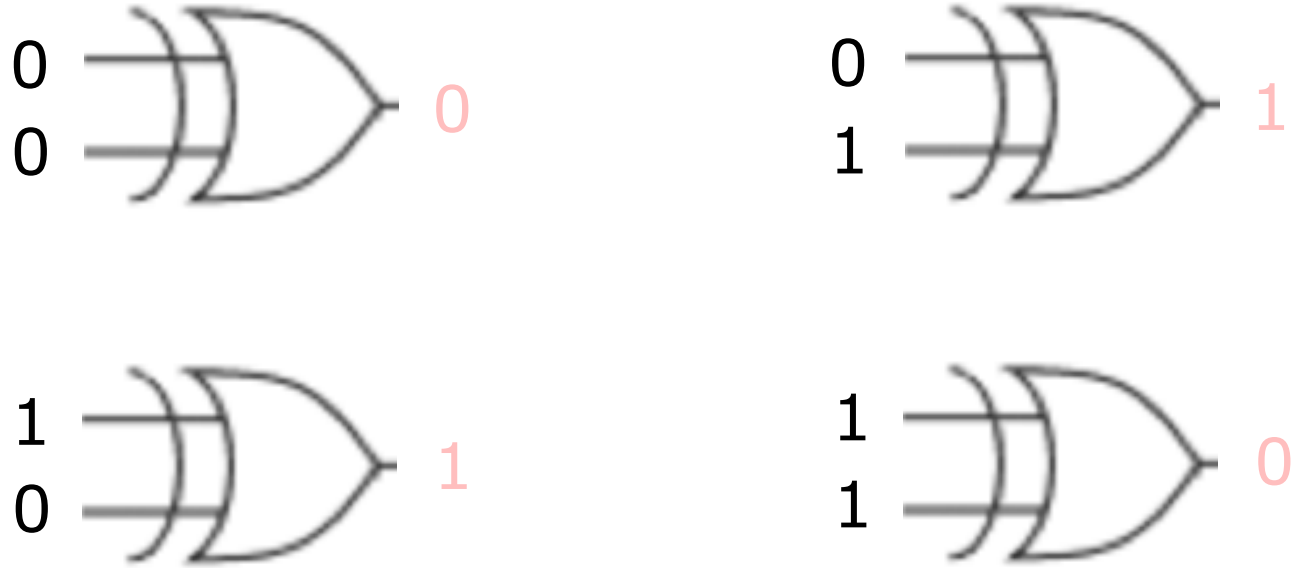


二つのゲートは、1対応している

qubit

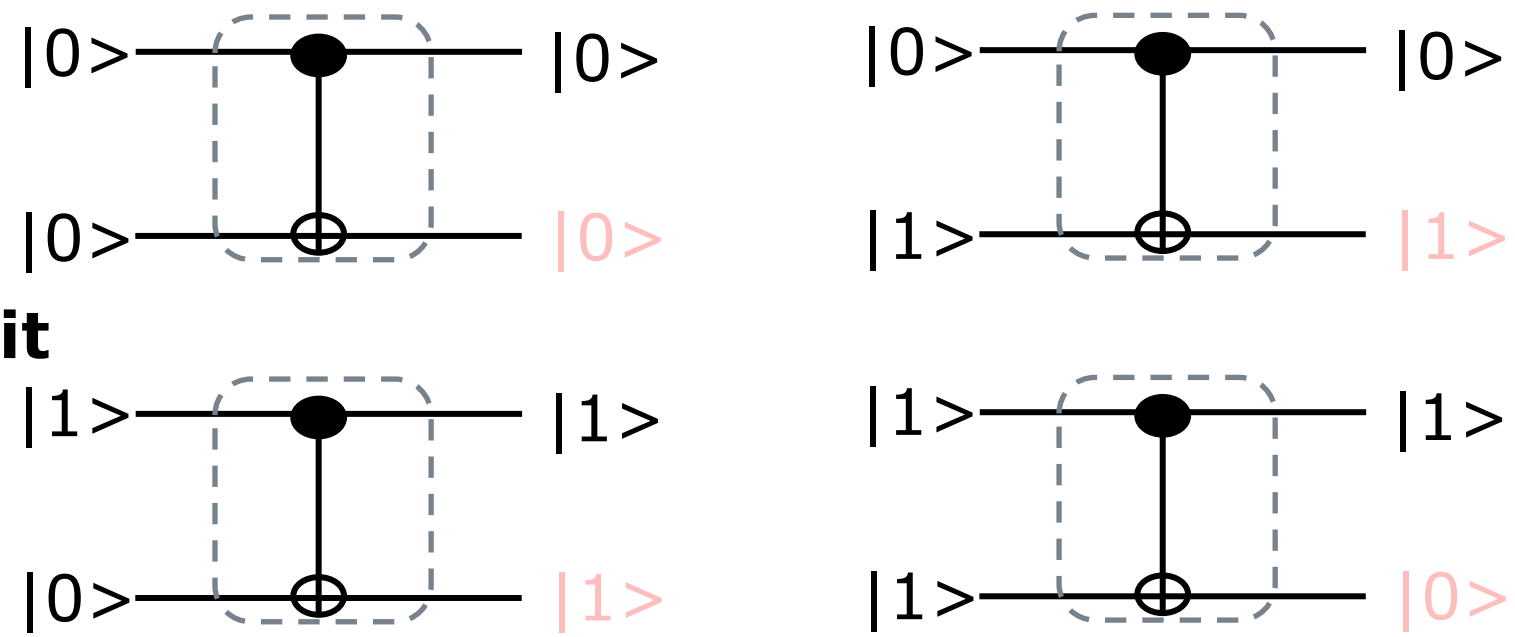


bit



二つのゲートは、対応している

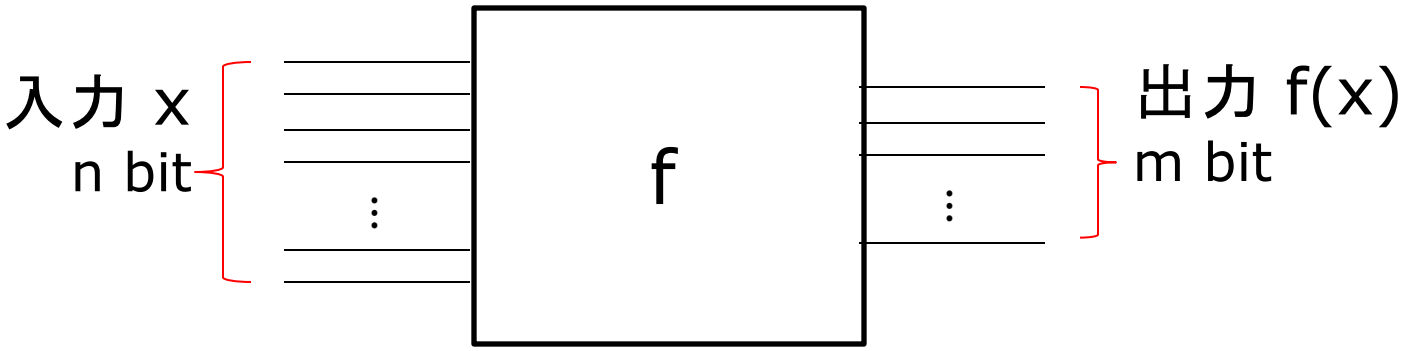
qubit



任意のf(x)を計算する
量子回路を考える

量子回路の一般的な形を考える

- 関数 f を計算する古典的な回路を考えよう。入力 x のビット数 n と、出力 f(x)のビット数 m とは一般に独立である。



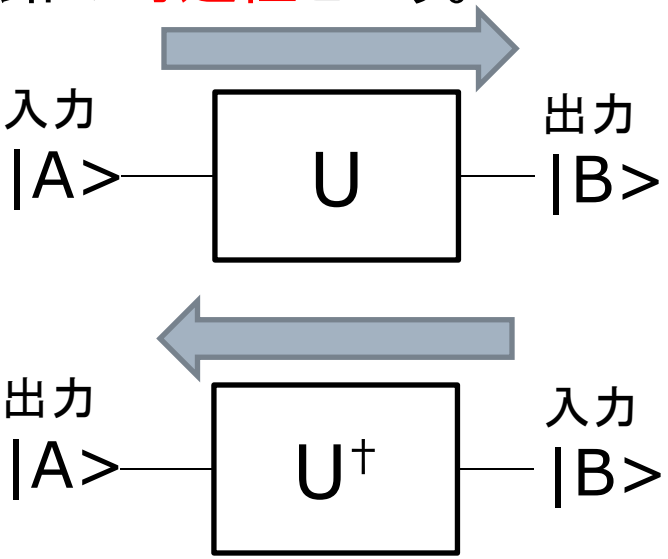
- 例えば、XORゲートでは、入力のビット数 nは2で、出力のビット数 mは1である。

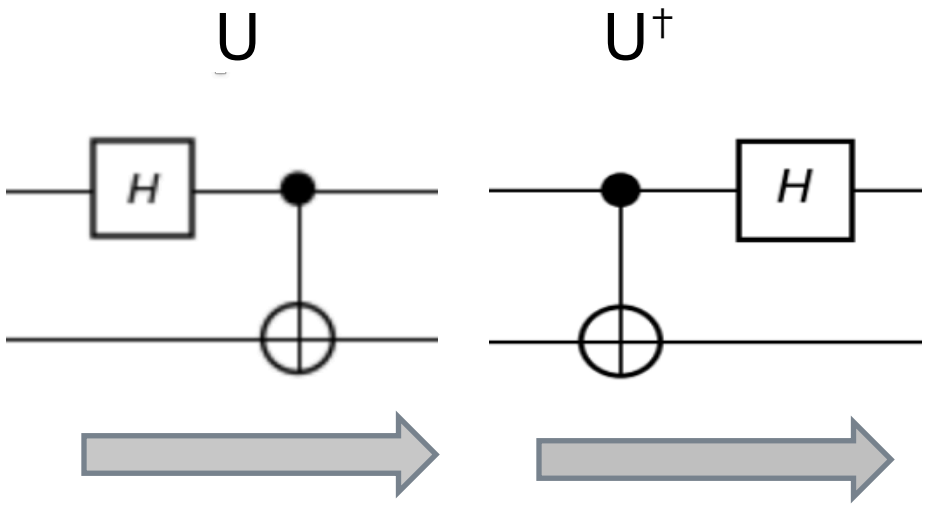
量子回路のユニタリ性

- 関数 f を計算する量子的なアルゴリズでも、入力 x のqubit数 n と、出力 f(x)のqubit数 m とは一般に独立である。
- それでは、関数 f を計算する量子回路は、先の古典回路と同じような形を取るのだろうか？　そうはならない。
- 量子ゲートはユニタリ変換を行うゲートなのだが、量子ゲートから構成される量子回路も、それ自体がユニタリ性を持たねばならない。量子回路全体に対応するユニタリ行列は、巨大なものになる。
- 例えば、2-qubitの入力を受け取り、2-qubitを出力する量子ゲート（CNOTがそうである）は、4次元のベクトルを受け取り、4次元のベクトルに変換する 4x4の行列で表現される。
- n-qubitの入力を受け取り、n-qubitを出力する量子回路は、$2^n \times 2^n$ のユニタリ行列で表現される。

量子回路の可逆性（reversibility）

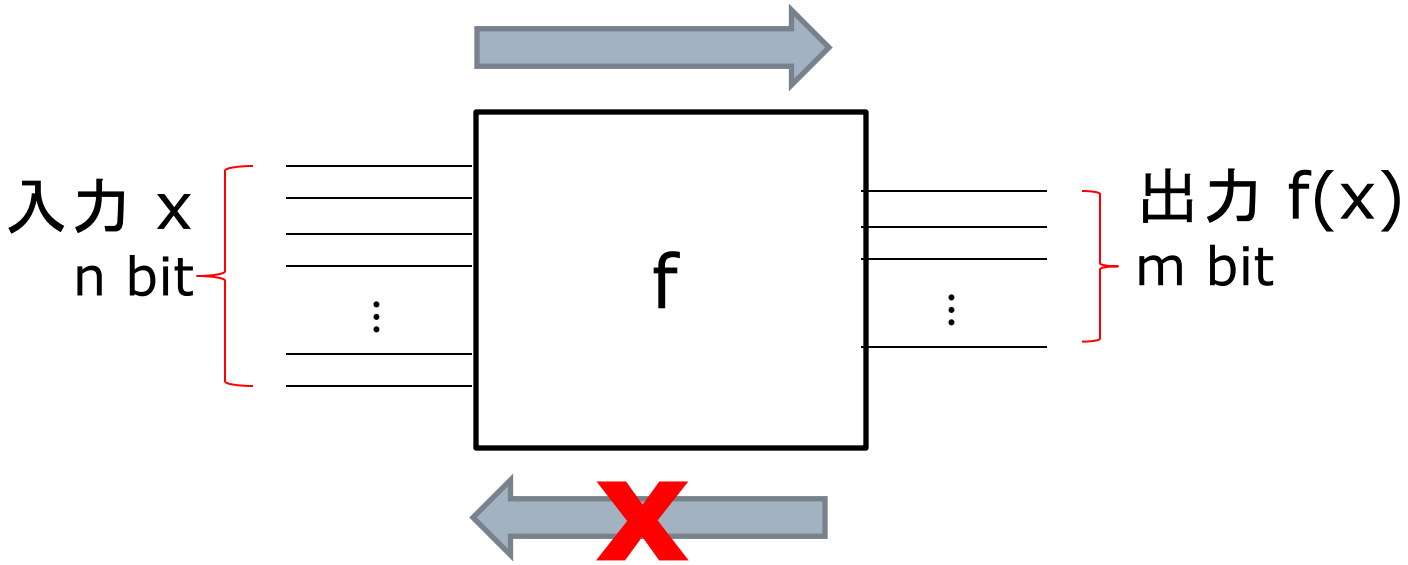
- 回路全体を表現するユニタリ行列Uを書き下ろすことができないにしても、Uの性質 $UU^{\dagger}=U^{\dagger}U=I$ から次のことがわかる。
- 回路U（行列Uに対応する）の入力と出力を入れ替えても、すなわち、Uに対するある入力|A>の結果であるUの出力|B>を、Uの出力側に与えても、Uは|A>を出力する。これを、量子回路の可逆性という。



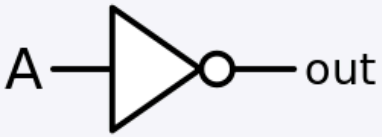


古典回路は非可逆である
任意の関数は、ユニタリでは表現できない

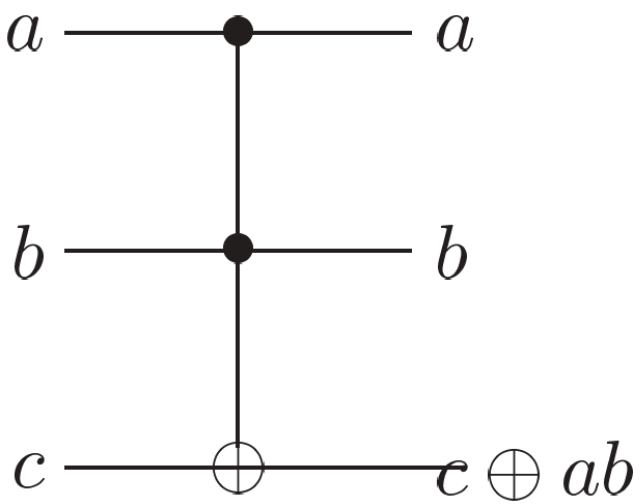
- 古典回路は、非可逆である。
- また、任意の関数 f をユニタリ行列で表現できるわけではない。
  - 例えば、f(x)=z, f(y)=z という関数 f がユニタリ行列で表現できたとする。これは、U|x>=|z>, U|y>=|z>を意味するのだが、両式を引くと $U(|x\rangle-|y\rangle)=\vec{0}$ となるのだが、こうしたUはユニタリではない。



古典論理ゲートは可逆か？

論理	論理式	回路記号（MIL記号）	
NOT	$\overline{A}$		○
OR	$A + B$		×
AND	$A \cdot B$		×
XOR	$A \oplus B$		×

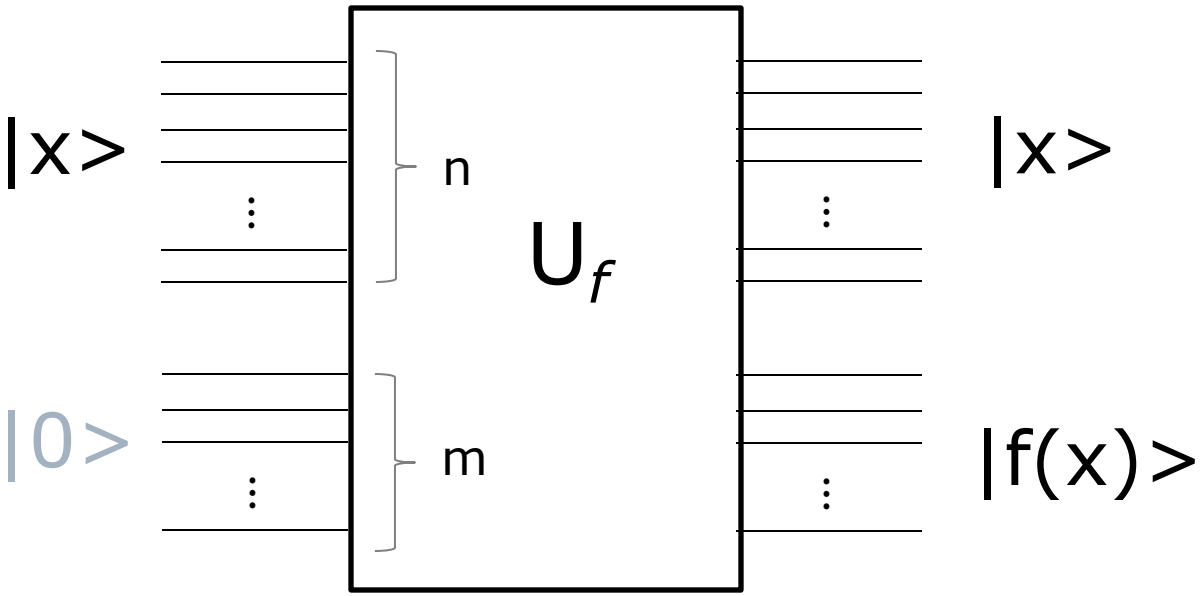
実は、次のような
ゲートを使えば、
古典回路も可逆に
できる



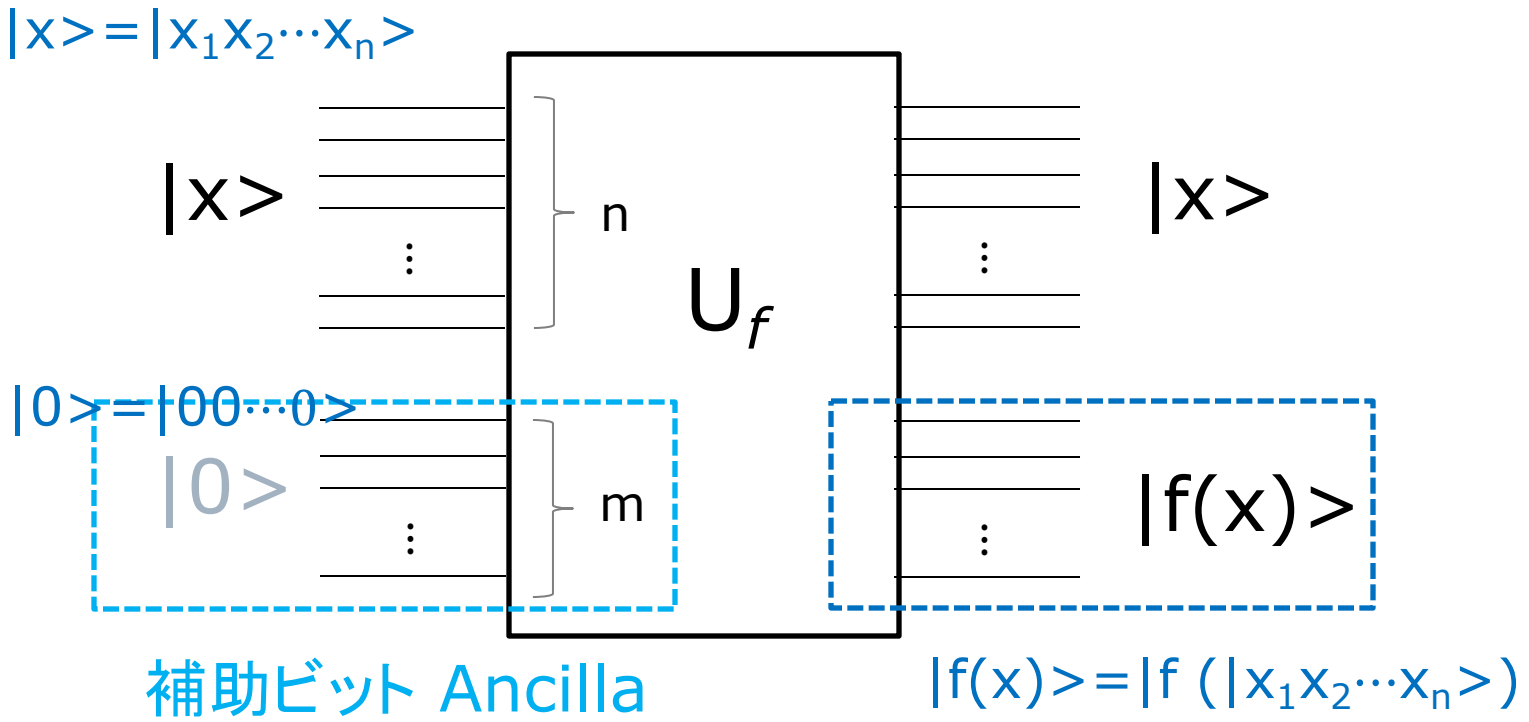
Toffoli ゲート

任意のf(x)を計算し、かつユニタリな量子回路U_fの一般的な形

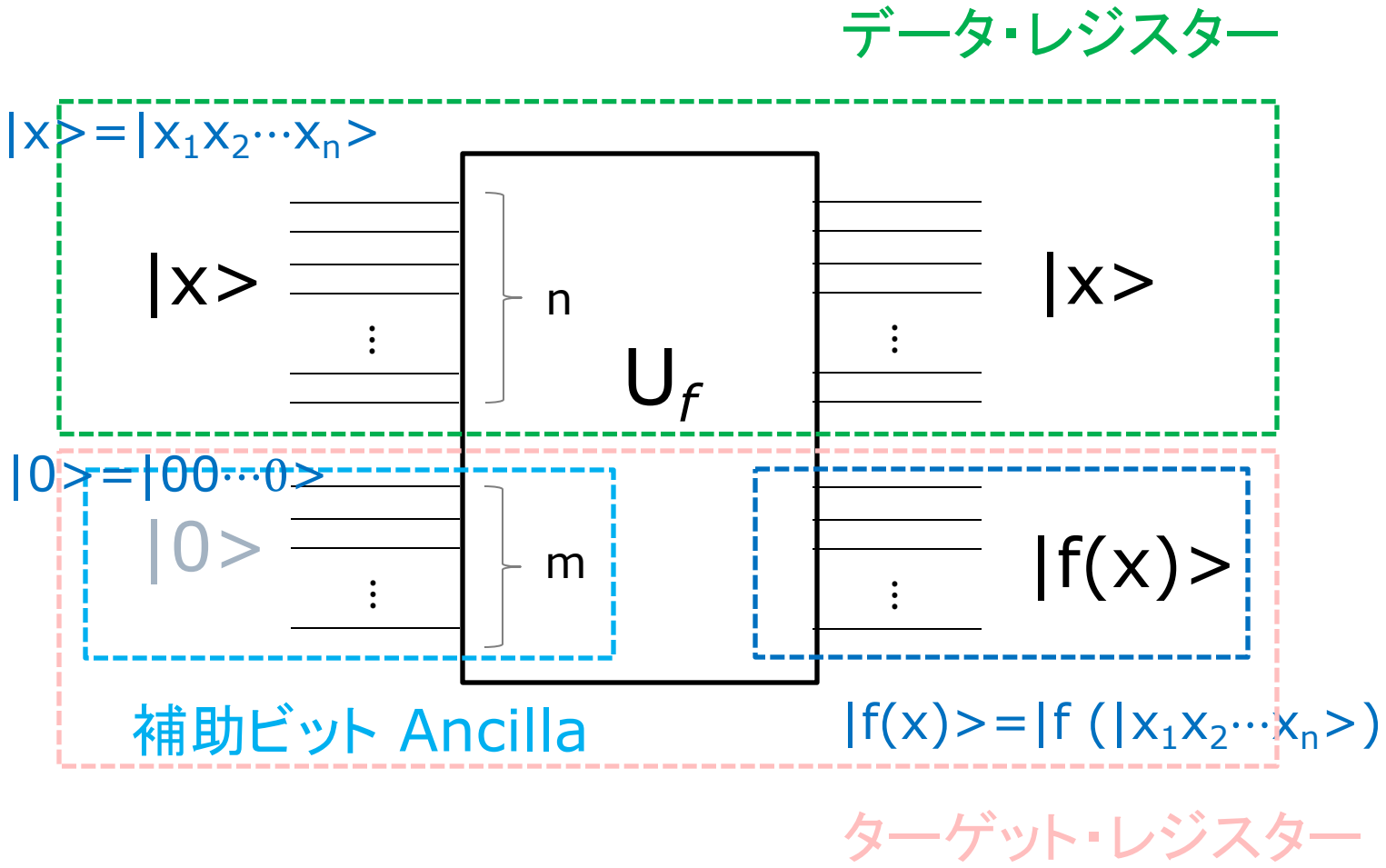
f(x)の具体的な実装を与えているわけではない。
ブラックボックスとかOracleと言ったりする



任意のf(x)を計算し、かつユニタリな量子回路U_fの一般的な形

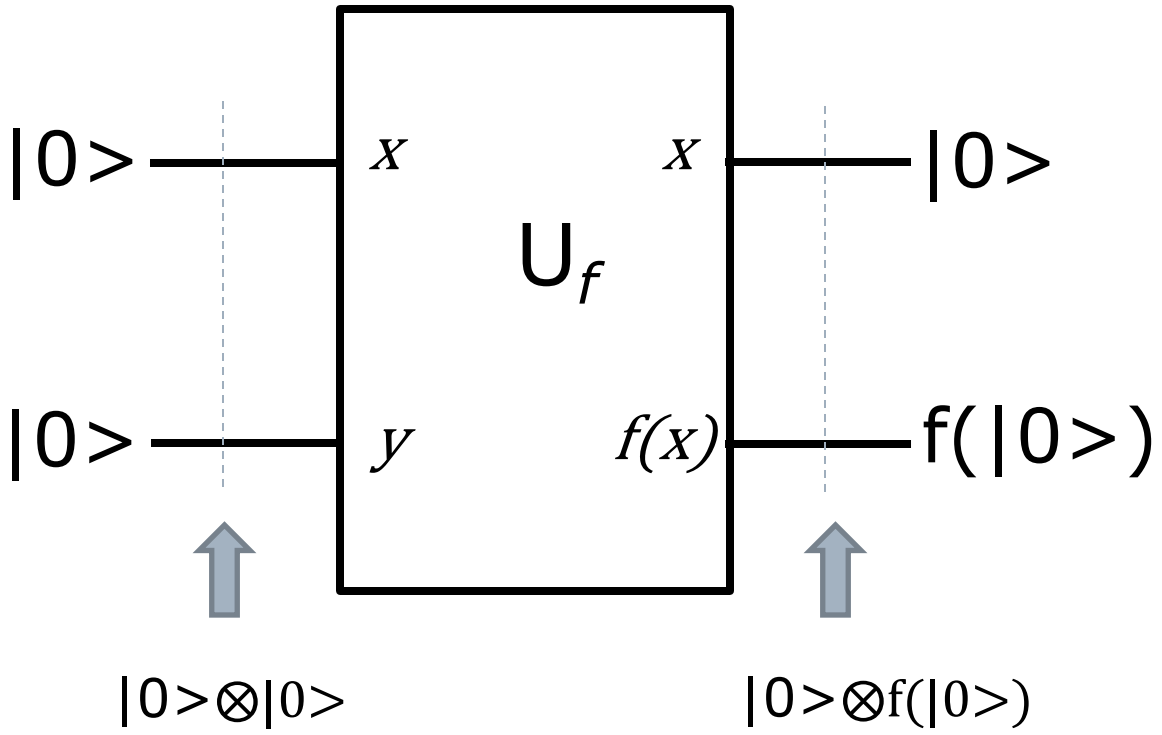


任意のf(x)を計算し、かつユニタリな量子回路U_fの一般的な形



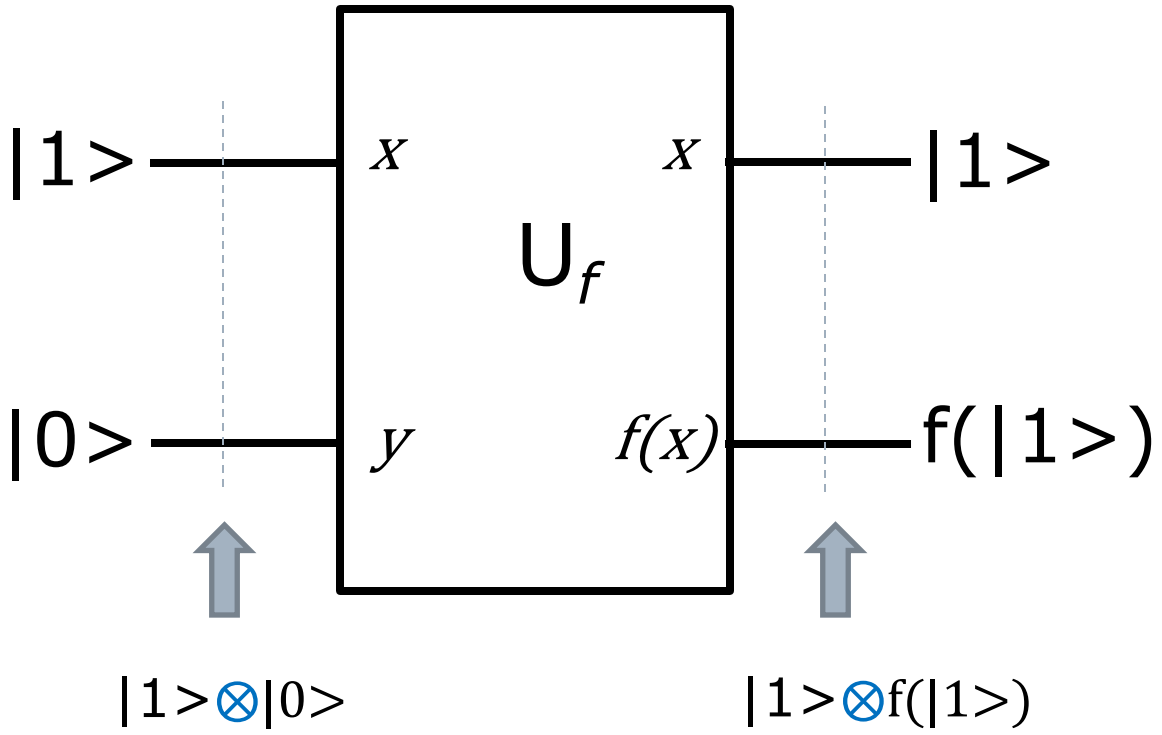
2-qubit(入力　1-qubit, 出力 1-qubit)の単純な回路U_fを考える

入力 x = |0> の時



2-qubit(fの入力 1-qubit, fの出力 1-qubit)の単純な回路U_fを考える

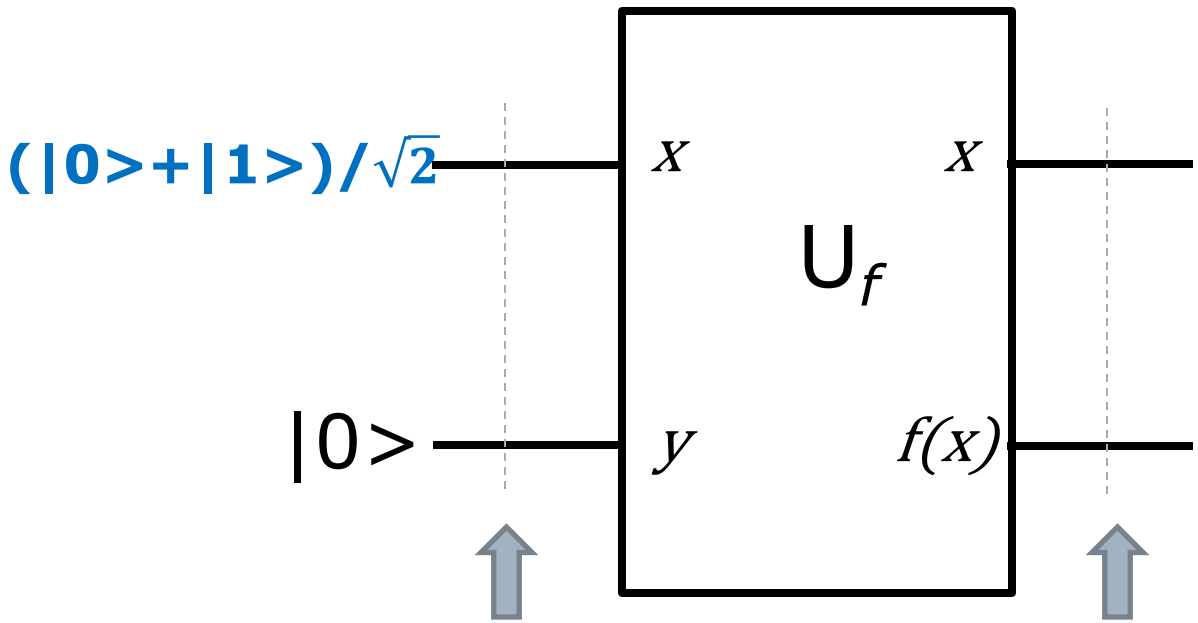
入力 x = |1> の時



意味が明確ならテンソル記号を省略してもいい

2-qubit(fの入力 1-qubit, fの出力 1-qubit)の単純な回路U_fを考える

入力 $x = (|0>+|1>)/\sqrt{2}$ の時



回路の内部で
entangleが起きると
ライン毎に分離できない
可能性がある

ただ、Ufの出力全体
の状態は計算できる

$|(|0>+|1>)/\sqrt{2} \otimes |0>$

$|0>f(|0>)/\sqrt{2} + |1>f(|1>)/\sqrt{2}$

テンソル記号を省略した

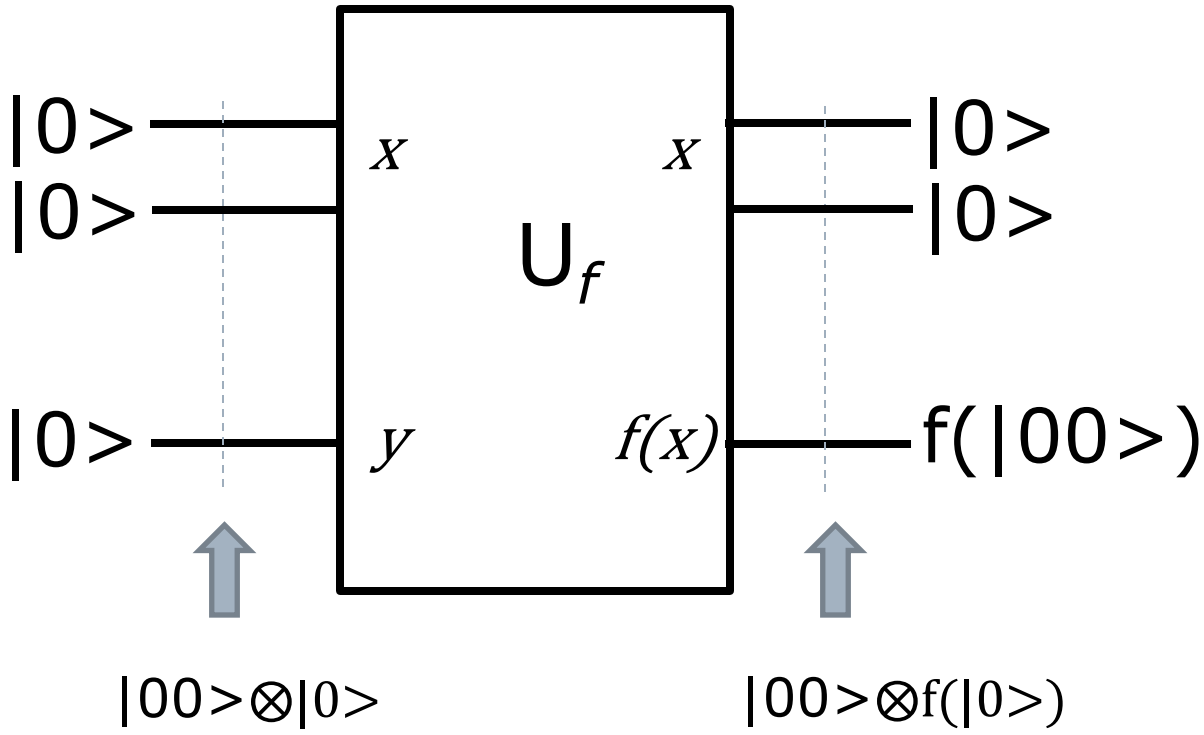
なぜ？

$|0\rangle f(|0\rangle)/\sqrt{2} + |1\rangle f(|1\rangle)/\sqrt{2}$

- 関数 f は線形であるとは限らないが、Ufはユニタリであり線形である。Mが線形であるとは、
  $M(a|A\rangle+b|B\rangle)=aM|A\rangle+bM|B\rangle$が成り立つことをいう。
- $x = |0\rangle$ の時と $x = |1\rangle$ の時の例から、Ufは次の式を満たすことがわかる。
  $Uf(|0\rangle\otimes|0\rangle)= |0\rangle\otimes f(|0\rangle)$
  $Uf(|1\rangle\otimes|0\rangle)= |1\rangle\otimes f(|1\rangle)$
- この時、Ufは線形であるので、
  $Uf((|0\rangle+|1\rangle)/\sqrt{2}\otimes|0\rangle)$
  $= Uf((|0\rangle/\sqrt{2}\otimes|0\rangle+|1\rangle/\sqrt{2}\otimes|0\rangle)$
  $= Uf(|0\rangle\otimes|0\rangle)/\sqrt{2} + Uf(|1\rangle\otimes|0\rangle)/\sqrt{2}$
  $= |0\rangle\otimes f(|0\rangle)/\sqrt{2} + |1\rangle\otimes f(|1\rangle)/\sqrt{2}$
  $= |0\rangle f(|0\rangle)/\sqrt{2} + |1\rangle f(|1\rangle)/\sqrt{2}$

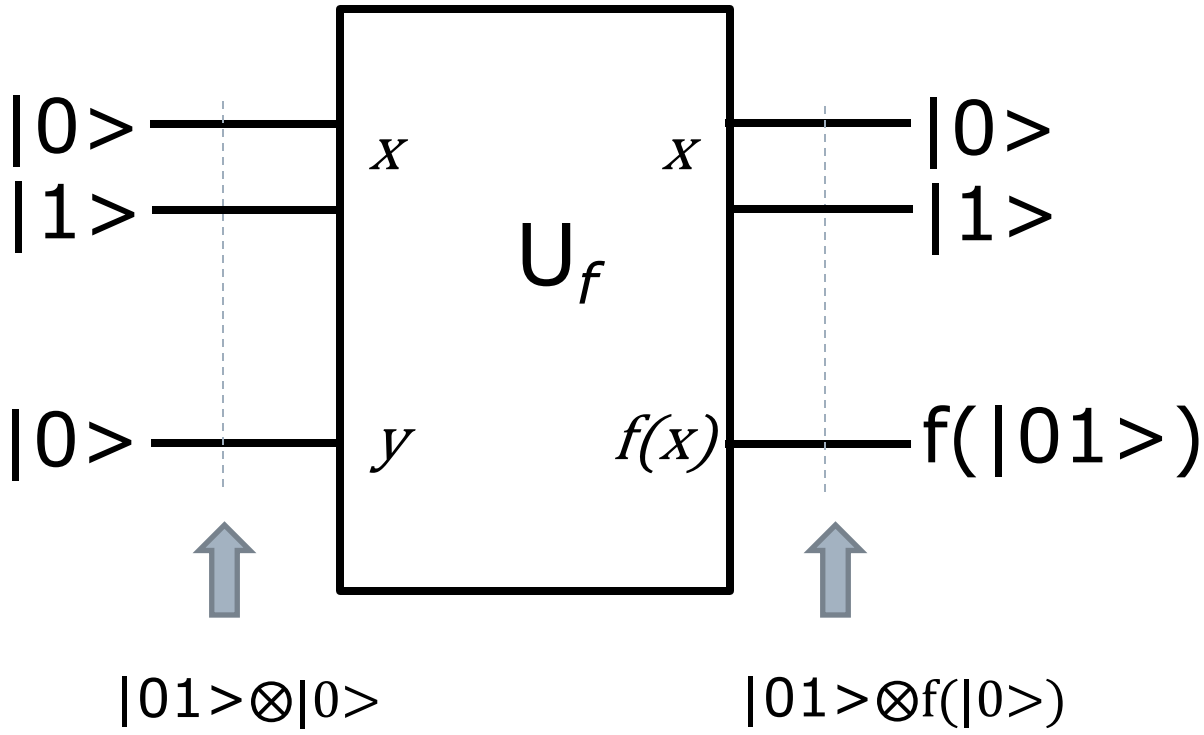
3-qubit(fの入力 2-qubit, fの出力 1-qubit)の単純な回路U_fを考える

入力 x = |0>⊗|0> = |00> の時



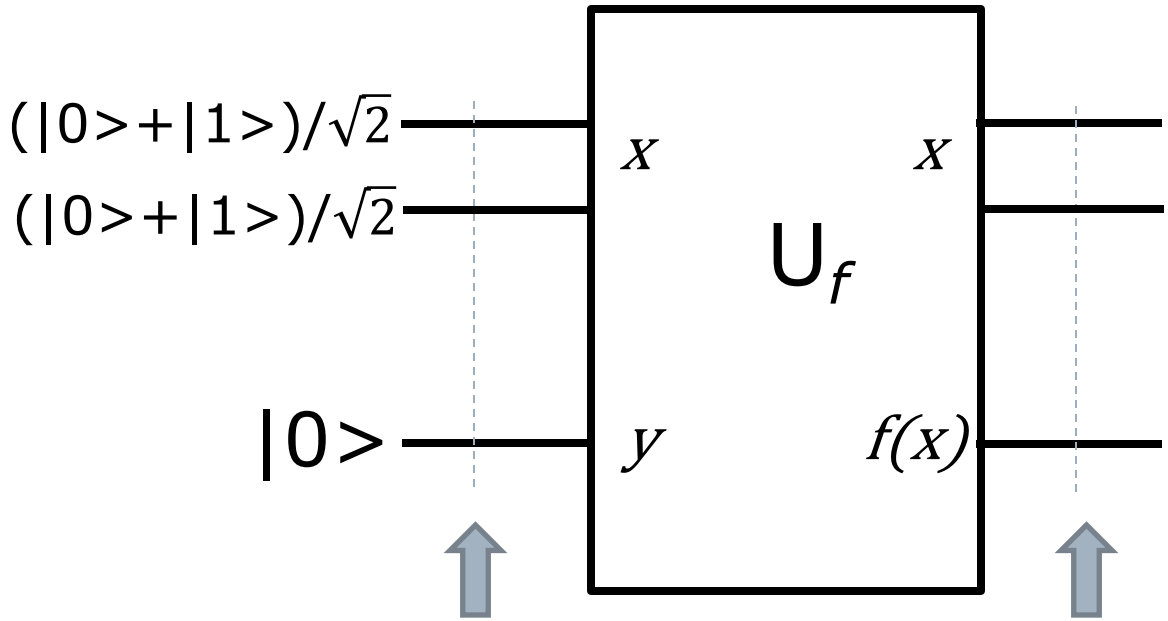
3-qubit(fの入力 2-qubit, fの出力 1-qubit)の単純な回路U_fを考える

入力 x = |0>⊗|1> = |01> の時



3-qubit(fの入力 2-qubit, fの出力 1-qubit)の単純な回路U_fを考える

入力 $x = (|0>+|1>)/\sqrt{2})\otimes(|0>+|1>)/\sqrt{2}$ の時



回路の内部で
entangleが起きると
ライン毎に分離できない
可能性がある

ただ、Ufの出力全体
の状態は計算できる

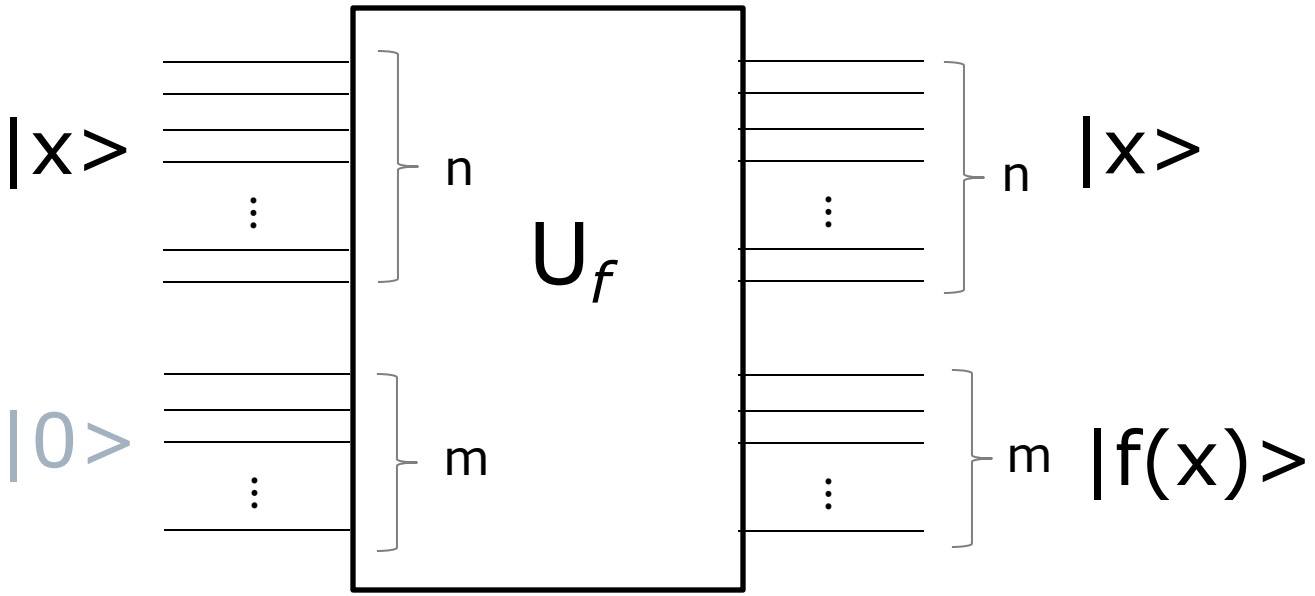
|x>⊗|0>

(|00>f(|00>) + |01>f(|01>) +
|10>f(|10>) + |11>f(|11>))/2

なぜなら

- x = (|0>+|1>)/$\sqrt{2}$)⊗(|0>+|1>)/$\sqrt{2}$ とする。
  x = (|00>+|01>+|10>+|11>)/2 である。
- この時、
  Uf(|x>⊗|0>)
  = Uf((|00>+|01>+|10>+|11>)⊗|0>)/2
  = (Uf(|00>⊗|0>)+Uf(|01>⊗|0>)+
  Uf(|10>⊗|0>)+Uf(|11>⊗|0>))/2
  = (|00>⊗f(|00>)+|01>⊗f(|01>)+
  |10>⊗f(|10>)+|00>⊗f(|11>)/2
  =(|00>f(|00>) + |01>f(|01>) +
  |10>f(|10>) + |11>f(|11>))/2

ここまでは、f(x)が1-qubitで表現される例をみてきた。このことは、関数 f(x)が、0または1の値をとることを意味する。
もし、関数f(x)が m-qubitで表現されるのなら、そのことは関数 fが、0～2^{m-1}の範囲の値を取ることを意味する。それは、古典ビットでも同様である。こうした一般化のもとでも、先に見たQuantum Parallelismは可能である。



$\{0,1\}^n \ni x$ で、

Uf($\Sigma|x>|0>$) = Σ Uf ($|x>|0>$) = Σ $|x>|f(x)>$

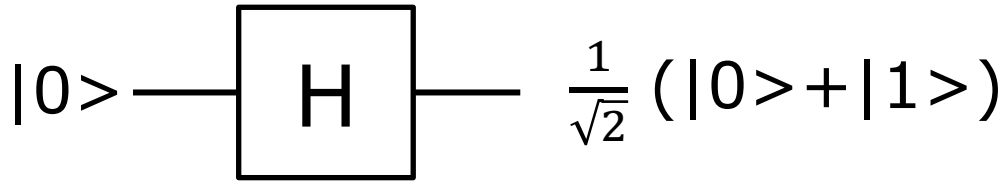
Part II

Quantum Parallelism

量子コンピュータではなぜ高速な計算ができるのか？

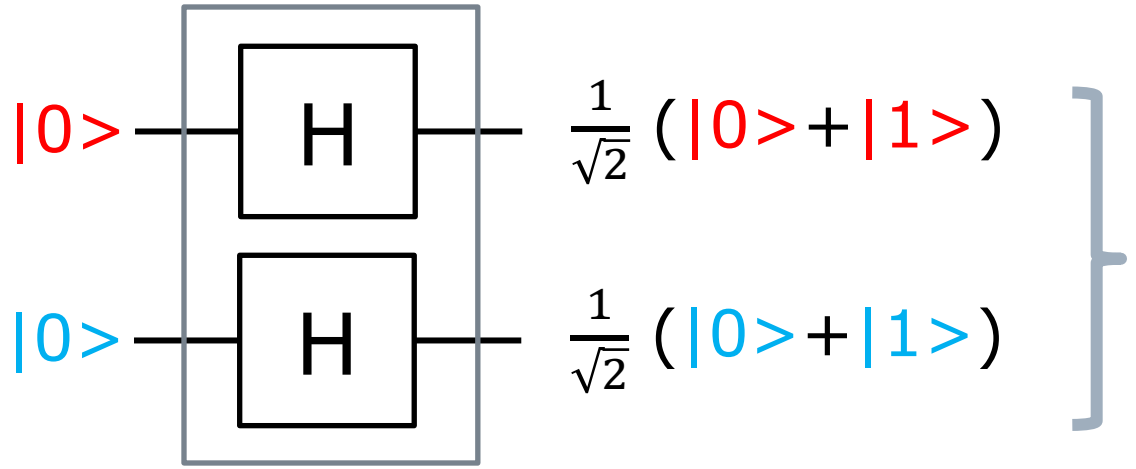
可能なすべての基底の
等しい重ね合わせを作る

アダマール・ゲート 1 個の場合



これは、可能な基底 |0>, |1>
の等しい重ね合わせである。

アダマール･ゲート 2 個の場合



$$|0\rangle\otimes|0\rangle \Rightarrow \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$$

$$= \left(\frac{1}{\sqrt{2}}\right)^2(|0\rangle\otimes|0\rangle+|0\rangle\otimes|1\rangle+|1\rangle\otimes|0\rangle+|1\rangle\otimes|1\rangle)$$

$$= \left(\frac{1}{\sqrt{2}}\right)^2(|00\rangle+|01\rangle+|10\rangle+|11\rangle)$$

これは、可能な基底 |00>, |01>, |10>, |11>
の等しい重ね合わせである。

アダマール･ゲート 2 個の場合（別解）

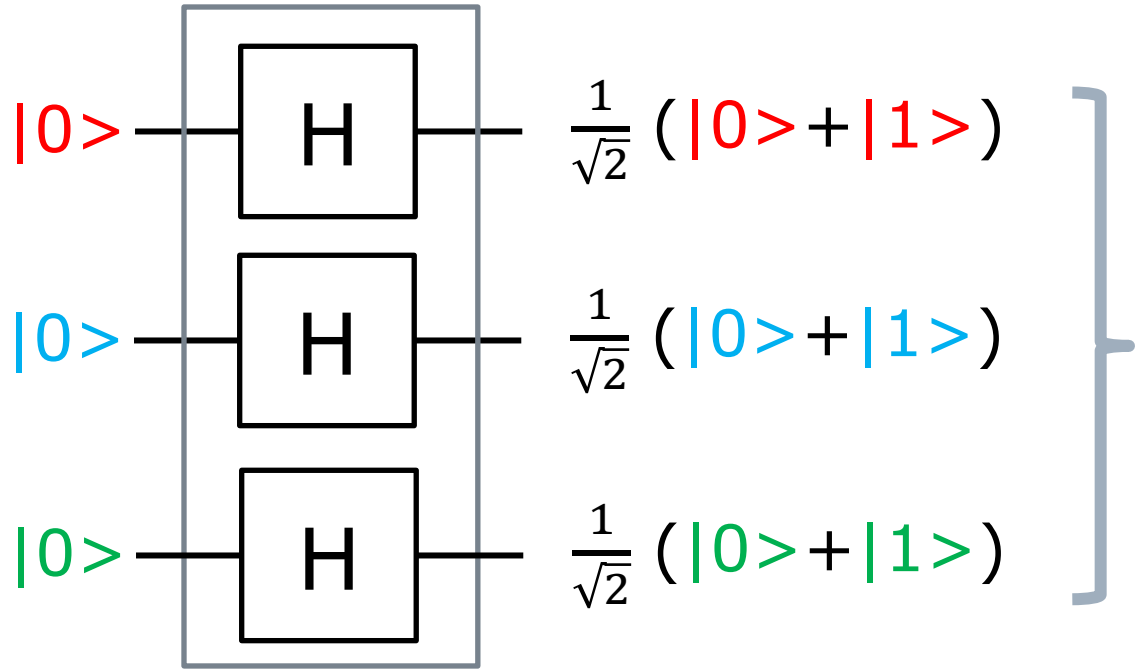
$$H \otimes H|00\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

行列のテンソル積を計算し、先に見た「ベクトルのテンソル積の例(2)」で見た |00>のベクトルを利用している。

ただ、このやり方では、並列のゲートの数が増えると行列が、2 x 2, 4 x 4(この場合), 8 x 8, 16 x 16, … と大きくなって計算しにくい。

アダマール･ゲート 3 個の場合

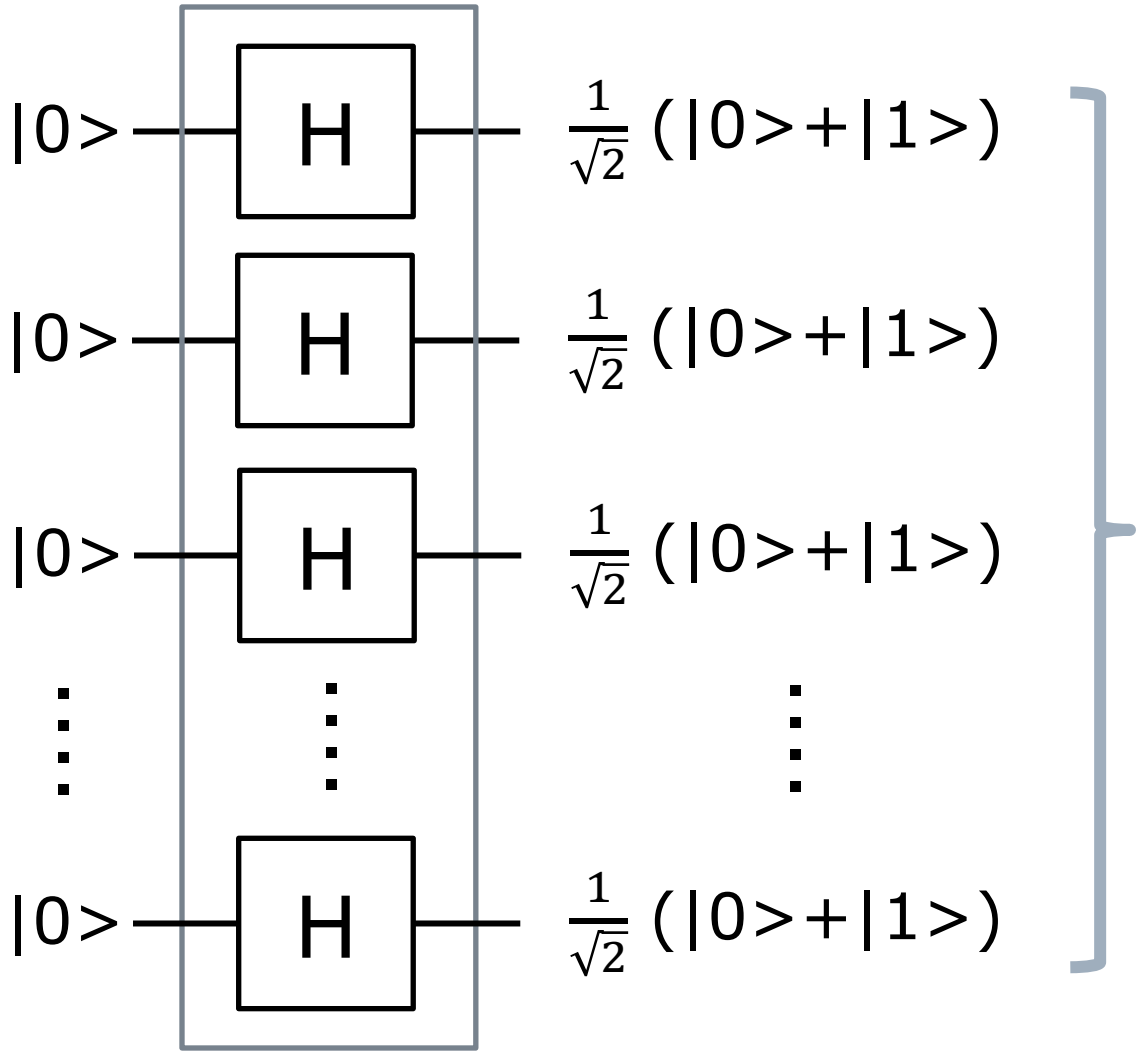


$$\Rightarrow \frac{1}{\sqrt{2}}(|0>+|1>)\otimes\frac{1}{\sqrt{2}}(|0>+|1>)\otimes\frac{1}{\sqrt{2}}(|0>+|1>)$$

$$= \left(\frac{1}{\sqrt{2}}\right)^3(|000>+|001>+|010>+|011>+$$
$$|100>+|101>+|110>+|111>)$$

これは、可能な基底すべての等しい重ね合わせである。

アダマール・ゲート n 個の場合



アダマール変換は、全ての計算基底の等しい重ね合わせを生み出す。これは、n個のゲートを使って2^n個の状態の重ね合わせを生み出す非常に効率的な方法である。

n個の出力のテンソル積

n個

$$\Rightarrow \frac{1}{\sqrt{2}}(|0>+|1>)\otimes\frac{1}{\sqrt{2}}(|0>+|1>)\otimes\cdots\otimes\frac{1}{\sqrt{2}}(|0>+|1>)$$

$$=\left(\frac{1}{\sqrt{2}}\right)^n\sum|Xi>$$

$|Xi>$は、次のようなすべての基底である

n桁

$$|X_0> = |000\cdots\cdots000>$$
$$|X_1> = |000\cdots\cdots001>$$
$$|X_2> = |000\cdots\cdots010>$$
$$|X_3> = |000\cdots\cdots011>$$
$$\vdots \qquad\qquad \vdots$$
$$|X_k> = |111\cdots\cdots111>$$

2^n個

Notation

- $H^{\otimes n}$: パラレルに置かれた n個のアダマール・ゲート
- $|0\rangle^{\otimes n}$:パラレルに置かれた n個の $|0\rangle$
- $\{0,1\}^n$:0か1の n個の並び

$$H^{\otimes n}|0\rangle^{\otimes n} = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{x_i \in \{0,1\}^n} |x_i\rangle$$

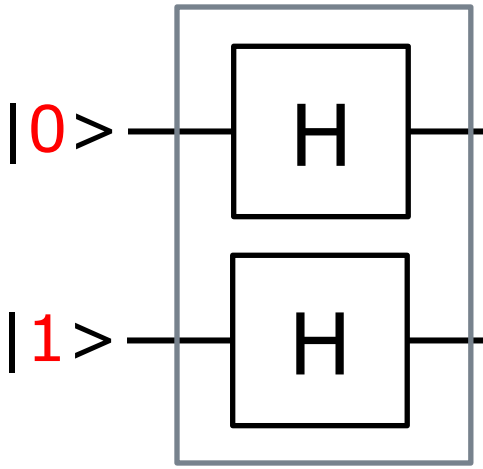
一般の場合

$$H^{\otimes n}|x\rangle = \sum_{\substack{x \in \{0,1\}^n \\ y \in \{0,1\}^n}} (-1)^{x\cdot y}|y\rangle$$

ただし、$x\cdot y = \sum x_i y_i$ (ビット毎の内積 mod 2)

$$H^{\otimes 2}|x\rangle = \left(\frac{1}{\sqrt{2}}\right)^2 \sum (-1)^{x\cdot y}|y\rangle$$

計算例



$$\frac{1}{\sqrt{2}}\left((-1)^{0\cdot 0}|0\rangle + (-1)^{0\cdot 1}|1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left((-1)^{1\cdot 0}|0\rangle + (-1)^{1\cdot 1}|1\rangle\right)$$

x:01 y:00
x･y=0+0=0
x:01 y:01
x･y=0+1=1
x:01 y:10
x･y=0+0=0
x:01 y:11
x･y=0+1=1

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$= \left(\frac{1}{\sqrt{2}}\right)^2 \Sigma\ (-1)^{x\cdot y}|y\rangle$$

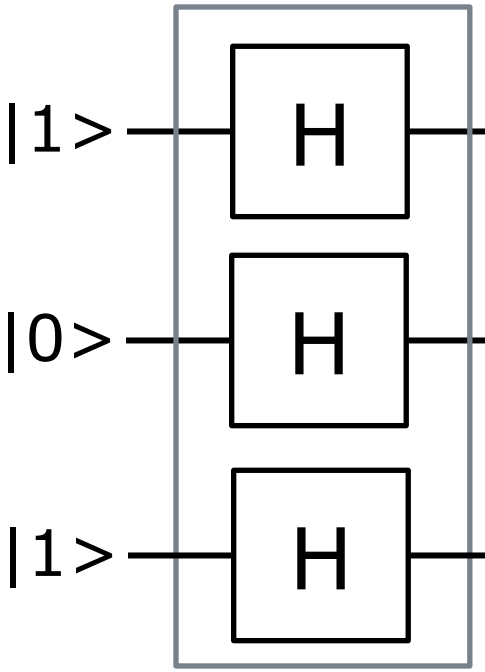
$$= \left(\frac{1}{\sqrt{2}}\right)^2 ((-1)^0|00\rangle + (-1)^1|01\rangle + (-1)^0|10\rangle + (-1)^1|11\rangle)$$

$$= \left(\frac{1}{\sqrt{2}}\right)^2 (|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

$$H^{\otimes 3}|x\rangle = \left(\frac{1}{\sqrt{2}}\right)^3 \sum (-1)^{x\cdot y}|y\rangle$$

計算例

x:101 y:000
x･y=0+0+0=0
x:101 y:001
x･y=0+0+1=1
x:101 y:010
x･y=0+0+0=0
x:101 y:011
x･y=0+0+1=1
x:101 y:100
x･y=1+0+0=1
x:101 y:101
x･y=1+0+1=0
x:101 y:110
x･y=1+0+0=1
x:101 y:111
x･y=1+0+1=0



$$\frac{1}{\sqrt{2}}\left((-1)^{1\cdot 0}|0\rangle + (-1)^{1\cdot 1}|1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left((-1)^{0\cdot 0}|0\rangle + (-1)^{0\cdot 1}|1\rangle\right)$$

$$\frac{1}{\sqrt{2}}\left((-1)^{1\cdot 0}|0\rangle + (-1)^{1\cdot 1}|1\rangle\right)$$

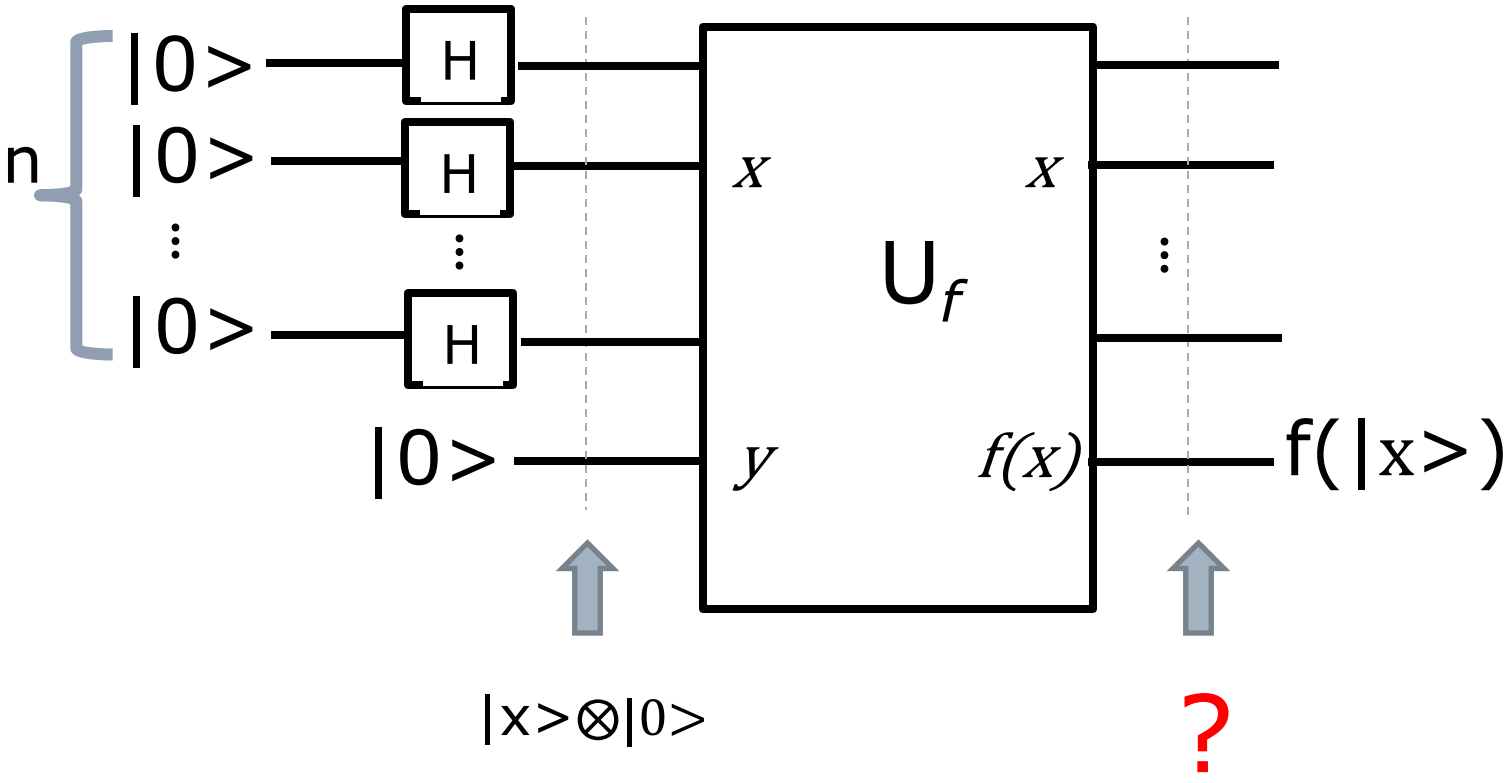
$$= \left(\frac{1}{\sqrt{2}}\right)^3 \Sigma\ (-1)^{x\cdot y}|y\rangle$$

$$= \left(\frac{1}{\sqrt{2}}\right)^3 (\ |000\rangle - |001\rangle + |010\rangle - |011\rangle - |100\rangle + |101\rangle - |110\rangle + |111\rangle\ )$$

Quantum Parallelism

(n+1)-qubit(fの入力　n-qubit, fの出力 1-qubit)の回路U_fを考える

Ufの入力に、パラレルに置かれたアダマール・ゲート n個の出力（それぞれの入力は|0>）をつなげてみよう



アダマール･ゲート n 個の出力

n個の出力のテンソル積

n個

$$\Rightarrow \frac{1}{\sqrt{2}}(|0>+|1>)\otimes\frac{1}{\sqrt{2}}(|0>+|1>)\otimes\cdots\otimes\frac{1}{\sqrt{2}}(|0>+|1>)$$

$$= \left(\frac{1}{\sqrt{2}}\right)^n \sum |Xi>$$

$|Xi>$は、次のようなすべての基底である

n桁

$$|X_0> = |000\cdots\cdots000>$$
$$|X_1> = |000\cdots\cdots001>$$
$$|X_2> = |000\cdots\cdots010>$$
$$|X_3> = |000\cdots\cdots011>$$
$$\vdots \qquad\qquad \vdots$$
$$|X_k> = |111\cdots\cdots111>$$

2^n個

先の回路U_fの出力を考える

- Ufへの入力xは、$H^{\otimes n}|0\rangle^{\otimes n}$ で、これは$\left(\frac{1}{\sqrt{2}}\right)^n \sum |Xi\rangle$ に等しい。ただし、$Xi \in \{0,1\}^n$であるすべてについて和をとる。
- $Uf(|x\rangle \otimes |0\rangle) = Uf\left(\left(\frac{1}{\sqrt{2}}\right)^n \sum |Xi\rangle \otimes |0\rangle\right)$

  $= \left(\frac{1}{\sqrt{2}}\right)^n \sum Uf(|Xi\rangle \otimes |0\rangle)$

  $= \left(\frac{1}{\sqrt{2}}\right)^n \sum (|Xi\rangle \otimes f(|Xi\rangle))$

Quantum Parallelism

- 例えば、n=3の時、Ufの出力は次のようになる。

$$\left(\frac{1}{\sqrt{2}}\right)^3 (\; |000> f(|000>) + |001> f(|001>) + |010> f(|010>) + |011> f(|011>) + |100> f(|100>) + |101> f(|101>) + |110> f(|110>) + |111> f(|111>) \;)$$

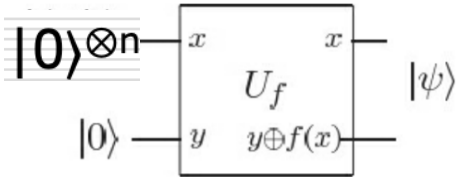
ただし、|x> ⊗ f(|x>)のテンソル記号を省略している。
三つのqubitに対する一回のUfの呼び出しが、f(|000>)から
f(|111>)までの8つのfの値を計算していることがわかる。

一般に、n個のqubitに対するUfの呼び出しは、2^n個の f の値を計算することになる。

Quantum Parallelism

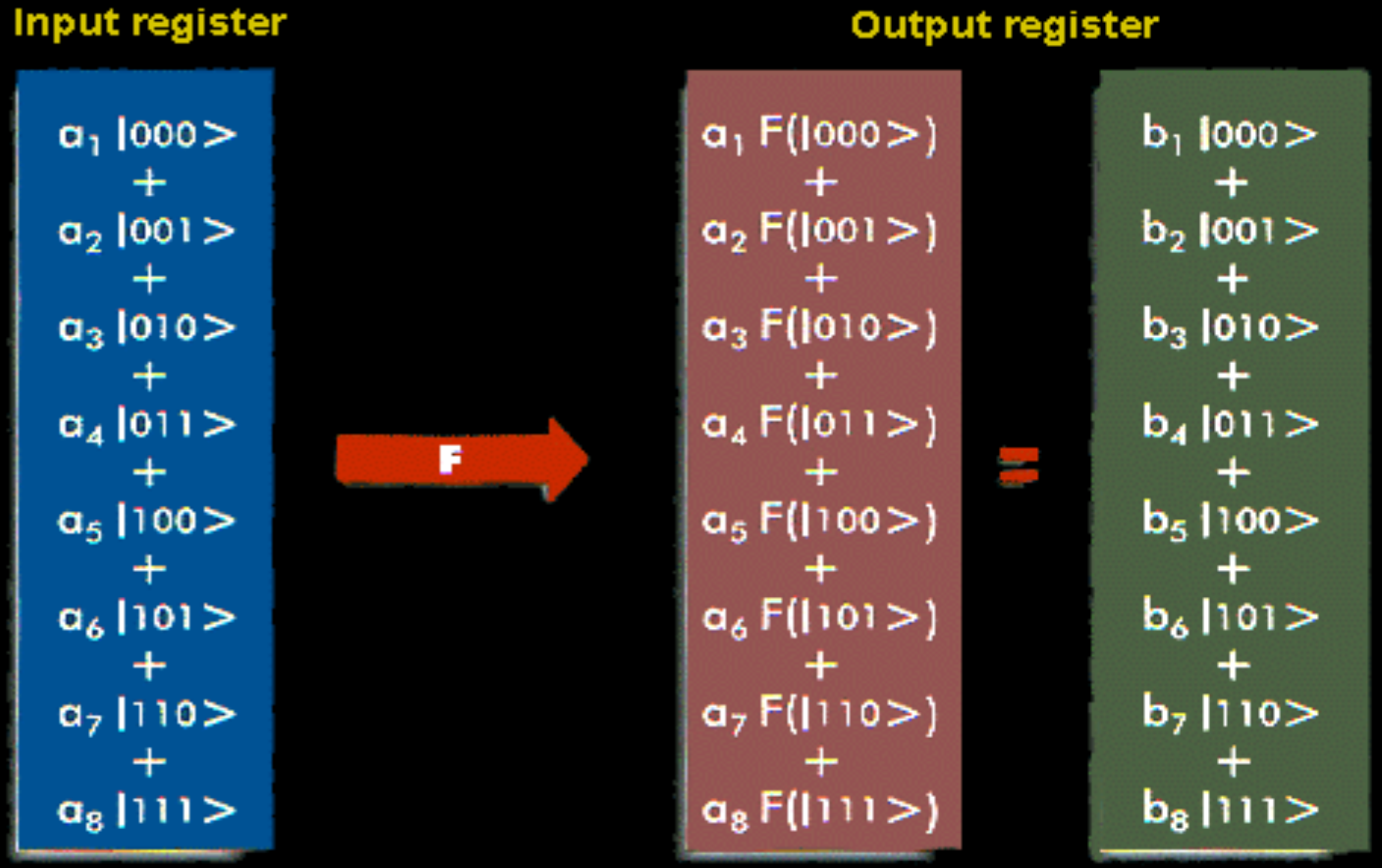
- nビットの入力xと1ビットの出力 f(x)を持つ関数の量子並列評価は、次のように実行される。
- n+1 qubitの状態 $|0\rangle^{\otimes n}|0\rangle$を用意する。次に、最初のn qubitに、アダマール変換を適用する。その出力をU_fの量子回路の入力に接続すると、次の状態が生み出される。

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle|f(x)\rangle$$



- ある意味では、量子並列処理は、関数fの全ての可能な2^n個の値を同時に評価する。たとえ、我々は明らかにfを一回だけ評価するのであるが。

n個のqubitで、2^n個の並行計算が可能



量子コンピュータの出力は、
どう取り出せるのか？

出力を取り出す＝出力を観測する

- 先のような回路を組めば、一回のUfの適用で出力側に $\{0,1\}^n \ni x$ なるすべてのxについて、$\Sigma|x> f(|x>)$ が現れるので、2^n 回の f の計算ができるように見える。
- ところが、話はそう簡単ではない。出力結果を取り出すということは、結果を観測することである。
- 観測について復習しておこう。

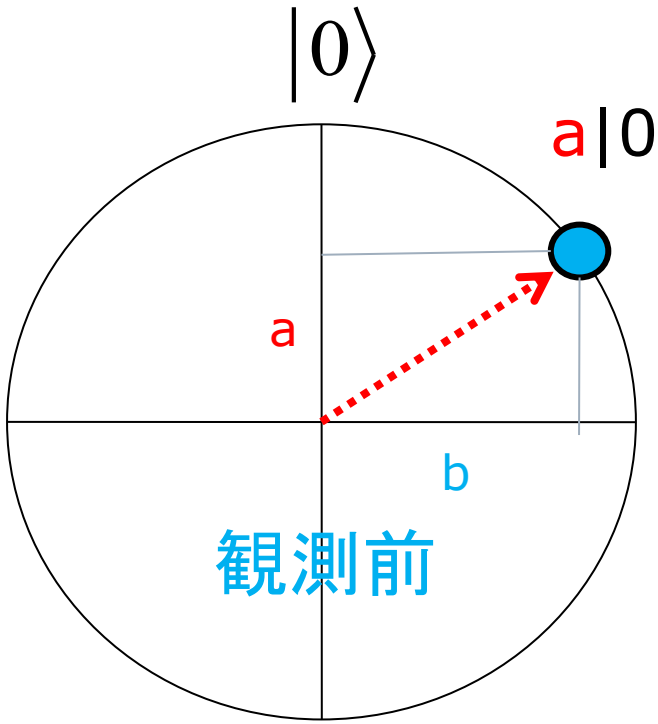
qubitの観測

- |Qubit> = a|0>+b|1> を観測すると、重ね合わせの状態は破れて失われ、|0>または|1>のいずれかの状態になる。

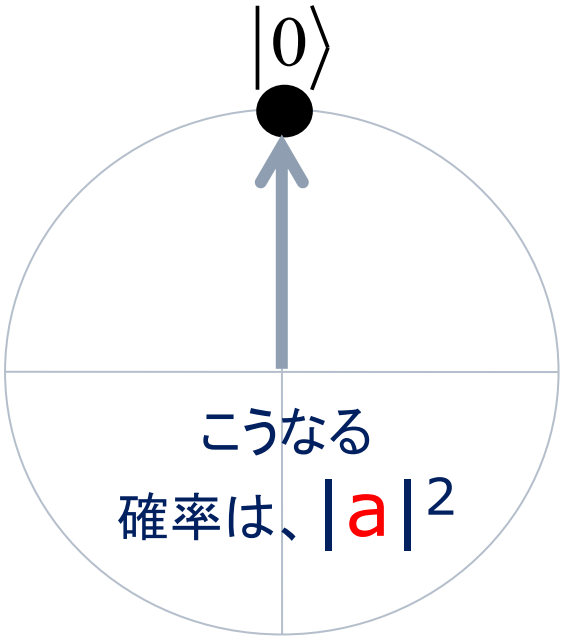
- この時、
  |0>を観測する確率は、$|a|^2$で与えられ、
  |1>を観測する確率は、$|b|^2$で与えられる。

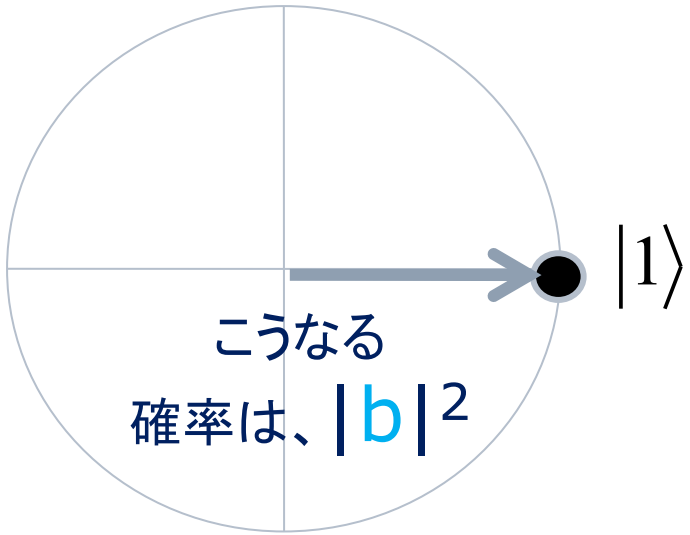
qubitの観測

観測によって、
「重ね合わせの状態」は
失われる



観測後





n-qubitの量子状態の観測
Bornのルール

一般の量子状態 $|\psi\rangle = \sum_{k=0}^{n-1} c_k|k\rangle$ が与えられた時、

状態 $|0\rangle$ が観測される確率は、$|c_0|^2$ で与えられる。
観測前の状態 $|\psi\rangle$は、新しい状態 $|0\rangle$に変わる。

状態 $|1\rangle$ が観測される確率は、$|c_1|^2$ で与えられる。
観測前の状態$|\psi\rangle$は、新しい状態 $|1\rangle$に変わる。

…

状態 $|k\rangle$ が観測される確率は、$|c_k|^2$ で与えられる。
観測前の状態$|\psi\rangle$は、新しい状態 $|k\rangle$に変わる。

一般化されたBornのルール

システムAの状態が $|k\rangle_A^n$ で表され、システムBの状態が $|\psi_k\rangle_B^m$ で表される時、次の状態は、A ⊗ Bの状態である。

$$|\varphi\rangle^{n+m} = |0\rangle_A^n \, |\psi_0\rangle_B^m + |1\rangle_A^n \, |\psi_1\rangle_B^m + \cdots + |2^n - 1\rangle_A^n \, |\psi_{2^n-1}\rangle_B^m$$

この状態が観測された時、
システムAの状態は、特定の一つのk_0について $|k_0\rangle_A^n$ に変化し、
システムBの状態は、この同じk_0について $|\psi_{k_0}\rangle_B^m$ に変化する。
システムA ⊗ Bとしてみれば、
状態 $|\varphi\rangle^{n+m}$ は、新しい状態 $|k_0\rangle_A^n \, |\psi_{k_0}\rangle_B^m$ に変化する。

$\Sigma|x> f(|x>)$の観測

- 重ね合わせの状態の$\Sigma|x> f(|x>)$ を観測したとたんに、重ね合わせの状態は失われ、我々が観測するのは 個別の $|x> f(|x>)$のみである。
- 例えば、n=3の時、Ufの出力が次のようになっていても、

$$\left(\frac{1}{\sqrt{2}}\right)^3 (\\ |000> \mathrm{f}(|000>) + |001> \mathrm{f}(|001>) \\ + |010> \mathrm{f}(|010>) + |011> \mathrm{f}(|011>) \\ + |100> \mathrm{f}(|100>) + |101> \mathrm{f}(|101>) \\ + |110> \mathrm{f}(|110>) + |111> \mathrm{f}(|111>) \\ )$$

我々が観測できるのは、xの一つの値についての $|000> \mathrm{f}(|000>)$ または $|001> \mathrm{f}(|001>)$ または・・・ $|111> \mathrm{f}(|111>)$ のみである。

何かが必要である

- 量子並列計算は、直ちに有用なわけではない。
  最初の単一qubitのサンプルで $\frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}}$、

  この状態の測定は、|0> f(0) 〉か |1> f(1)〉の値のどれかを返すだけだ。もっと一般的に、状態 Σ_x |x> f(x) 〉の測定は、一つの値xについてのf(x)の値を返すだけだ。もちろん、古典的なコンピュータは、そうしたことを簡単にやってのける。
- 量子計算が役に立つためには、単なる量子並行計算以上の何かが必要になる。すなわち、Σ_x |x> f(x) 〉のような重ね合わせの状態から、f(x)の一つの値より多くの値の情報を引き出せるような能力が必要になる。

Part III

Simonのアルゴリズム

Simonの問題

$f(x)=f(x \oplus a)$ となる 関数fの「周期」 aを求める

Simonの問題
「f(x)=f(x⊕a) となるaを求めよ」

- n-bitで表される整数からn-bitで表される整数への関数 fがあって、ある整数aについて f(x)=f(x⊕a) が成り立つという。この時、「周期」aを求めよという問題を、Simonの問題という。

- x⊕a は、ビット列x とビット列a のビット毎の排他的論理和XORである。（対応するbitが同じなら0に、違っていたら1）
  例えば、 $1101 \oplus 1101 = 0000, 0000 \oplus 1101 = 1101$である。
- この問題は、aとして全てのビットが0である 000…00 をとれば、x⊕a=x ⊕ 0^n =x となるので、自動的に満たされる。（n個の0の並びを 0^n と表している）
- ただ、関数f によっては、a=0^n以外にも、 f(x)=f(x⊕a) を満たすaが存在する場合がある。次に、そうした例を見ておこう。

「f(x)=f(x⊕a) となるaを求めよ」の a=0ⁿ以外の解を持つ例

- 右の表でf(x)が定義されているとする。
  このf(x)にはf(x)=f(x⊕a) を満たすaが存在する。
  ただ、f((x)の定義を与えるこの表を眺めているだけでは、f(x)=f(x⊕a) を満たすaを見つけるのは難しいかもしれない。

x	f(x)
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

- ただ、このf(x)が、二つの異なるxの値について同じ値をとることは、確認できる。
  f(x)が同じ値をとる部分を同じ色で表してみた。

x	f(x)
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

「f(x)=f(x⊕a) となるaを求めよ」の a=0^n以外の解を持つ例

- この時、a=110をとれば（各自 x⊕aを計算されたい）、f(x)=f(x⊕a)であることが確認できる。

x	a	x⊕a
000	110	110
001	110	111
010	110	100
011	110	101
100	110	010
101	110	011
110	110	000
111	110	001

x	f(x)
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

- では、どうやってこのaを求めればいいのだろう？

「f(x)=f(x⊕a) となるaを求めよ」の a=0ⁿ以外の解を持つ、もっと簡単な例

- もっと簡単な例も作れる。
  右の表で、f が定義されているとする。ここでも、f(x)は、二つの異なるxの値について同じ値をとることを確認しよう。

x	f(x)
00	10
01	11
10	10
11	11

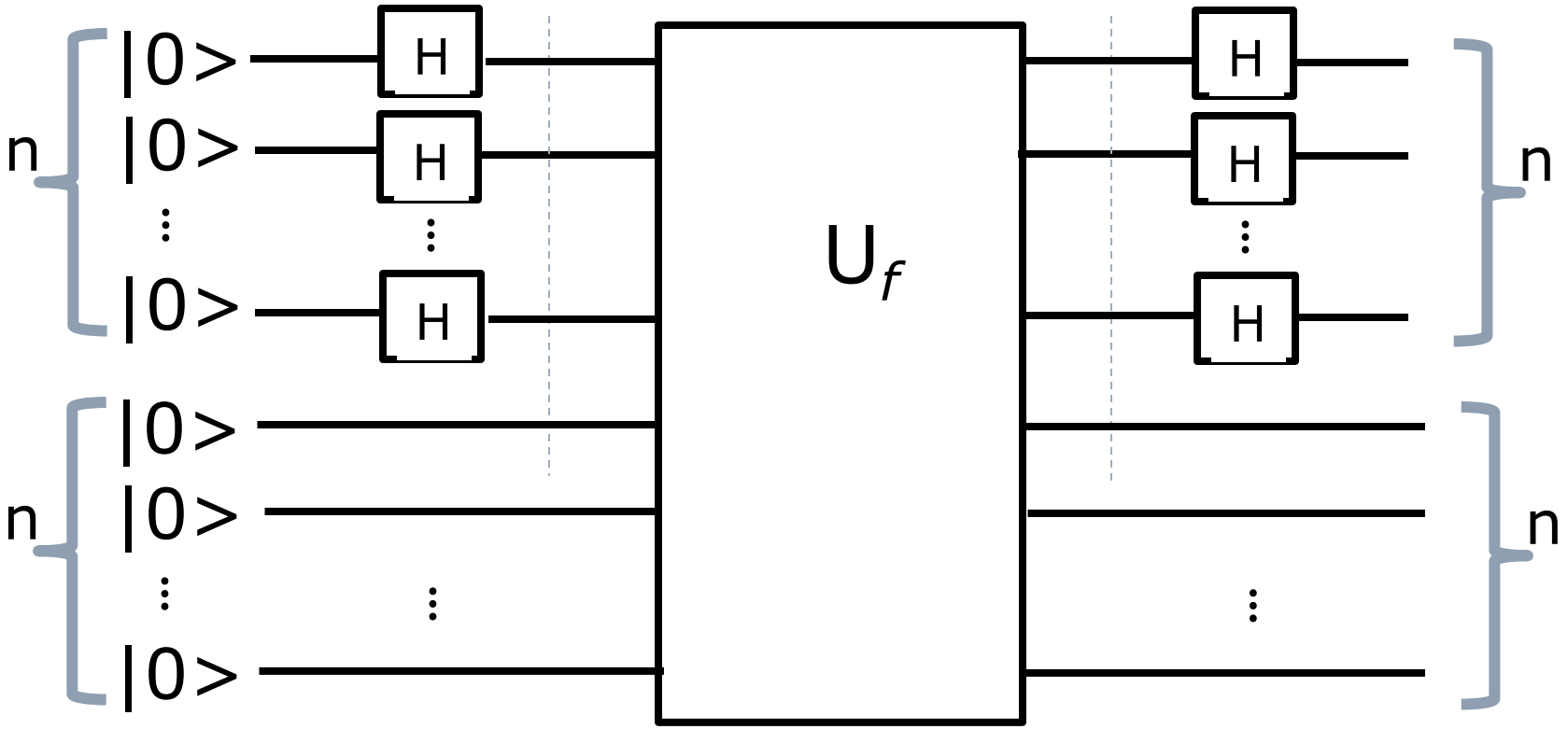
- この時、a=10 とすれば、このf(x)について f(x)=f(x⊕a) なることは、確認できる。

x	a	x⊕a
00	10	10
01	10	11
10	10	00
11	10	01

- では、どうやってこのaを求めればいいのだろう？ xが2-bit, 3-bitで表現される時には、総当たりでも解を見つけることは可能だが、xのビット数が大きくなるにつれて、aを見つけることは急速に難しくなる。

Simonの問題を解く回路
n=3の場合の実行例

Simonの問題を解く量子回路



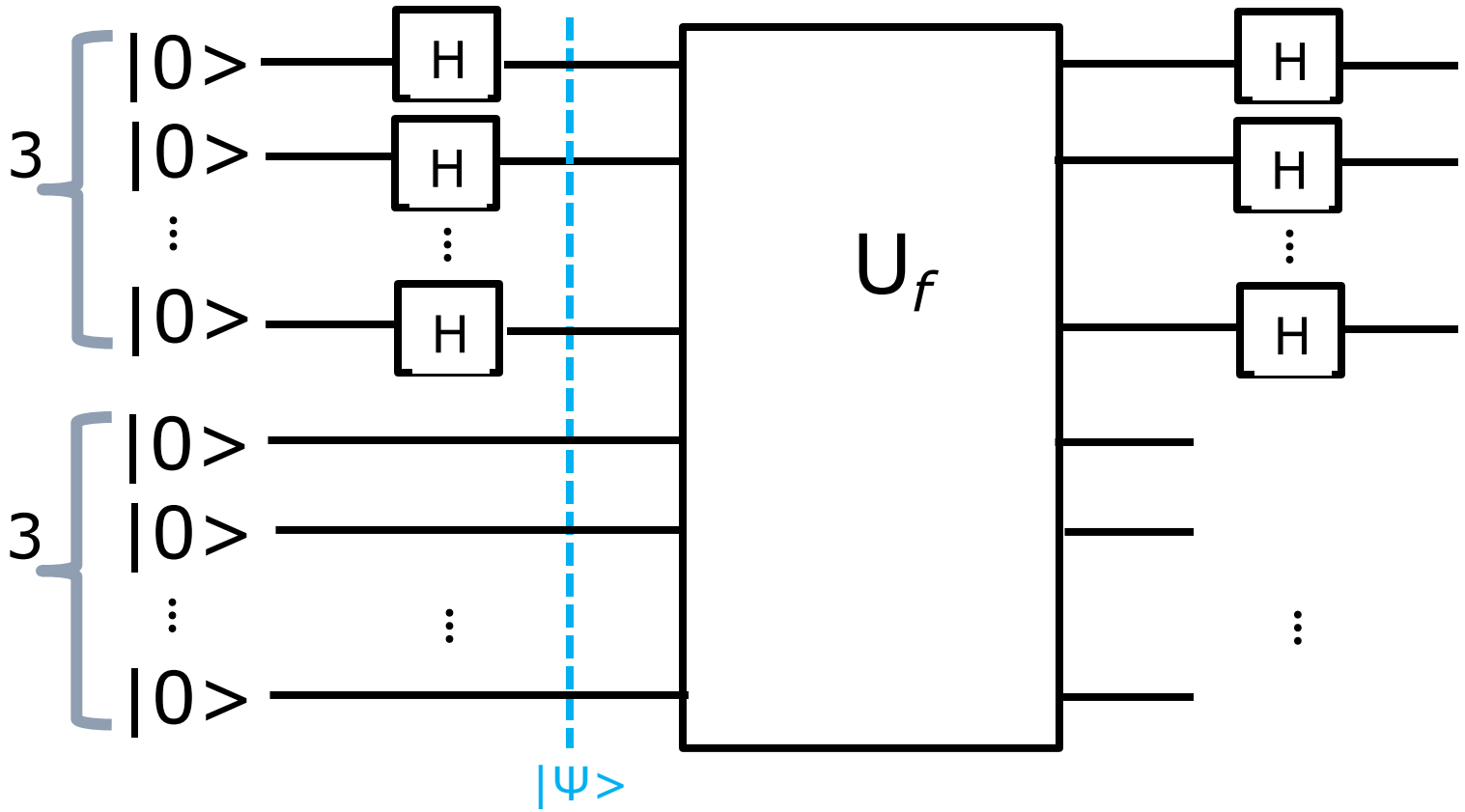
動作例

- 先に見た右の表で与えられるf(x)で
  この回路の各段階の動作を確認してみよう。

f (000) = f (011) = 010
f (100) = f (111) = 110
f (001) = f (010) = 101
f (101) = f (110) = 001
である。

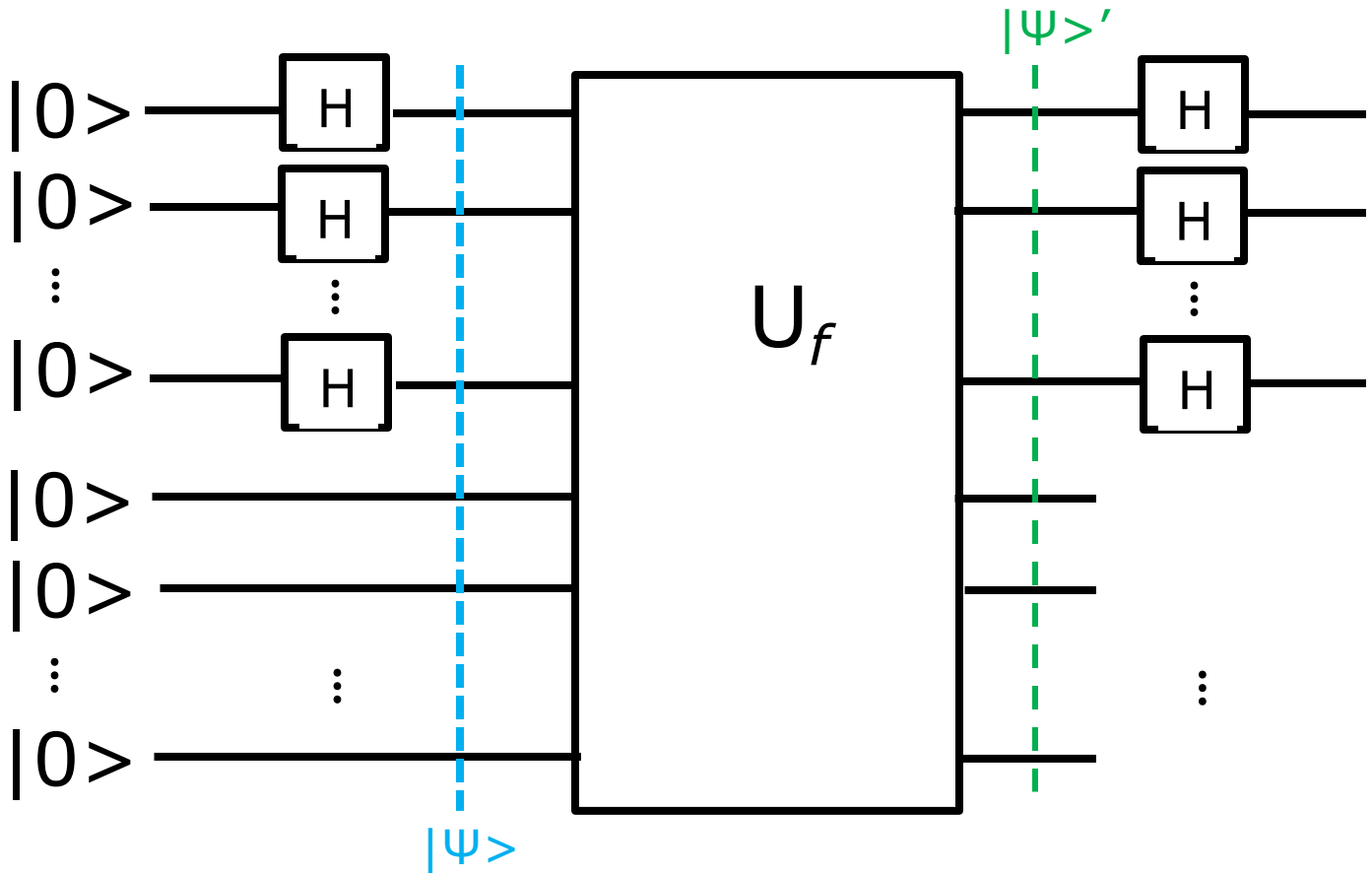
x	f(x)
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

Hadamard で重ね合わせを作る



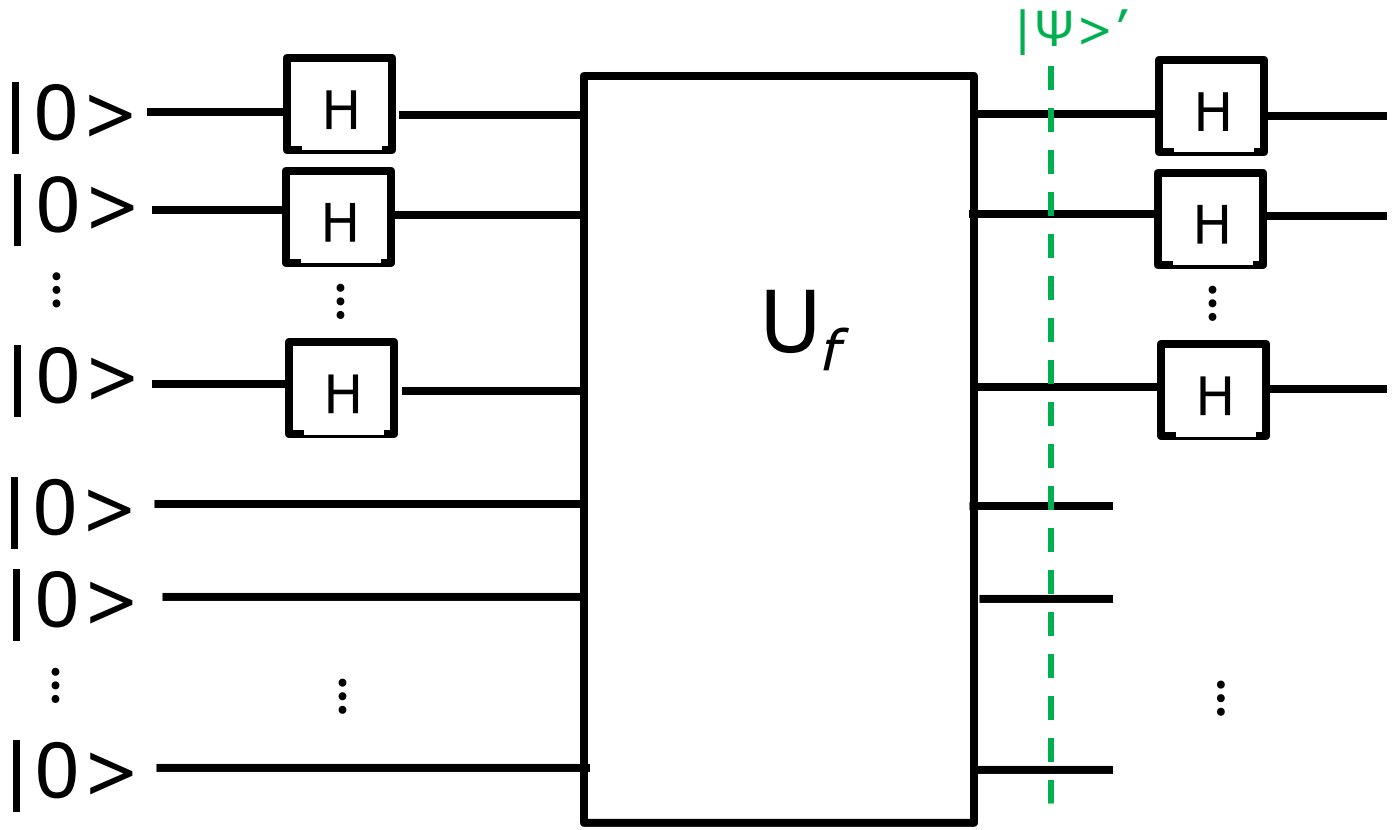
$$\frac{1}{\sqrt{8}}(|000\rangle \otimes |000\rangle + |001\rangle \otimes |000\rangle + |010\rangle \otimes |000\rangle + |011\rangle \otimes |000\rangle + |100\rangle \otimes |000\rangle + |101\rangle \otimes |000\rangle + |110\rangle \otimes |000\rangle + |111\rangle \otimes |000\rangle)$$

その出力|Ψ>を U_fの入力にいれる
結果は|Ψ>′



$$|\Psi\rangle' = \frac{1}{\sqrt{8}}(|000\rangle f(|000\rangle) + |001\rangle f(|001\rangle) + |010\rangle f(|010\rangle) + |011\rangle f(|011\rangle) + |100\rangle f(|100\rangle) + |101\rangle f(|001\rangle) + |110\rangle f(|110\rangle) + |111\rangle f(|111\rangle))$$

出力|Ψ>'を整理する



$$|\Psi\rangle' = \frac{1}{\sqrt{8}}((|000\rangle+|010\rangle)|010\rangle + (|001\rangle+|010\rangle)|101\rangle + (|100\rangle+|111\rangle)|110\rangle + (|101\rangle+110\rangle)|001\rangle\ )$$

二つのx0, x0⊕aと同じ値をとるf(x0)とのペアをまとめる

|Ψ>′=∑ $|x>f(|x>$ の次のような変形は、

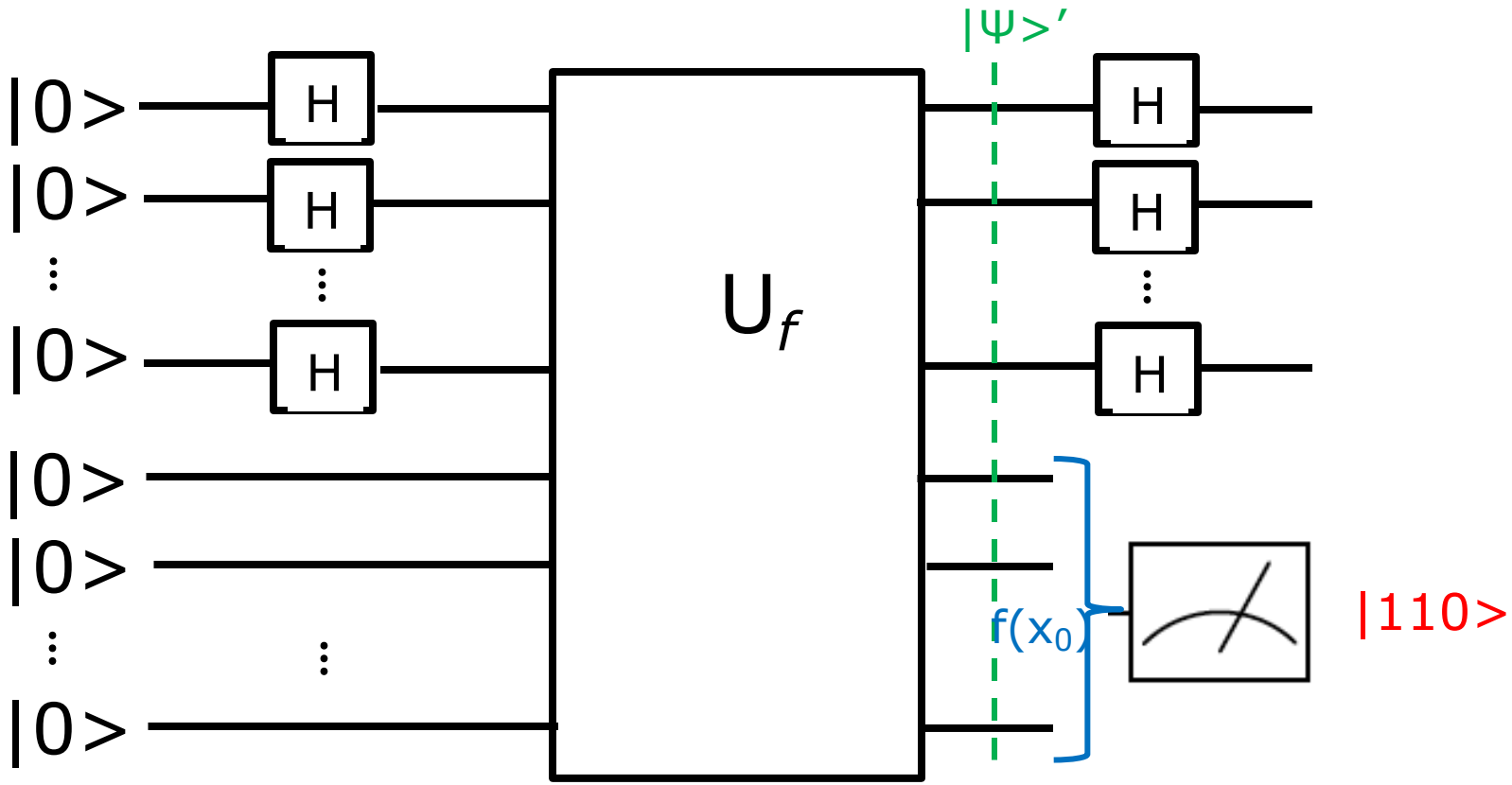
1/√8((|000>+|010>)|010> + (|001>+|010)|101>
+ (|100>+|111>)|110> + (|101>+110>)|001>)

二つのx_0, x_0⊕aと同じ値をとるf(x_0)とのペアをまとめたものである。

f(000)=f(010)=010, f(001)=f(010)=101
f(100)=f(111)=110, f(101)=f(110)=001

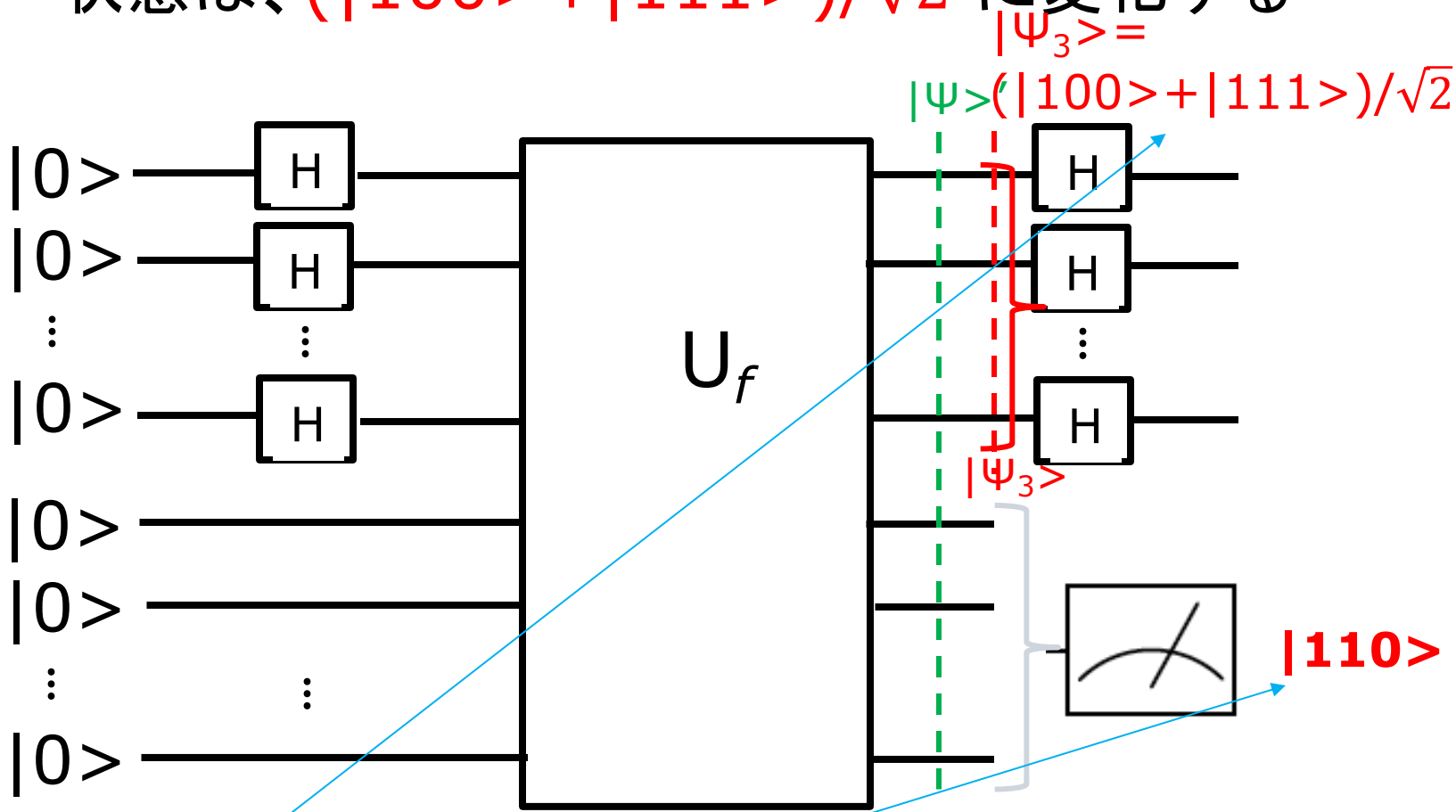
次の例でみるように、下3-qubitが 110 と観測された場合、観測によって|Ψ>′の重ね合わせの状態は失われ、上3-qubitの状態は
$|\Psi_3>=(|100>+|111>)/\sqrt{2}$ になる。
下3-qubitが 010 と観測されたなら、上3-qubitの状態は
$|\Psi_3>=(|000>+|010>)/\sqrt{2}$ になる。

$|\Psi\rangle'$の下3-qubit $f(x_0)$の観測で $f(x_0)=|110\rangle$を観測したとしよう



$$|\Psi\rangle' = \frac{1}{\sqrt{8}}((|000\rangle+|010\rangle)|010\rangle + (|001\rangle+|010\rangle)|101\rangle + (|100\rangle+|111\rangle)\boxed{|110\rangle} + (|101\rangle+110\rangle)|001\rangle\ )$$

$f(x_0)=|110\rangle$の観測によってデータ･レジスターの状態は、$(|100\rangle+|111\rangle)/\sqrt{2}$ に変化する



$|\Psi\rangle' =$

$\frac{1}{\sqrt{8}}((|000\rangle+|010\rangle)|010\rangle + (|001\rangle+|010\rangle)|101\rangle$

$+ (|100\rangle+|111\rangle)|110\rangle + (|101\rangle+110\rangle)|001\rangle\ )$

あるx$_0$についてのf(x$_0$)の観測は、f(x$_0$)の「原像」の重ね合わせを作り出す

f(p)=f(x$_0$) を満たすようなpをf(x$_0$)の「原像」という。

今回の例では、

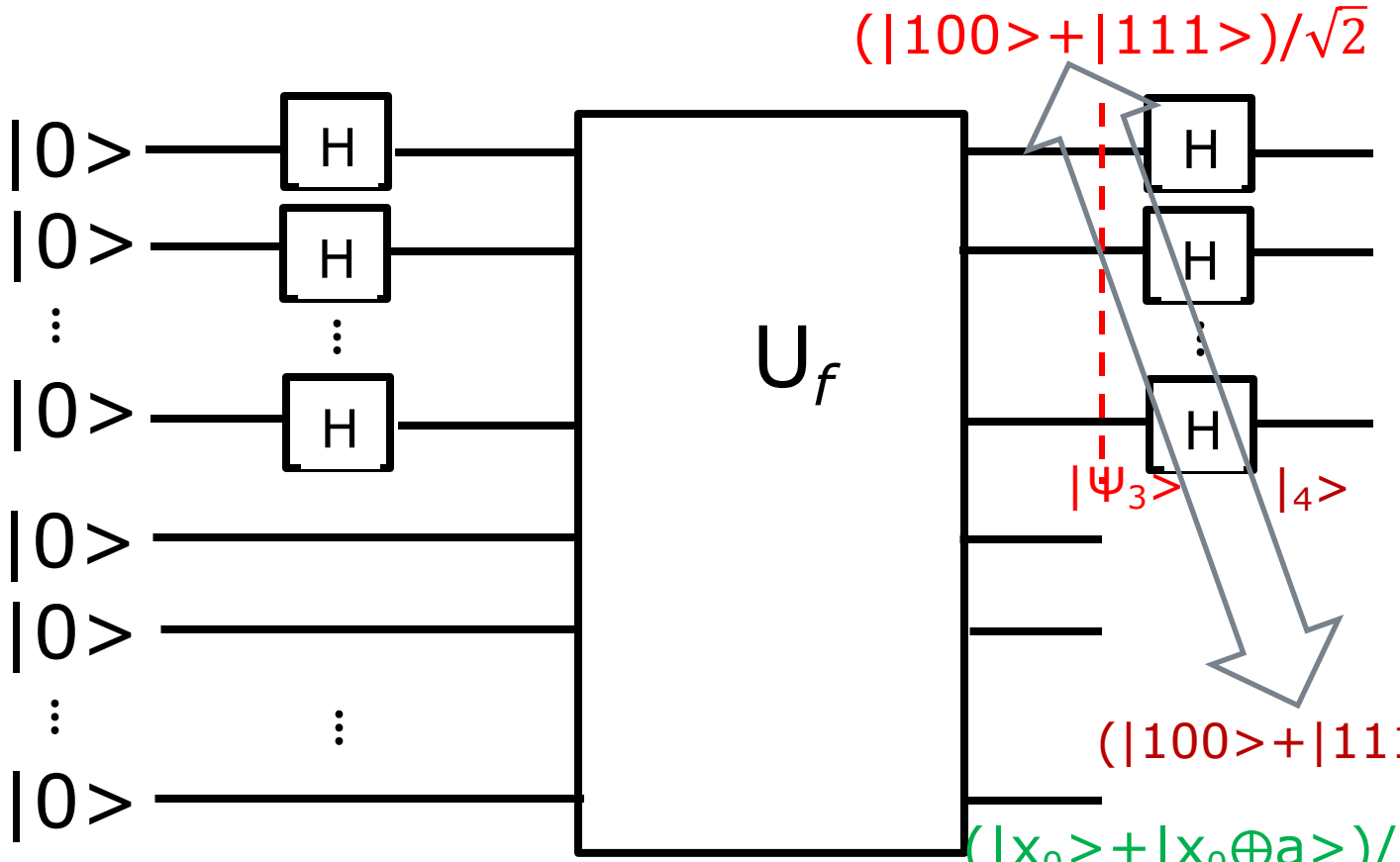
f(x$_0$)=f(x$_0$)

f(x$_0$⊕a)=f(x$_0$)

であるので、f(x$_0$)の「原像」は二つあって、x$_0$, x$_0$⊕a である。

状態|Ψ>′=∑ $|x>f(|x>$ のターゲット・レジスター $f(|x>$ 部分の観測は、重ね合わせが破れてシグマがなくなり、あるx$_0$についてのf(x$_0$)を観測することになるのだが、この時、データ・レジスターの状態は、このf(x$_0$)についての「原像」の重ね合わせの状態に変わる。

$f(x_0)$の二つの「原像」x_0と$x_0 \oplus a$の重ね合わせが、データ・レジスターに現れる。



$$\frac{1}{\sqrt{8}}((|000>+|010>)|010> + (|001>+|010>)|101> + (|100>+|111>)|110> + (|101>+110>)|001>)$$

$|\Psi_3\rangle$をn個のアダマールに通し、その結果の$|\Psi_4\rangle$を計算する

この例では、$|\Psi_3\rangle=(|100\rangle+|111\rangle)/\sqrt{2}$ に対して、

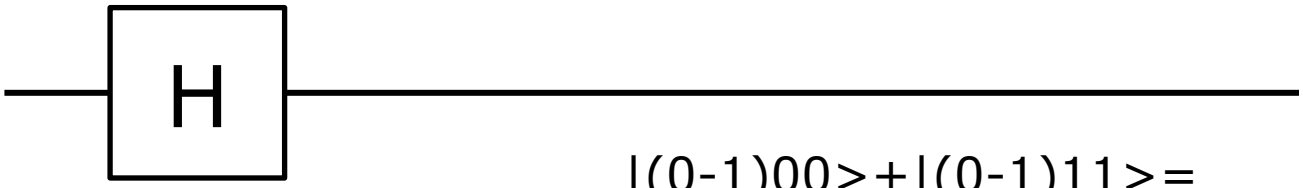
$H^{\otimes 3}|\Psi_3\rangle$を計算する。

一般に、 $H^{\otimes n}\ |x_0\rangle^n \ = \ \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{y}\cdot\mathbf{x_0}}\ |y\rangle^n$ である。

ここでは、筆算でこの計算をしてみよう。

$H^{\otimes 3}(|100\rangle+|111\rangle)/\sqrt{2}$ の計算

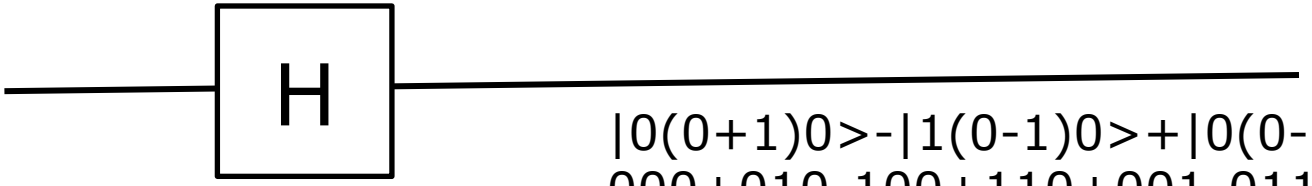
第一qubitにHを適用



正規化の因数と、
途中からケット記号も
省略している

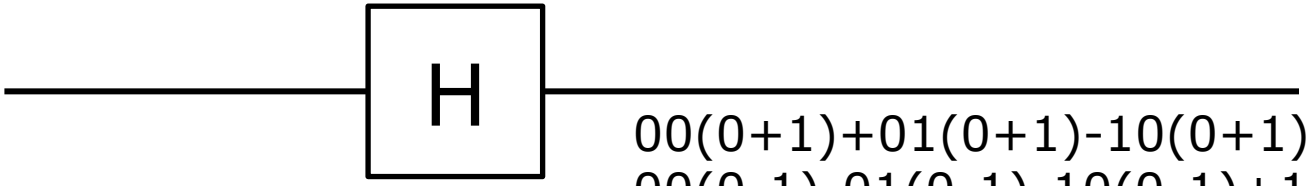
|(0-1)00>+|(0-1)11>=
|000>-|100>+|011>-|111>

第二qubitにHを適用



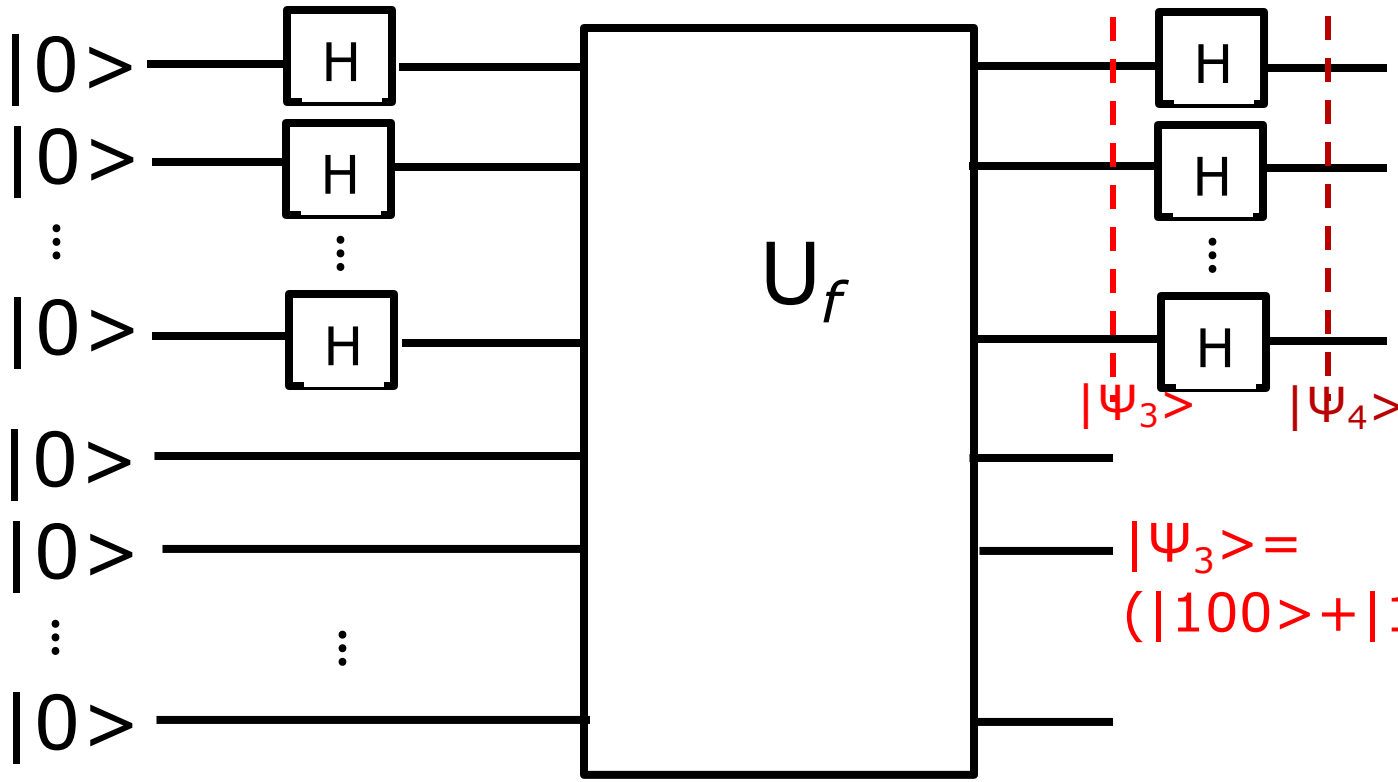
|0(0+1)0>-|1(0-1)0>+|0(0-1)1>-|1(0-1)1>=
000+010-100+110+001-011-101+111

第三qubitにHを適用



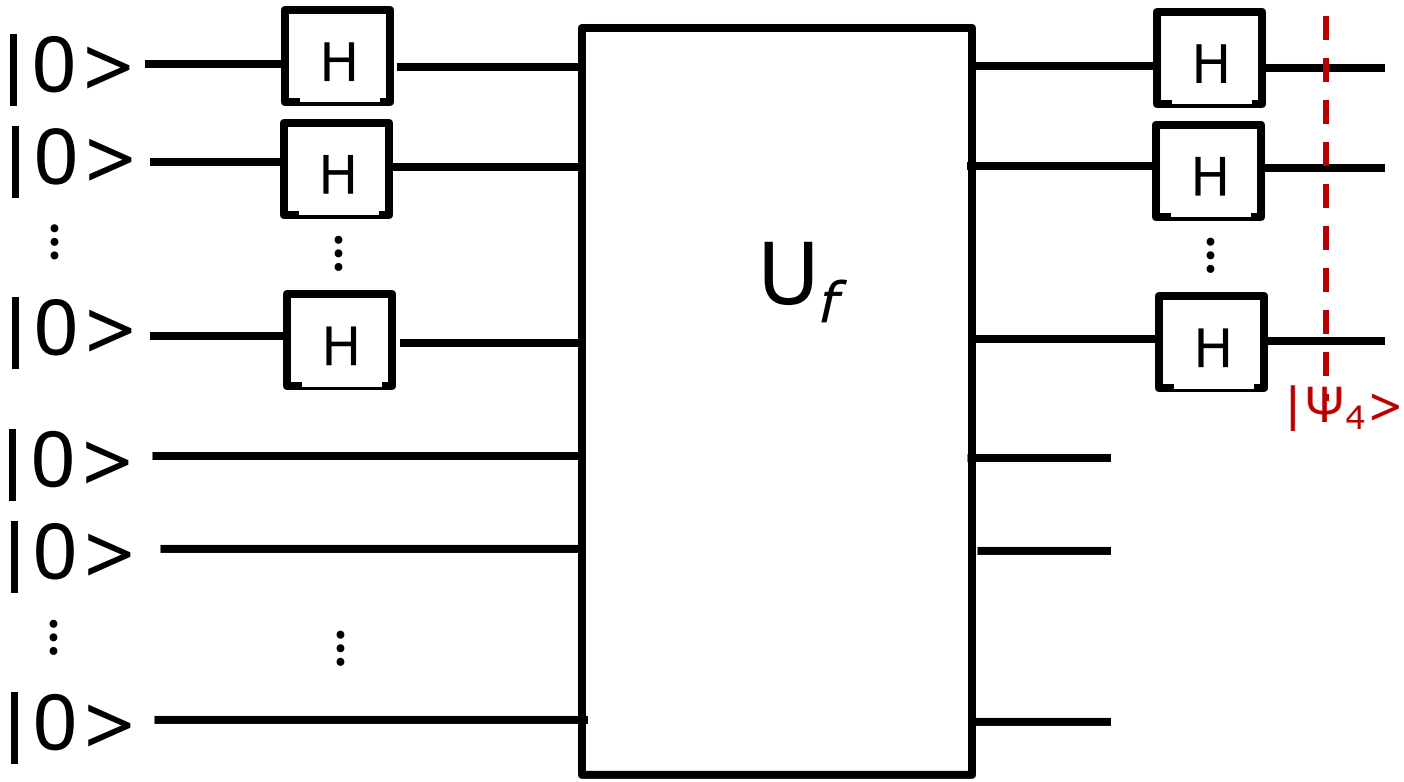
00(0+1)+01(0+1)-10(0+1)+11(0+1)+
00(0-1)-01(0-1)-10(0-1)+11(0-1)
= 000+001+010+011-100-101+110+111+
000-001-010+011-100+101+110-111=
2(000+011-100-110)

$|\Psi_3\rangle$をn個のアダマールに通し、その結果の$|\Psi_4\rangle$を計算する



$|\Psi_4\rangle=(|000\rangle+|011\rangle-|100\rangle-|110\rangle)/2$

$|\Psi_4\rangle$を観測する



$|\Psi_4\rangle=(|000\rangle+|011\rangle-|100\rangle-|110\rangle)/2$

$|\Psi_4\rangle$を何度か観測して a を求める

$|\Psi_4\rangle=(|000\rangle+|011\rangle-|100\rangle-|111\rangle)/2$ なので、
$|\Psi_4\rangle$の観測で、

y=000, 011, 100, 111 のいずれかを得る。

これらのyは、$f(x)=f(x\oplus a)$なるaについて $y\cdot a=0$を満たす（この理由は、後述）ので、$a=a_0a_1a_2$とビットで表されるとすると

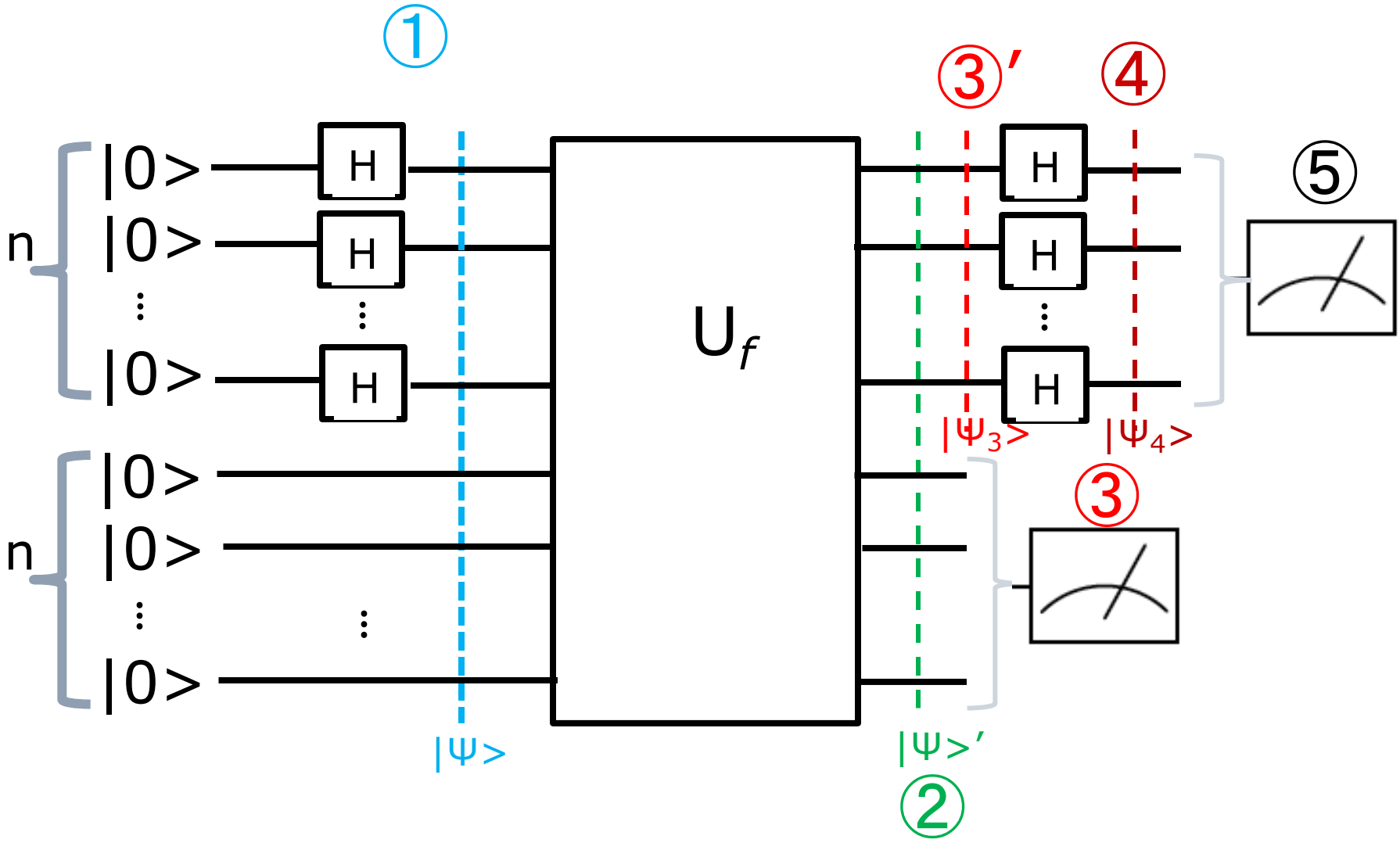
y=000の時、$0\times a_0\oplus 0\times a_1\oplus 0\times a_2=0$　情報なし
y=011の時、$0\times a_0\oplus 1\times a_1\oplus 1\times a_2=0$　$a_1\oplus a_2=0$
y=100の時、$1\times a_0\oplus 0\times a_1\oplus 0\times a_2=0$　$a_0=0$
y=111の時、$1\times a_0\oplus 1\times a_1\oplus 1\times a_2=0$　$a_0\oplus a_1\oplus a_2=0$

これから、a=000 または a=011

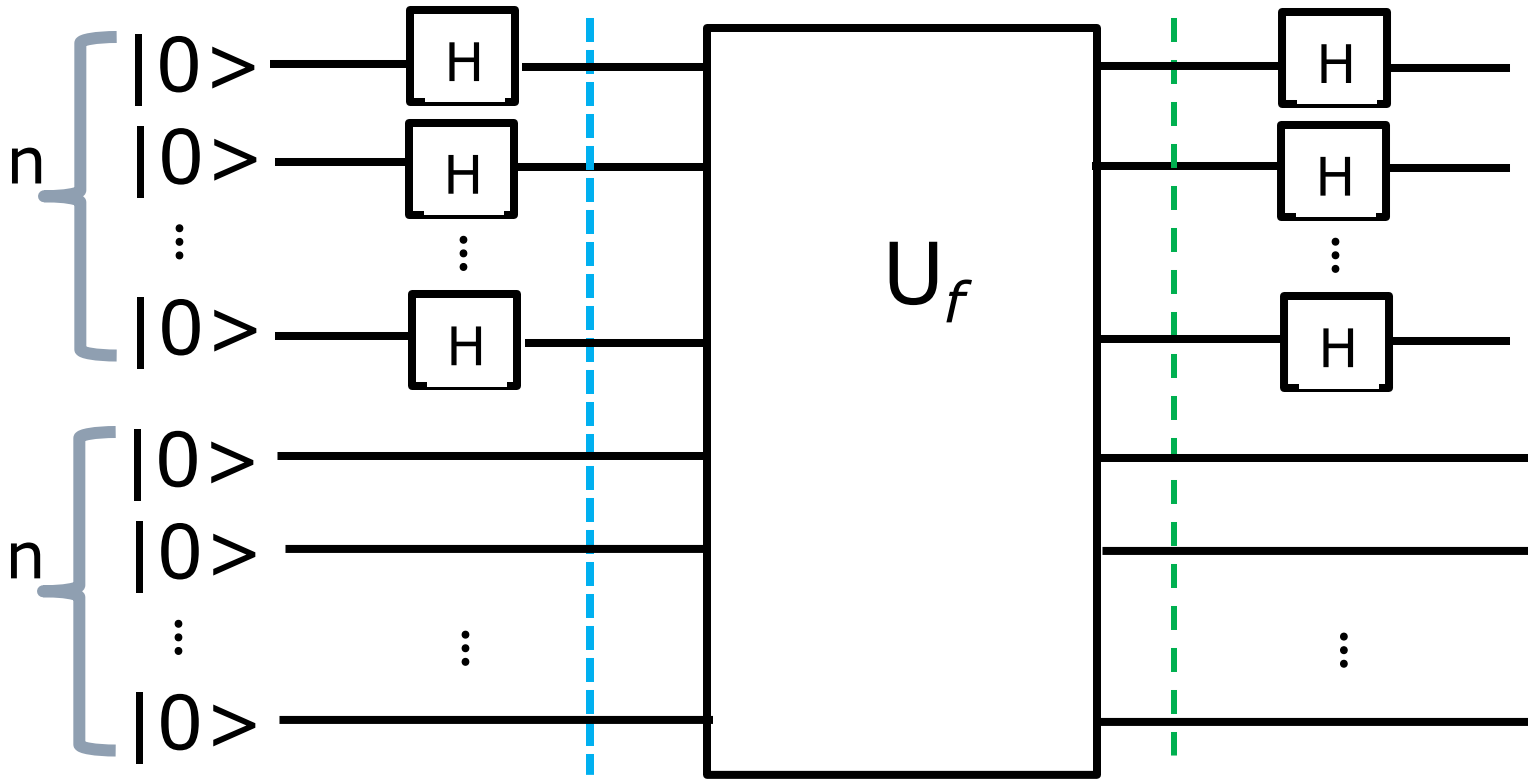
a=000は、条件に反するので、a=011である。

Simonの問題
一般的な解法

一連の流れ



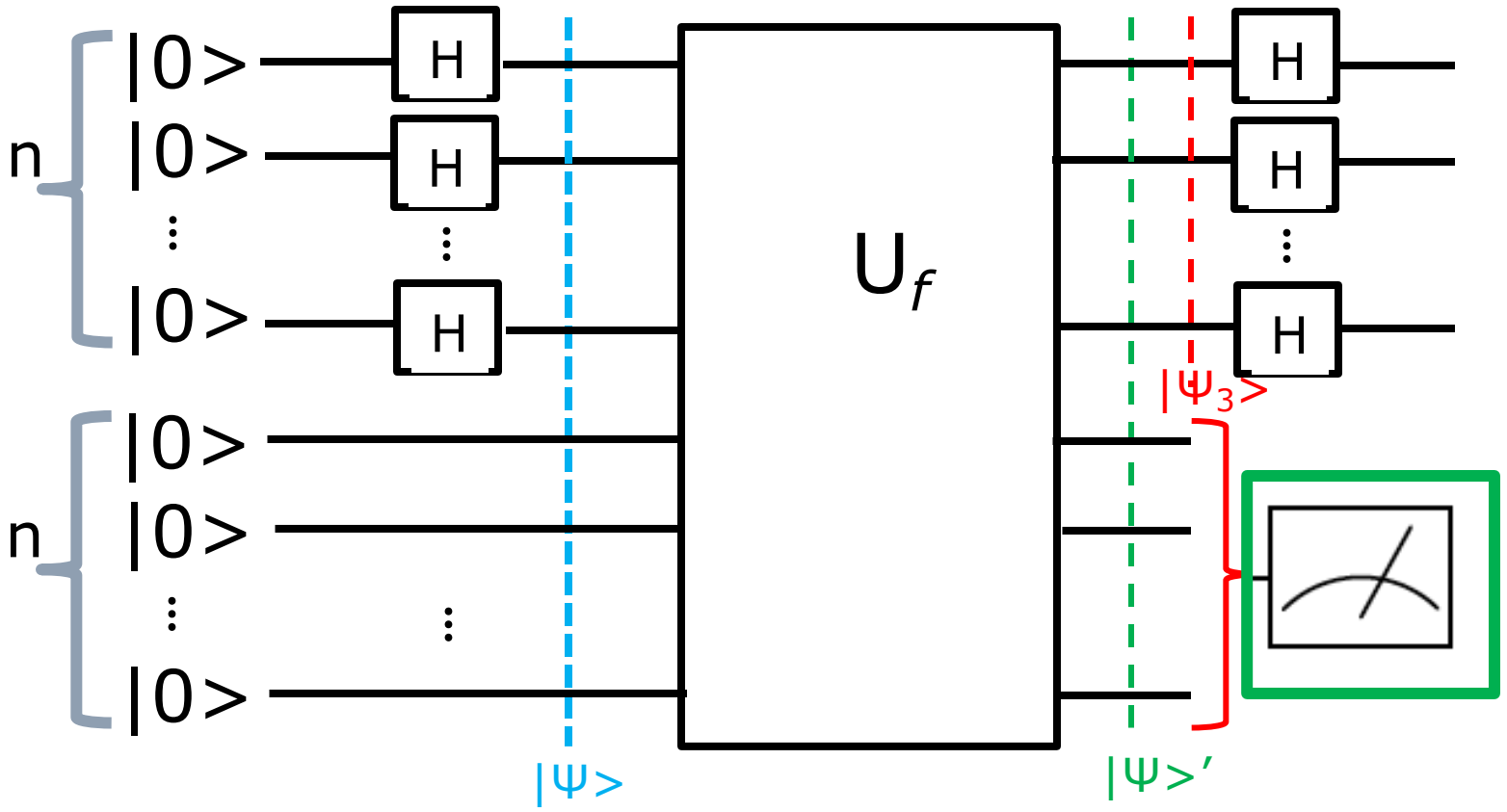
各段階での |Ψ>, |Ψ>′の状態は、
すでに見たように、次のように計算できる



$$|\Psi\rangle = \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} (|x\rangle \otimes |0^n\rangle)$$

$$|\Psi\rangle' = \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} (|x\rangle \otimes |f(x)\rangle)$$

ターゲット・レジスターを観測する。観測によって変化したデータ・レジスターの状態を$|\Psi_3\rangle$とする



f(x)=f(x⊕a)を使って、|Ψ>′を変形する

$$|\Psi\rangle' = \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} (|x\rangle \otimes |f(x)\rangle)$$ だが

f(x)が xの異なる二つの値 x と x⊕a で、同じ値 f(x)=f(x⊕a) をとるのだから、|Ψ>′は次のように変形できる。

$$\left(\frac{1}{\sqrt{2}}\right)^n \sum_{x=0}^{2^n-1} |x\rangle^n \; |f(x)\rangle^n \quad = \quad \left(\frac{1}{\sqrt{2}}\right)^n \sum_{x\in R} \left(|x\rangle^n \; + \; |x\oplus a\rangle^n \right) \; |f(x)\rangle^n$$

$$= \left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{x\in R} \left(\frac{|x\rangle^n \; + \; |x\oplus a\rangle^n}{\sqrt{2}} \right) \; |f(x)\rangle^n$$

ここで、Rは、xが取りうる値の半分を走り、残りの半分はx⊕a が走る。

$$\left.\begin{matrix} x \\ x\oplus a \end{matrix}\right\}_{x\in R} \quad \overset{f}{\longmapsto} \quad f(x)$$

観測による
データ･レジスターの状態変化

$$|\Psi\rangle' = \left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{x \in R} \left(\frac{|x\rangle^n + |x \oplus a\rangle^n}{\sqrt{2}}\right) |f(x)\rangle^n$$

のf(x)部分の観測で、|Ψ>'の重ね合わせの状態は失われて、
あるx_0について、

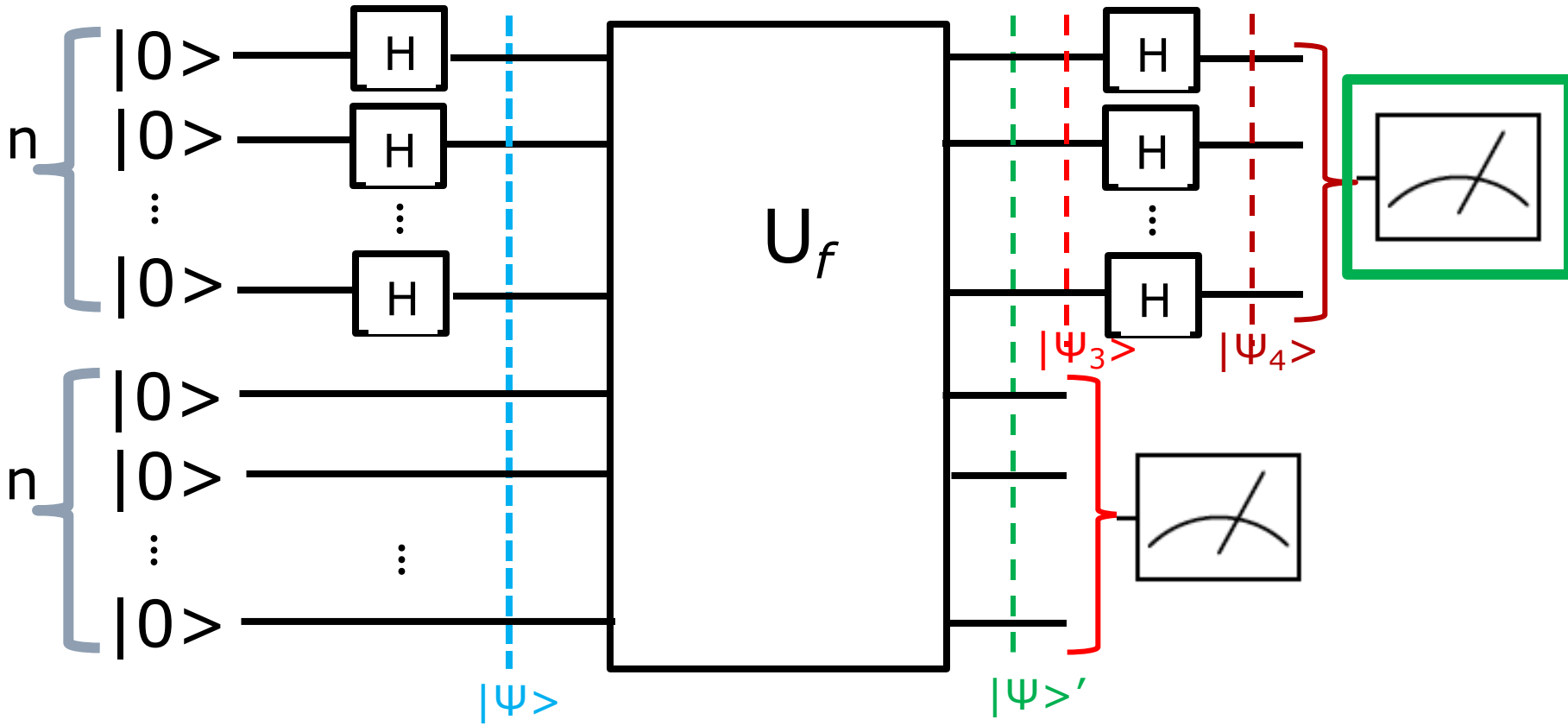
$$\left(\frac{|x_0\rangle^n + |x_0 \oplus a\rangle^n}{\sqrt{2}}\right) |f(x_0)\rangle^n$$

の状態に変化する。

このことは、データ･レジスターの状態$|\Psi_3\rangle$は、次のようになることを意味する。

$$|\Psi_3\rangle = \left(\frac{|x_0\rangle^n + |x_0 \oplus a\rangle^n}{\sqrt{2}}\right).$$

$|\Psi_3\rangle$をn個のアダマールに通し、その結果の$|\Psi_4\rangle$を観測する



$|\Psi_3\rangle$をn個のアダマールに通し、その結果の$|\Psi_4\rangle$を計算する

$$H^{\otimes n}|\Psi_3\rangle = H^{\otimes n}\left(\frac{|x_0\rangle^n + |x_0 \oplus a\rangle^n}{\sqrt{2}}\right) = \frac{H^{\otimes n}|x_0\rangle^n + H^{\otimes n}|x_0 \oplus a\rangle^n}{\sqrt{2}}$$

$$H^{\otimes n}|x_0\rangle^n = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{y}\cdot\mathbf{x_0}} |y\rangle^n$$

$$H^{\otimes n}|x_0 \oplus a\rangle^n = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{y}\cdot(\mathbf{x_0}\oplus\mathbf{a})} |y\rangle^n$$

$$(-1)^{\mathbf{y}\cdot(\mathbf{x_0}\oplus\mathbf{a})} = (-1)^{\mathbf{y}\cdot\mathbf{x_0}}(-1)^{\mathbf{y}\cdot\mathbf{a}}$$ だから、

$$|\Psi_4\rangle = \frac{H^{\otimes n}\,|x_0\rangle^n + H^{\otimes n}\,|x_0\oplus a\rangle^n}{\sqrt{2}}$$

$$= \frac{\left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1}(-1)^{\mathbf{y}\cdot\mathbf{x_0}}\left(1+(-1)^{\mathbf{y}\cdot\mathbf{a}}\right)|y\rangle^n}{\sqrt{2}}$$

$$= \left(\frac{1}{\sqrt{2}}\right)^{n+1}\sum_{y=0}^{2^n-1}(-1)^{\mathbf{y}\cdot\mathbf{x_0}}\left(1+(-1)^{\mathbf{y}\cdot\mathbf{a}}\right)|y\rangle^n .$$

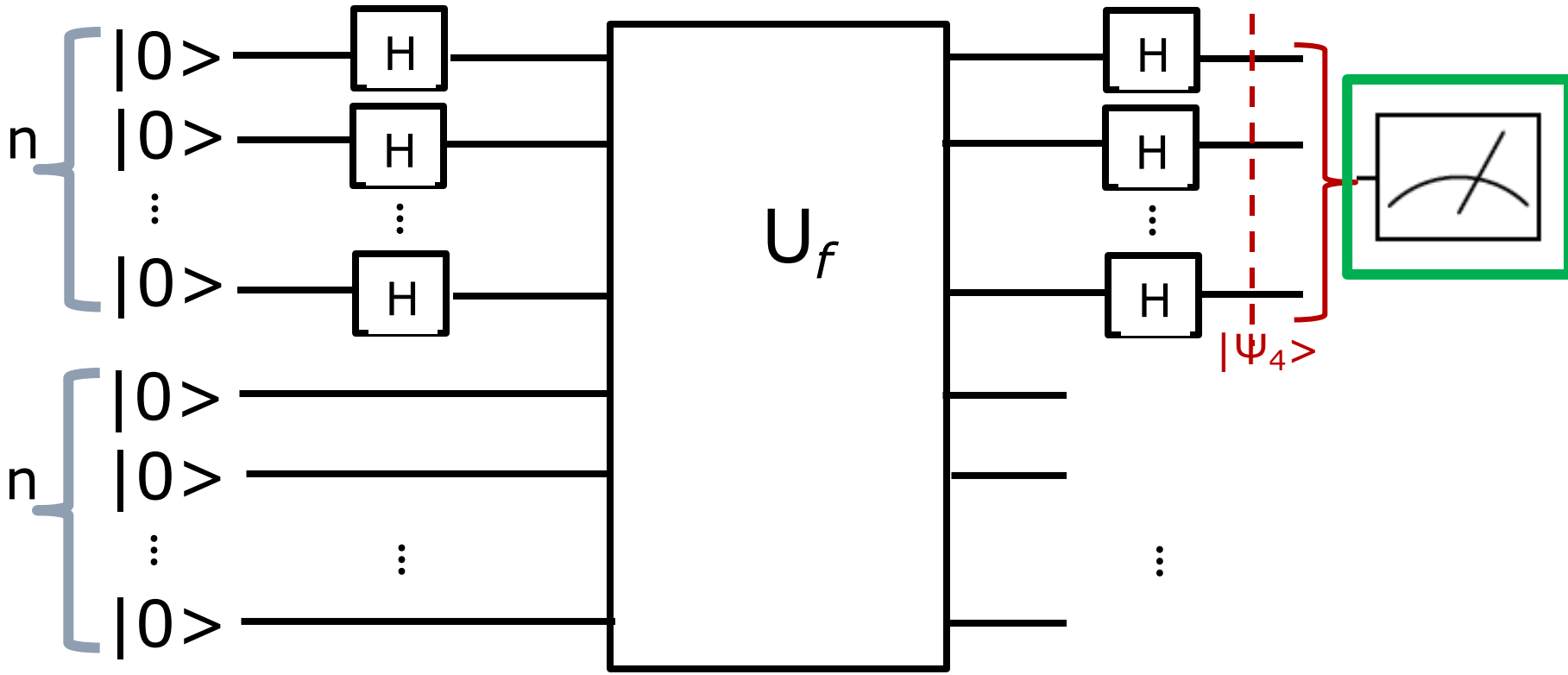
$$|\Psi_4\rangle = \left(\frac{1}{\sqrt{2}}\right)^{n+1} \sum_{y=0}^{2^n-1} (-1)^{\mathbf{y}\cdot\mathbf{x_0}} \left(1 + (-1)^{\mathbf{y}\cdot\mathbf{a}}\right) |y\rangle^n$$

$$1 + (-1)^{\mathbf{y}\cdot\mathbf{a}} = \begin{cases} 0, & \text{if } \mathbf{y}\cdot\mathbf{a} = 1 \pmod 2 \\ 2, & \text{if } \mathbf{y}\cdot\mathbf{a} = 0 \pmod 2 \end{cases}$$

だから、

$$|\Psi_4\rangle = \left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{\substack{\mathbf{y}\cdot\mathbf{a}=0 \\ (\mathrm{mod}\ 2)}} (-1)^{\mathbf{y}\cdot\mathbf{x_0}} |y\rangle^n$$

$|\Psi_4\rangle$を観測する



$|\Psi_4\rangle$を観測する

$$|\Psi_4\rangle = \left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{\substack{\mathbf{y}\cdot\mathbf{a}=0 \\ (\mathrm{mod}\ 2)}} (-1)^{\mathbf{y}\cdot\mathbf{x_0}} \quad |y\rangle^n$$

を観測すると、$|\Psi_4\rangle$ は、状態 $|y_0\rangle$ に変化する。
ただし、このy_0について、$y_0 \cdot a=0$ である。

試行を繰り返してaを求める

- 一回の試行で得られるのは $y_0 \cdot a = 0$ なる y_0であってaではない。
- ただ、何度か試行を繰り返せば、異なるyの値について、次のような一連の式を得ることができる。ここには、aについての情報が含まれている。

$$
\begin{aligned}
y_1 \cdot a &= 0 \\
y_2 \cdot a &= 0 \\
y_3 \cdot a &= 0 \\
&\vdots \\
y_{n-1} \cdot a &= 0
\end{aligned}
$$

- yもaも n-bitなので、n-1個のこうした式があれば、aを決定することができる。

Part II

Shorのアルゴリズム

Shorの素因数分解アルゴリズム

関数の「周期」から、素因数を求める

いくつかの予備準備
フェルマーの小定理

pが素数の時、pと互いに素な整数aについて、次の式が成り立つ。

$a^p \equiv a \pmod{p}$　　この式は、両辺をaで割って

$a^{p-1} \equiv 1 \pmod{p}$　と同じである。

$2^3 = 8 = 3 \times 2 + 2 \equiv 2 \pmod{3}$

$2^5 = 32 = 5 \times 6 + 2 \equiv 2 \pmod{5}$

$2^7 = 128 = 7 \times 18 + 2 \equiv 2 \pmod{7}$

$2^{11} = 2048 = 11 \times 186 + 2 \equiv 2 \pmod{11}$

対偶をとれば、

aとnが互いに素で、$a^{n-1} \not\equiv 1 \pmod{n}$ ならば、nは素数ではない

ことがわかる

フェルマー・テスト　確率的素数判定

1. n以下の自然数aを一つ選ぶ
2. aとnが互いに素でないなら、nは素数でない
3. $a^{n-1} \not\equiv 1 \pmod{n}$ ならば、nは素数でない
4. そうでないなら、nは素数である確率がある

この時、別のaを選んで、この処理を繰り返す。

十分な回数だけ a を取り替えて繰りかえしても、nがこのテストをパスするなら、その数は実際に素数である可能性が高い。

shorのアルゴリズムも、同じような確率的方法をとる。

関数a^xの周期（あるいは、群の位数）

$a^r \equiv 1 \pmod{N}$ なるrが存在する時、この最小のrを（Nについての）aの周期（あるいは位数）という。

例えば、$N = 15, a = 7$ とすると、周期は4である。

$7^0 = 1 \pmod{15}$
$7^1 = 7 \pmod{15}$
$7^2 = 4 \pmod{15}$
$7^3 = 13 \pmod{15}$
$7^4 = 1 \pmod{15}$
$7^5 = 7 \pmod{15}$
$7^6 = 4 \pmod{15}$
$7^7 = 13 \pmod{15}$
$7^8 = 1 \pmod{15}$
$7^9 = 7 \pmod{15}$

周期（位数）を使った素因数分解

xの周期をrとする。rが偶数の時、次のようにかける。

$$
\begin{aligned}
x^r &\equiv 1 \mod N &\iff \\
(x^{r/2})^2 &\equiv 1 \mod N &\iff \\
(x^{r/2}+1)(x^{r/2}-1) &\equiv 0 \mod N &\iff \\
(x^{r/2}+1)(x^{r/2}-1) &= kN \text{ for some } k.
\end{aligned}
$$

すなわち、$x^{r/2}$-1 と$x^{r/2}$ +1は、Nと約数を共有している。

ただ、$x^{r/2}$-1 ≡ 0 mod Nにはならない。rは x^r-1 ≡ 0 mod Nとなる「最小」の r だから。Nは、$x^{r/2}$-1 を割らない。N∤ $x^{r/2}$-1

もし、$x^{r/2}$ +1 ≢ 0 mod N で、Nが $x^{r/2}$ +1も割らないなら gcd($x^{r/2}$ +1,N)かgcd($x^{r/2}$ −1,N)のいずれかが、Nを割ることになる。

周期（位数）を使った素因数分解

例えば、先の、N=15, a=7, r=4の時、

$\gcd(a^{r/2}+1,N)=\gcd(7^2+1,15)=\gcd(50,15)=5$
$\gcd(a^{r/2}-1,N)=\gcd(7^2-1,15)=\gcd(48,15)=3$

3,5は、15を割る。

周期を使ったショアの素因数分解アルゴリズム

1. a <N なるaをランダムに選ぶ。
2. aとNの最大公約数gcd(a,N)を計算する。（ユークリッドの互除法を用いる） gcd(a,N)が1でなかったら、その数がNの約数である。
3. f(x)＝ a^x mod N なる関数の周期をr とする。$a^r \equiv 1 \bmod N$ すなわち、$a^r - 1$は、Nで割れる。N| $a^r - 1$
（この周期rを求めるのに、量子コンピュータを利用する）
4. rが奇数なら、1.に戻る。
5. rが偶数なら $a^r - 1 = (a^{r/2}+1)(a^{r/2}-1)$と表せる。
6. もし、$a^{r/2} = -1 \bmod N$ なら 1.に戻る。
7. そのいずれでもないなら、gcd($a^{r/2}+1$,N)かgcd($a^{r/2}-1$,N)のいずれかが、Nを割る。

どのように関数の「周期」を求めるのか？

概略編

Period Finding

- 周期的な関数 f を計算するブラックボックスが与えられているとしよう。周期的な関数とは、次のような関数のことだ。
  $f(x)=f(y)$ となるのは、$x \equiv y \pmod r$の時のみ
  fが周期的でその周期をrとすれば、
  $f(x_0)=f(x_0+r)=f(x_0+2r)=f(x_0+3r)\ldots,=f(x_0+(N/r-1)r)$
  となる。関数 $\sin x$ も $\cos x$ も、mod 2π で周期関数だ。
- 「周期探し」の目標は、f（そのブラックボックス）が与えられた時、周期 r を見つけることである。
- 古典的には、この f に何度も値を代入して繰り返しを見つければいいのだが、f が、n ビットの入力を受け付けるのなら、最悪 2^n 回の問い合わせを、このブラックボックスに対して行うことになる。他にもやり方はあるかもしれないが、古典的には、指数関数的な時間が必要になる。

Quantum Period Finding

- 量子コンピュータでは、n個のqubit で、$N=2^n$個の入力の組み合わせの重ね合わせの状態が表現できる。ここで、フーリエ変換の周期・周波数と線形シフトの性質を利用する。
- まず、最初に、線形シフトを含む重ね合わせの状態にアクセスする。そして、フーリエ変換で、線形シフトを除去する。

- 次に見るように、ShorのアルゴリズムのQuantum Period Findingの基本的流れは、先に見たSimonのアルゴリズムの基本的部分を踏襲している。
- 違うのは、$f(x_0)$の「原像」の重ね合わせの状態を得た後でアダマール変換 $H^{\otimes n}$ ではなく、逆フーリエ変換$QFT^{\dagger}$を利用するところである。

Quantum Period Findingの流れ

- 最初に重ね合わせの状態をQFTで準備する。これは、$H^{\otimes n}$ を使っても同じである。

$$|0\rangle\,|0\rangle \xrightarrow{QFT_N} \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle\,|0\rangle \qquad \text{N=}2^{n}$$

- f を計算するブラックボックス U_fを適用する。

$$\frac{|1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle\,|0\rangle \xrightarrow{U_f} \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle\,|f(x)\rangle$$

- ここで、|f(x)>を測定する。
  |f(x)>は、ある値 $f(x_0)$として観測されるのだが、同時に、
  |x>も、$f(x_0)$の元のイメージ「原像」に崩壊する。fは、周期的だから、その値は、$\{x_0, x_0+r, x_0+2r, \ldots, x_0+(N/r-1)r\}$ の重ね合わせになる。

観測によって、$f(x_0)$の「原像」の重ね合わせが第一レジスターに現れる

- |f>の観測は、次のような変化を引き起こしたことになる。

$$\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle\otimes|f(x)\rangle \xrightarrow{\text{measure}|f\rangle} \sqrt{\frac{r}{N}}\sum_{i=0}^{N/r-1}|ir+x_0\rangle\,|f(x_0)\rangle$$

- 第一レジスターには、関数fの周期ｒと同じ周期の重ね合わせの情報が含まれている。
- ここまでは、Simonのアルゴリズムと考え方は、全く同じである。
  ただ、このアルゴリズムを繰り返し、第一レジスターを測定しても、異なるx0についての情報が得られるだけである。

Phase Estimation

量子フーリエ変換は、Phase Estimationと呼ばれる一般的な手続きの重要なキーとなる。また、Phase Estimationは、数多くの量子アルゴリズムで重要な役割を果たす。

Phase Estimation

- 固有ベクトル|u>, 固有値$e^{2\pi i\varphi}$を持つユニタリ演算子Uを考えよう。ただし、φの値は未知とする。

$$U|u\rangle = e^{2\pi i\phi}|u\rangle$$

- Phase Estimation アルゴリズムの目標は、こうしたφを見つけること。
- 評価を実行するために、状態|u>を準備し、コントロール-U^{2^j}（jは非負の整数）演算を実行できる、ブラックボックスが利用できるものとする。

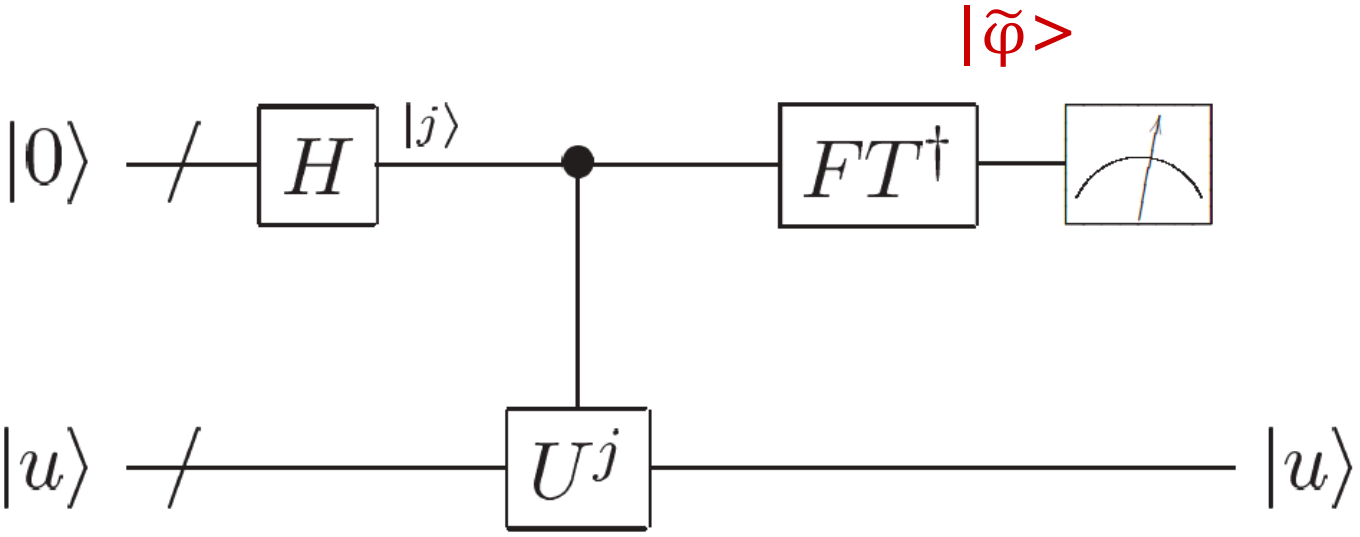
Phase Estimator

概観

Phase estimator 回路

- Quantum Phase Estimation 手続きは、二つのレジスターを使う。
- 第一のレジスターは、初期状態が|0>のt個のqubitを含んでいる。tをどのように選べばいいかは、どの程度の精度で φを評価するか、また、どの程度の確率でこの手続きが成功するかに依存する。
- 第二のレジスターは、|u>の状態で始まり、|u>を保存するにに、必要なだけのqubitからなる。

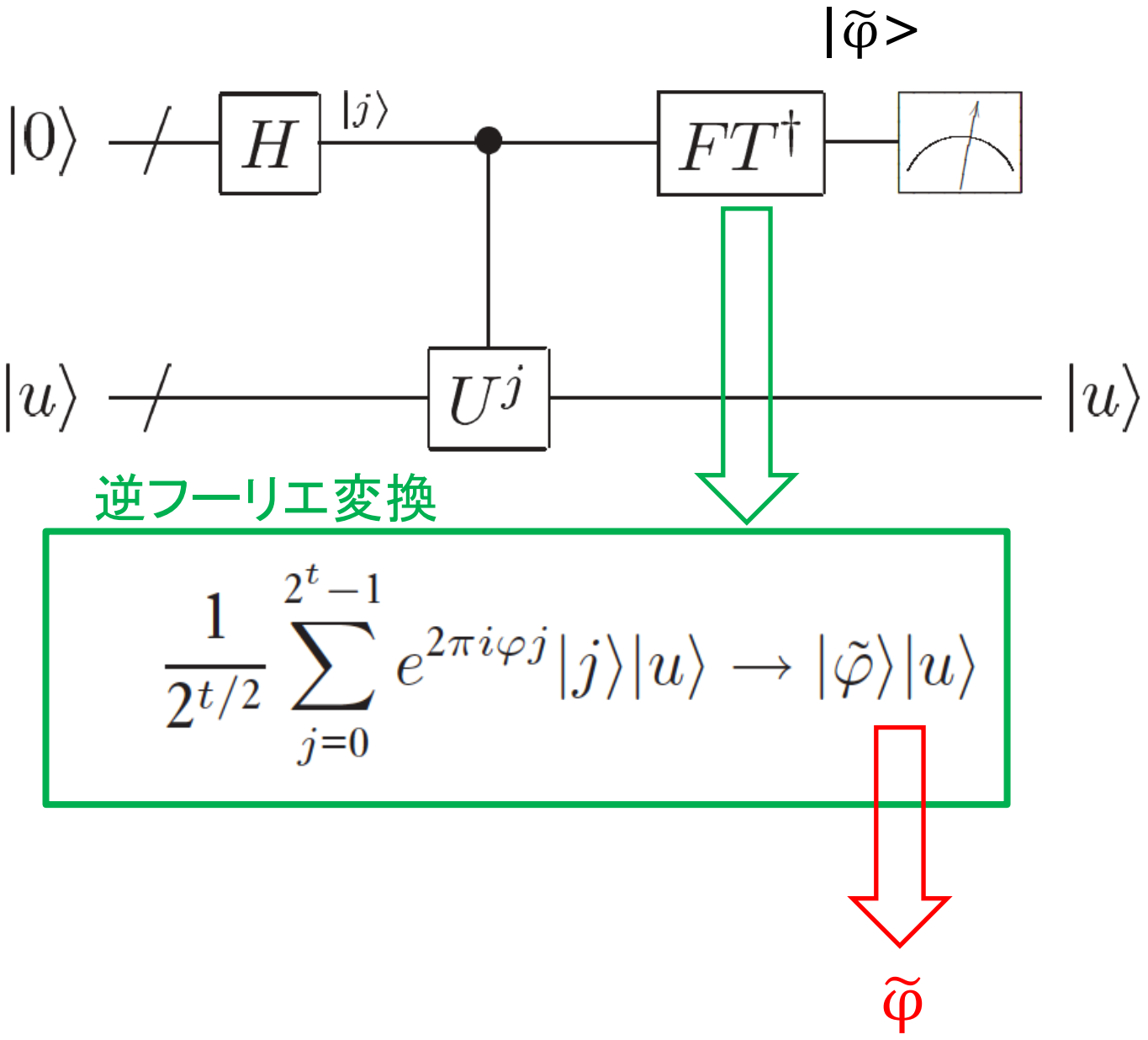
Phase estimator 回路



$U|u\rangle = e^{2\pi i\phi}|u\rangle$

|u>がUの固有ベクトルで、$e^{2\pi i\phi}$ がUの固有値に
なるようなφを求める回路をPhase estimatorという。

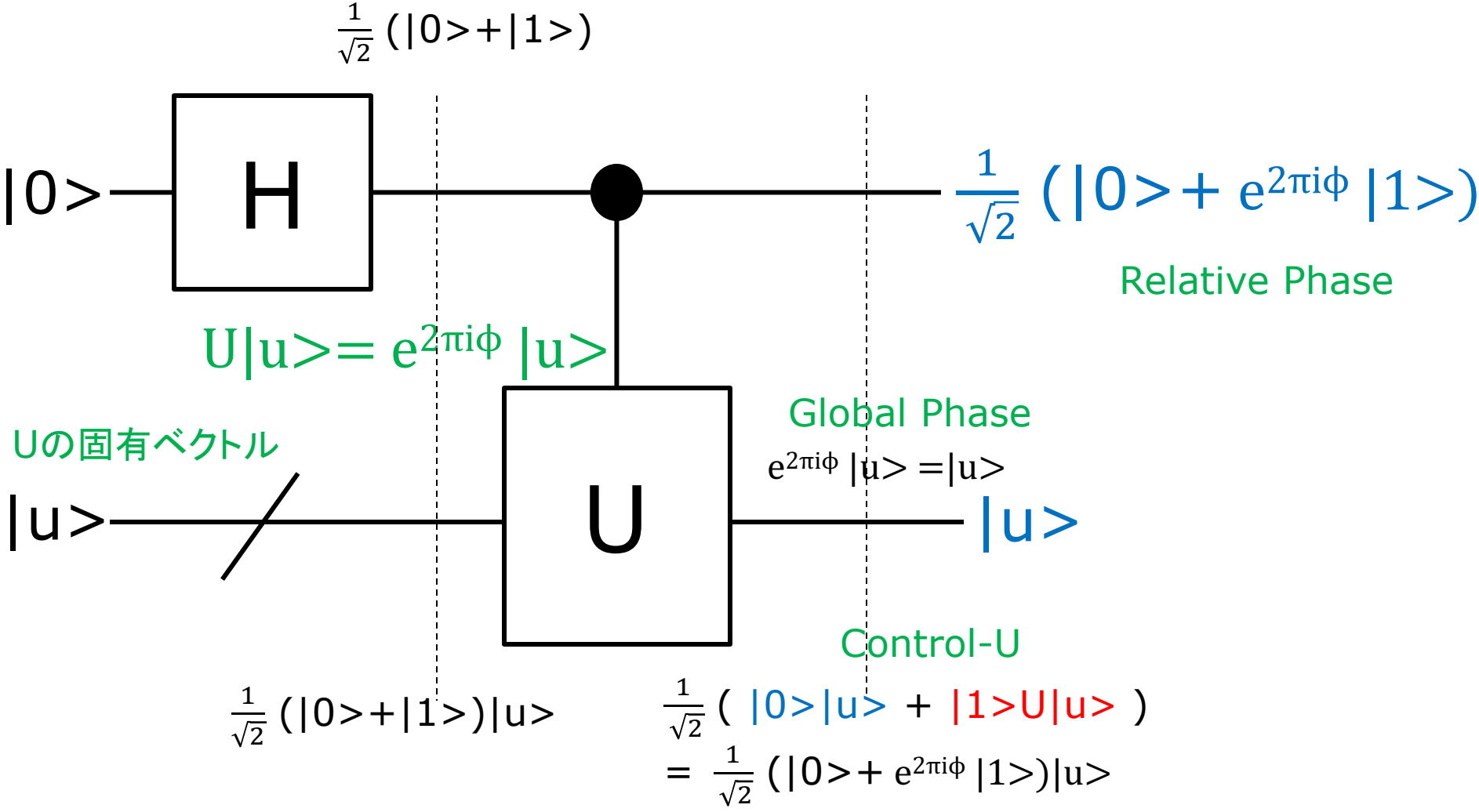
- この回路は、まず、第一レジスターに、アダマール変換を適用する。続いて、第二レジスターに、コントロールUを実行する。このコントロールUは、2のべき乗で指数が増えていく。
- Phase Estimationの第二段階では、第一レジスターに、逆フーリエ変換を適用する。
- Phase Estimationの第三段階では、第一レジスターを、計算基底で測定する。


|φ̃>
|0⟩
H
|j⟩
FT†
|u⟩
U^j
|u⟩
逆フーリエ変換
$\frac{1}{2^{t/2}} \sum_{j=0}^{2^t-1} e^{2\pi i \varphi j} |j\rangle |u\rangle \rightarrow |\tilde{\varphi}\rangle |u\rangle$
φ̃

Phase Estimator

Control-U^j 回路とフーリエ変換

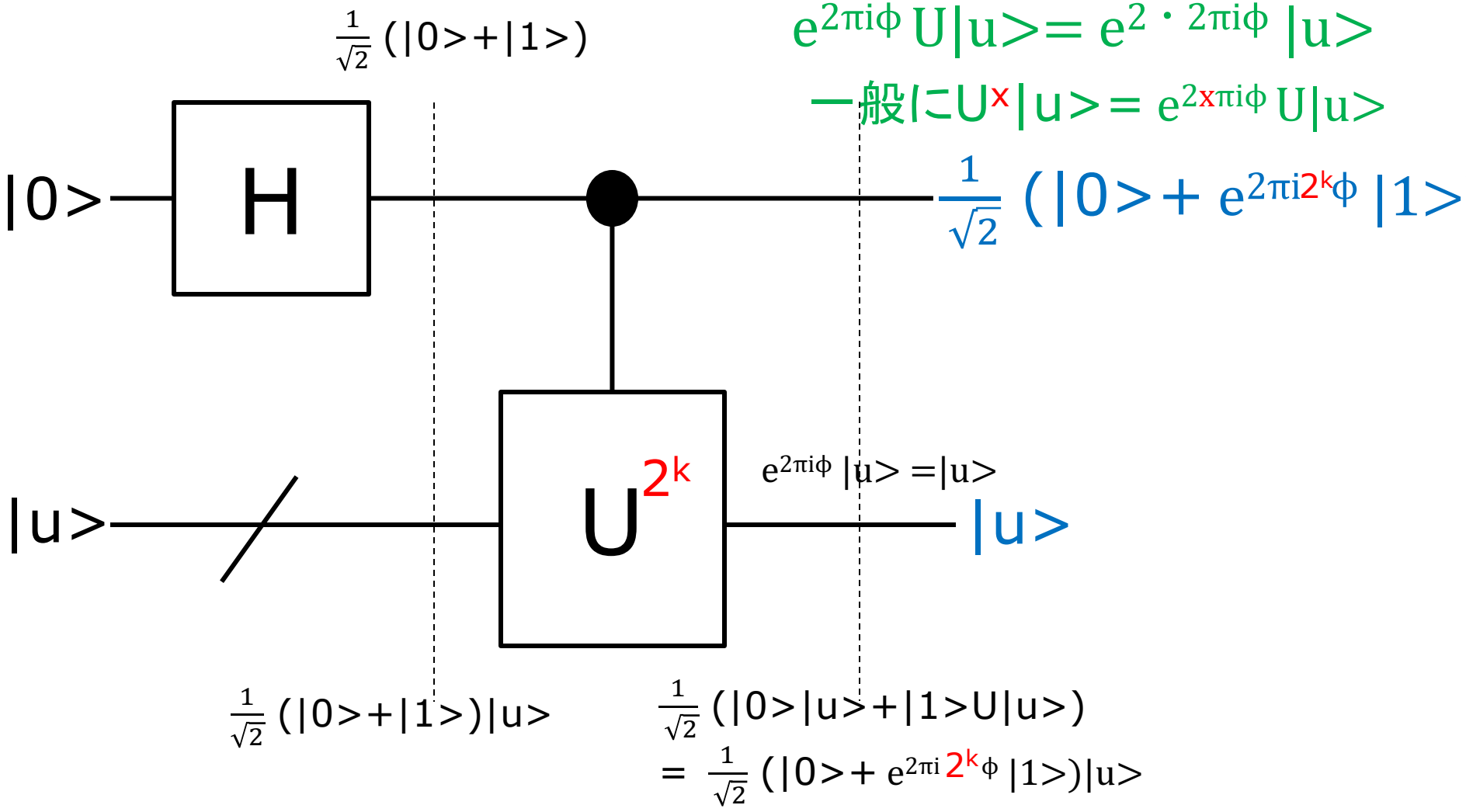
Phase kick-back



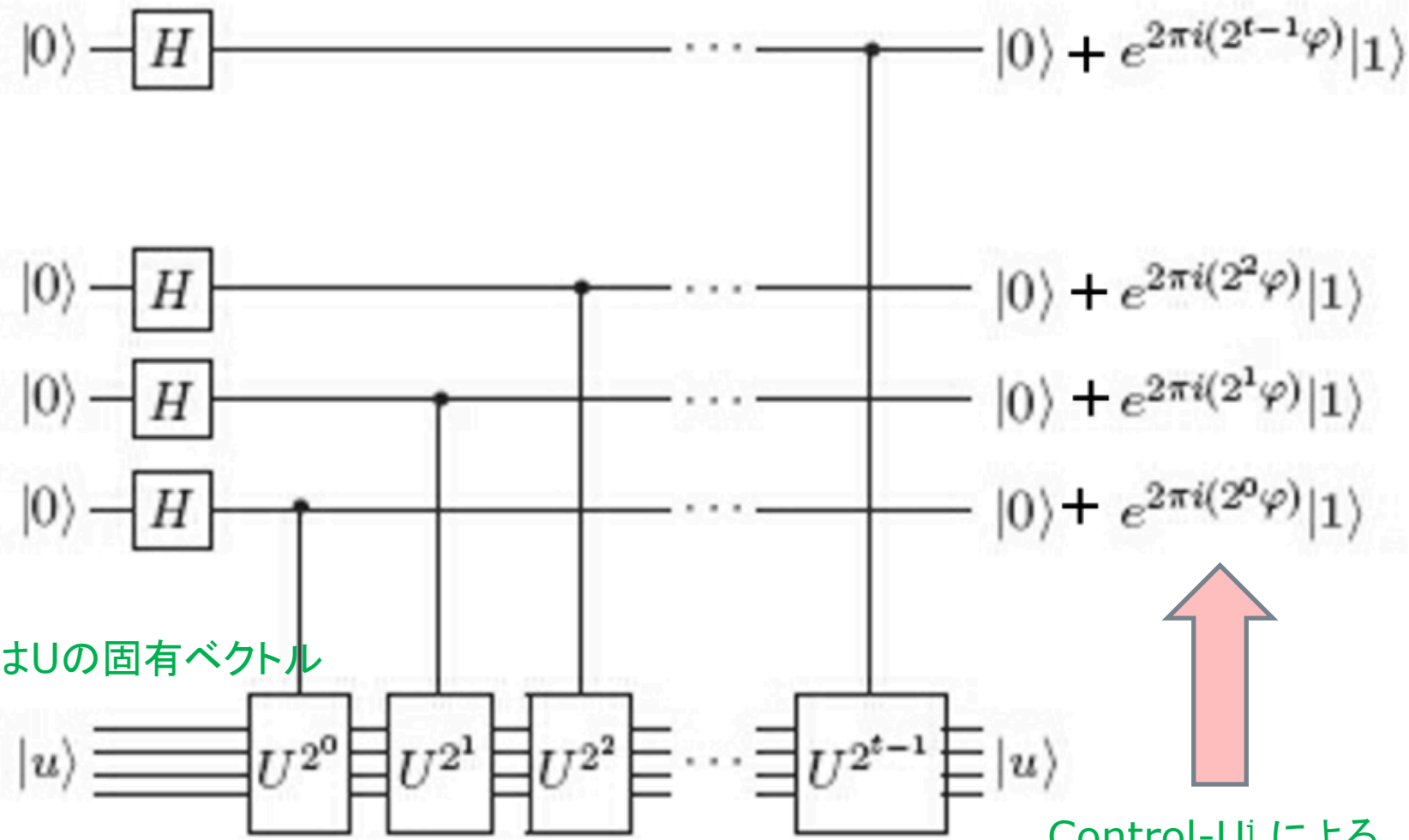
Phase kick-back

$U^2|u\rangle = U(U|u\rangle) = e^{2\pi i\phi} U|u\rangle = e^{2 \cdot 2\pi i\phi} |u\rangle$

一般に $U^x|u\rangle = e^{2x\pi i\phi} U|u\rangle$



Control-U^j 回路



|u>はUの固有ベクトル

$U|u\rangle = e^{2\pi i\phi}|u\rangle$

Control-U^j による
Phase kick-back

Control-Uj 回路の出力
なぜ、フーリエ変換が出てくるのか？

このControl-U^j 回路の出力は、次のようになる。

$$\frac{1}{2^{t/2}}\left(|0\rangle + e^{2\pi i 2^{t-1}\varphi}|1\rangle\right)\left(|0\rangle + e^{2\pi i 2^{t-2}\varphi}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i 2^{0}\varphi}|1\rangle\right)$$

もし、この式のφが、次のようなt-bitで、$\varphi = 0.\varphi_1 \ldots \varphi_t$ と表されているとすると、この出力は次のように変形される。

$$\frac{1}{2^{t/2}}\left(|0\rangle + e^{2\pi i 0.\varphi_t}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.\varphi_{t-1}\varphi_t}|1\rangle\right)\ldots\left(|0\rangle + e^{2\pi i 0.\varphi_1\varphi_2\cdots\varphi_t}|1\rangle\right)$$

これは、量子フーリエ変換の定義に他ならない。

フーリエ変換の和公式と積公式

□ フーリエ変換の和公式

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle$$

□ フーリエ変換の積公式

$$|j_1, \ldots, j_n\rangle \rightarrow \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right)}{2^{n/2}}$$

この証明は省略する

FT: Fourier 变换

$$|j_1,\ldots,j_n\rangle \rightarrow \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2\cdots j_n}|1\rangle\right)}{2^{n/2}}$$

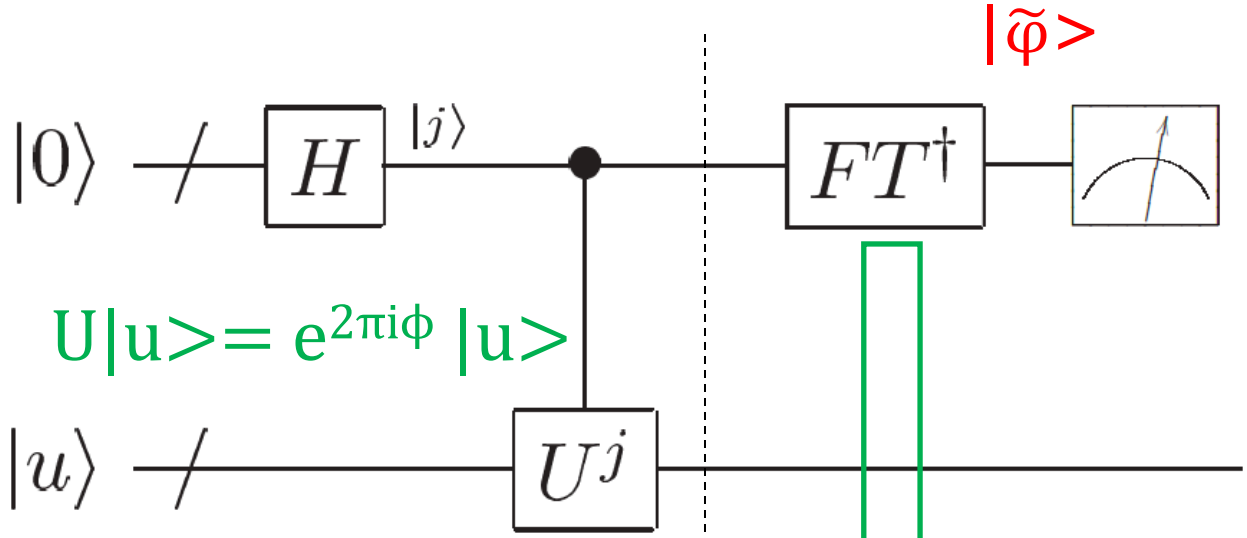
$$|j\rangle \longrightarrow \frac{1}{2^{t/2}}\sum_{j=0}^{2^t-1} e^{2\pi i\varphi j}|j\rangle$$

$$\frac{1}{2^{t/2}}\sum_{j=0}^{2^t-1} e^{2\pi i\varphi j}|j\rangle|u\rangle \rightarrow |\tilde{\varphi}\rangle|u\rangle$$

FT† : 逆Fourier 变换

$$\frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2\cdots j_n}|1\rangle\right)}{2^{n/2}} \rightarrow |j_1,\ldots,j_n\rangle$$

$|\tilde{\varphi}\rangle$



フーリエ変換

$|j\rangle|u\rangle \longrightarrow \frac{1}{2^{t/2}} \sum_{j=0}^{2^t-1} e^{2\pi i\varphi j}|j\rangle|u\rangle$

逆フーリエ変換

$$\frac{1}{2^{t/2}} \sum_{j=0}^{2^t-1} e^{2\pi i\varphi j}|j\rangle|u\rangle \rightarrow |\tilde{\varphi}\rangle|u\rangle$$

$\tilde{\varphi}$

Phase Estimationの働き

1. $|0\rangle|u\rangle$ 初期状態

2. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|u\rangle$ 重ね合わせを作る

3. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle U^j|u\rangle$ ブラックボックスU_jを作用させる

$= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i j \varphi_u} |j\rangle|u\rangle$ ブラックボックスU_jの作用の結果

4. $\rightarrow |\varphi_u\rangle|u\rangle$ 逆フーリエ変換を作用させる

5. $\rightarrow \varphi_u$ 第一レジスターを測定する

Phase Estimationを利用して、「周期」を発見する

f(x)＝ a^x mod N なる関数の「周期」

f(x)＝ a^x mod N なる関数の「周期」を発見する量子アルゴリズムは、先のPhase Estimationのアルゴリズムを、次のユニタリ演算子に適用する。

$$U|y\rangle \equiv |xy(\mathrm{mod}\ N)\rangle$$

$U^2|y\rangle=UU|y\rangle=U|xy\rangle=|x^2y\rangle$

$U^3|y\rangle=UU^2|y\rangle=U|x^2y\rangle=|x^3y\rangle$

$U^4|y\rangle=UU^3|y\rangle=U|x^3y\rangle=|x^4y\rangle$

$U^n|y\rangle=|x^ny\rangle$

Phase Estimator
回路のU^jの計算

$$\begin{aligned}|z\rangle|y\rangle \rightarrow |z\rangle U^{z_t 2^{t-1}} \ldots U^{z_1 2^0}|y\rangle & \\ = |z\rangle|x^{z_t 2^{t-1}} \times \cdots \times x^{z_1 2^0} y(\mathrm{mod}\ N)\rangle & \\ = |z\rangle|x^z y(\mathrm{mod}\ N)\rangle. &\end{aligned}$$

Uの固有ベクトル

次の状態は、Uの固有ベクトルだということがわかる。

$$|u_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] |x^k \bmod N\rangle$$

なぜなら、

$$\begin{aligned} U|u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] |x^{k+1} \bmod N\rangle \\ &= \exp\left[\frac{2\pi i s}{r}\right] |u_s\rangle \,. \end{aligned}$$

線形シフト

$$\mathcal{QFT}^{(N)} \left|x - x_0\right\rangle^n \quad = \quad \omega^{x_0 x} \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{yx} \left|y\right\rangle^n .$$

$$\mathcal{QFT}^{(N)} \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \left|x_0 + ja\right\rangle^n$$

$$= \quad \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x_0\, y}\ \omega^{jay}\ \left|y\right\rangle^n$$

$$= \quad \frac{1}{\sqrt{mN}} \sum_{y=0}^{N-1} \sum_{j=0}^{m-1} \omega^{x_0\, y}\ \omega^{jay}\ \left|y\right\rangle^n$$

$$= \quad \frac{1}{\sqrt{mN}} \sum_{y=0}^{N-1} \omega^{x_0\, y} \sum_{j=0}^{m-1} \omega^{jay}\ \left|y\right\rangle^n$$

$$\sum_{j=0}^{m-1} \omega^{jay} = \sum_{j=0}^{m-1} {\omega_m}^{jy} = \begin{cases} m, & \text{if } y \equiv 0 \pmod{m} \\ 0, & \text{if } y \not\equiv 0 \pmod{m} \end{cases}$$

- もし、f が N/r=k なる周期r を持つなら、フーリエ変換された$\hat{f}$は、周期 kを持つ。さらに、線形シフトの影響は、$\hat{f}$のフェーズ部分に現れるだけである。
- だから、第一レジスターの逆フーリエ変換を取ると、N/r の整数倍の状態が残るだけである。

$$\sqrt{\frac{r}{N}} \sum_{i=0}^{N/r-1} |ir + x_0\rangle \xrightarrow{QFT_N} \frac{1}{\sqrt{r}} \sum_{i=0}^{r-1} \left| i\frac{N}{r} \right\rangle \phi_i$$

ここに、ϕ_i
は、$x0$の線形シフトの影響で現れたフェーズの部分である。

周期を求めるアルゴリズム

1. $|0\rangle|0\rangle$ 初期状態

2. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|0\rangle$ 重ね合わせを作る

3. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|f(x)\rangle$ Uを作用させる

   $\approx \frac{1}{\sqrt{r2^t}} \sum_{\ell=0}^{r-1} \sum_{x=0}^{2^t-1} e^{2\pi i \ell x/r} |x\rangle|\hat{f}(\ell)\rangle$ Uの作用の結果

4. $\rightarrow \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} |\ell/r\rangle|\hat{f}(\ell)\rangle$ 逆フーリエ変換を作用させる

5. $\rightarrow \ell/r$ 第一レジスターを測定する

6. $\rightarrow r$ 周期 r を求める

実際の計算例

15をShorのアルゴリズムで素因数分解する

|x>|f(x)>を計算

- N=15とする。N以下の数字を一つ選ぶ。
  x=7を選んだとしよう。
- 量子回路では、まずアダマールで可能な全ての基底の重ね合わせを生成する。
- 次に、その出力を $|x^k \bmod 15\rangle$ を計算するブラックボックスに渡す。
- この時、回路の状態は次のようになる。(正規化因子等省略)

$$\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle |x^k \bmod N\rangle$$

=|0>|7^0>+|1>|7^1>+|2>|7^2>+|3>|7^3>+|4>|7^4> …

=|0>|1>+|1>|7>+|2>|49>+|3>|343>+|4>|2401> …

=|0>|1>+|1>|7>+|2>|4>+|3>|13>+|4>|1> …

mod 15で計算

第二レジスターを観測して
第一レジスター原像の重ね合わせを得る

- この結果の第二レジスターを観測する。

$$\frac{1}{\sqrt{2^t}}\left[|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + |6\rangle|4\rangle + \cdots\right]$$

|1>,|7>,|4>,|13>のどれかを観測することになる。
ここで、4を観測したとしよう。

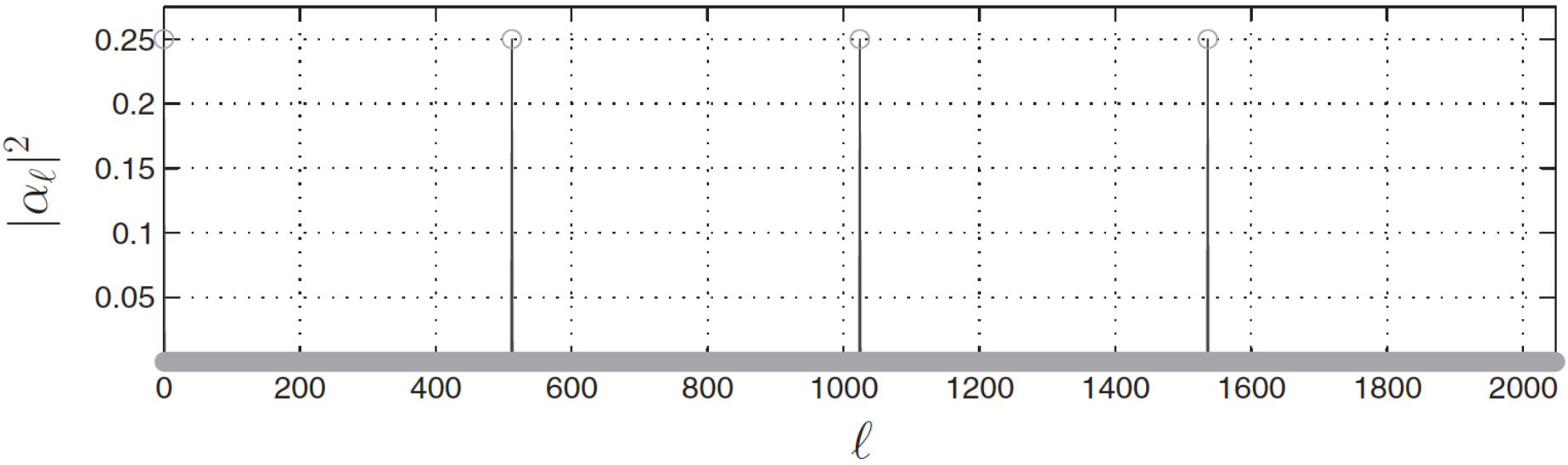
$$\frac{1}{\sqrt{2^t}}\left[|0\rangle|1\rangle + |1\rangle|7\rangle + \boxed{|2\rangle|4\rangle} + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + \boxed{|6\rangle|4\rangle} + \cdots\right]$$

この時、逆フーリエ変換への入力は、次のようになる。

$$\sqrt{\frac{4}{2^t}}\left[|2\rangle + |6\rangle + |10\rangle + |14\rangle + \cdots\right].$$

逆フーリエ変換を実行する

- その結果は、次のようになる。



ここで $FT^\dagger\left(\sqrt{\frac{4}{2^t}}\left[|2\rangle + |6\rangle + |10\rangle + |14\rangle + \cdots\right].\right)$

$= \sum_\ell \alpha_\ell |\ell\rangle$ としている。

逆フーリエ変換の結果を観測する

- lとして観測されるのは、有効ビットのN=2^{11}=2048以下では、0, 512, 1024, 1536 である。
  ここでは、1536 を観測したとしよう。

- lは、N/rの整数倍だから、1536/2048=1/(1+(1/3)) で 3/4。ここから x=7の場合の周期 r=4がわかる。

x=7の周期r=4から、15を素因数分解する

- rは偶数だったので、次のような変形ができる。
  $x^{r/2} \bmod 15 = 7^2 \bmod 15 = 4 \neq -1$
  これで、先に見たアルゴリズムが使えることがわかる。
- $\gcd(x^2 - 1, 15) = \gcd(48, 15) = 3$
  $\gcd(x^2 + 1, 15) = \gcd(50, 15) = 5$
- ここから、15 = 3 x 5 がわかる！

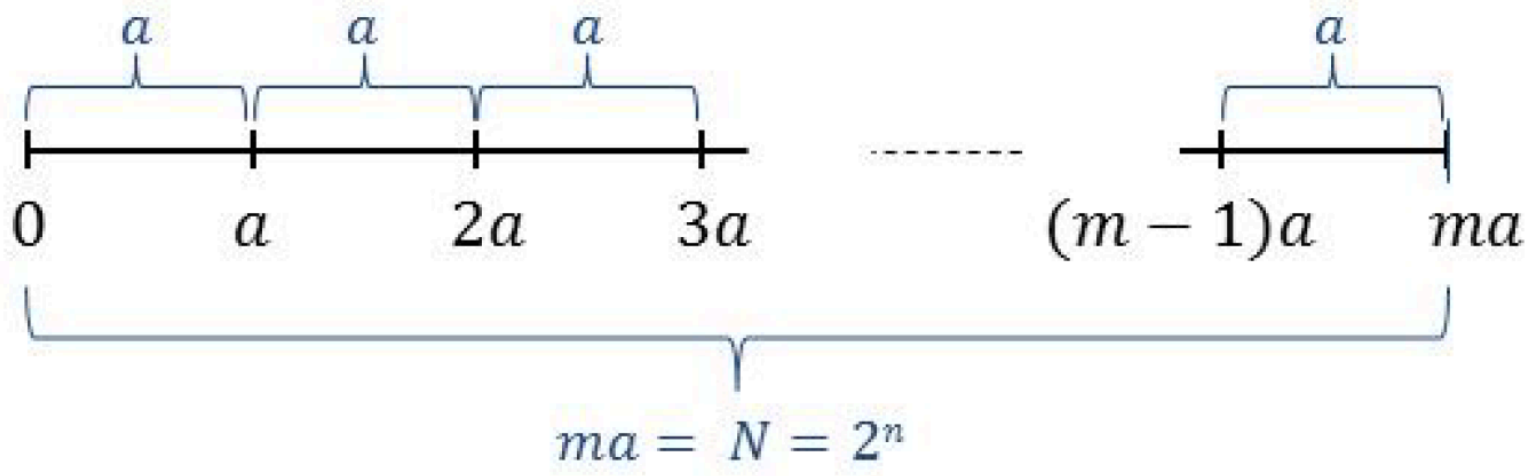
素晴らしい！？

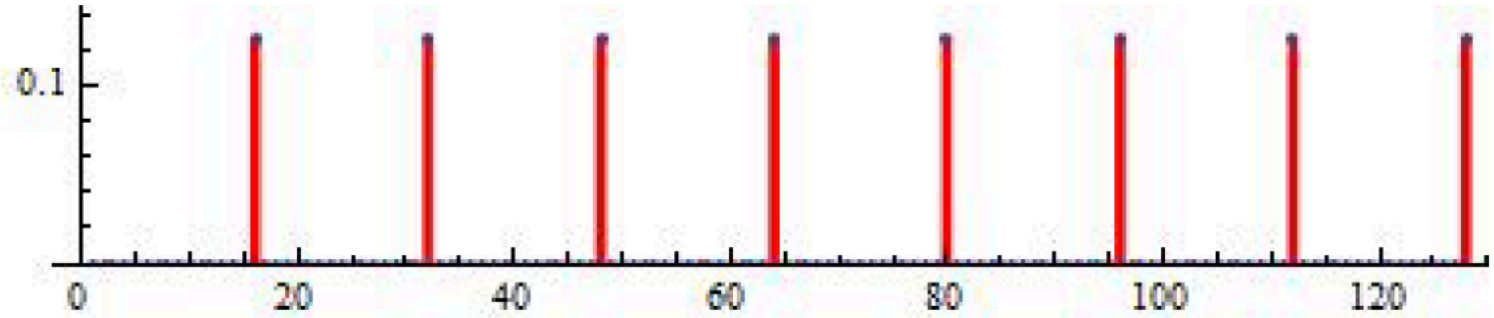
観測と周期の確定

観測値から周期を求める

- 観測されるのは l/r の整数倍の値で rそのものではない。何度か試行を行って、rを求めていく。それはSimonのアルゴリズムで y･a=0 を満たすいくつかのyを観測して、その複数の条件からaを求めたのと同様である。
- Shorのアルゴリズムで観測値からどのようにrを求めるのかについては、今回のセミナーでは省略した。ただ、どのような問題があるのかだけを簡単に記したい。

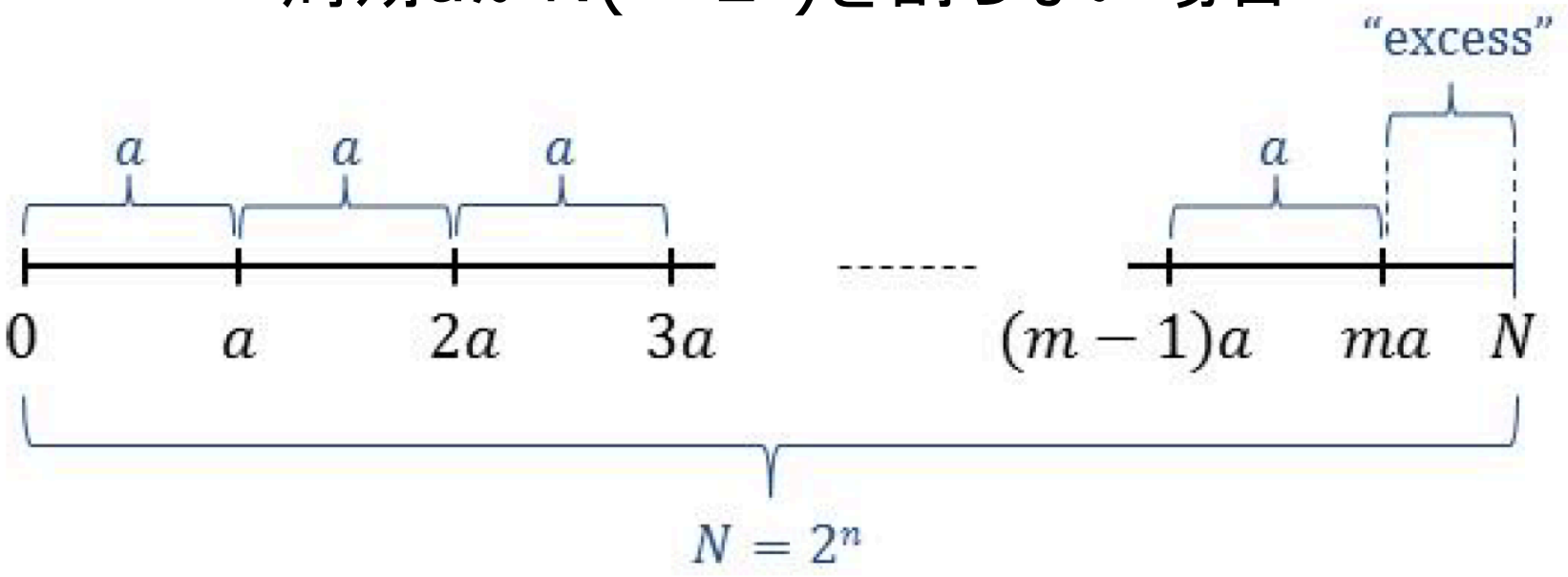
周期aがN(= 2^n)を割る場合

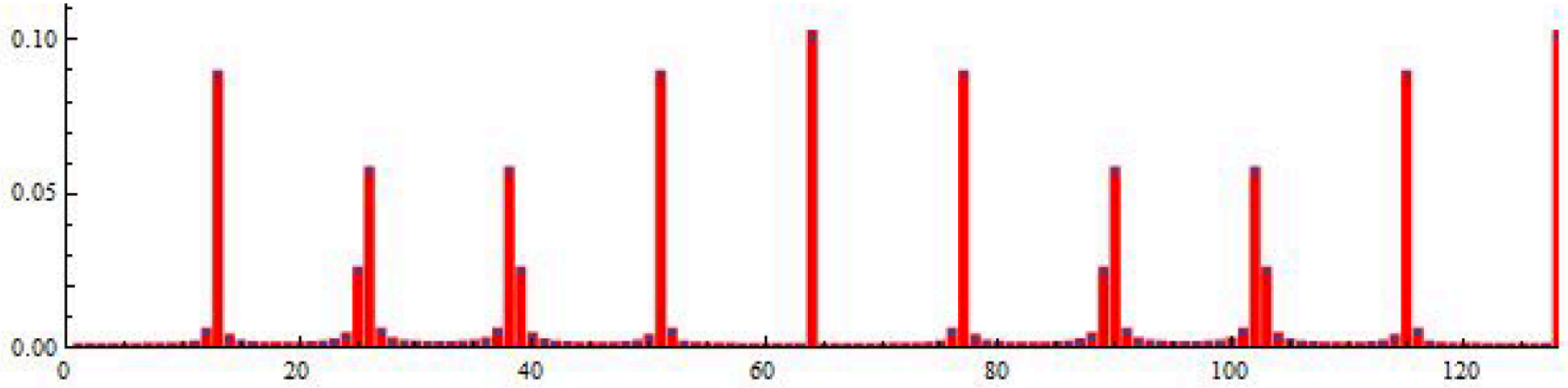




周期a=8, N=128

周期aがN(= 2^n)を割らない場合





周期a=10, N=128

Appendix

Period-finding	$\mathbf{Z}$, +	any finite set	$\{0, r, 2r, \ldots\}$ $r \in G$	$f(x + r) = f(x)$
Order-finding	$\mathbf{Z}$, +	$\{a^j\}$ $j \in Z_r$ $a^r = 1$	$\{0, r, 2r, \ldots\}$ $r \in G$	$f(x) = a^x$ $f(x + r) = f(x)$
Discrete logarithm	$\mathbf{Z}_r \times \mathbf{Z}_r$ + (mod r)	$\{a^j\}$ $j \in Z_r$ $a^r = 1$	$(\ell, -\ell s)$ $\ell, s \in \mathbf{Z}_r$	$f(x_1, x_2) = a^{kx_1 + x_2}$ $f(x_1 + \ell, x_2 - \ell s) = f(x_1, x_2)$
Order of a permutation	$\mathbf{Z}_{2^m} \times \mathbf{Z}_{2^n}$ + (mod 2^m)	$\mathbf{Z}_{2^n}$	$\{0, r, 2r, \ldots\}$ $r \in X$	$f(x, y) = \pi^x(y)$ $f(x + r, y) = f(x, y)$ π = permutation on X
Hidden linear function	$\mathbf{Z} \times \mathbf{Z}$, +	$\mathbf{Z}_N$	$(\ell, -\ell s)$ $\ell, s \in X$	$f(x_1, x_2) =$ $\pi(sx_1 + x_2 \bmod N)$ π = permutation on X
Abelian stabilizer	(H, X) H = any Abelian group	any finite set	$\{s \in H \mid$ $f(s, x) = x,$ $\forall x \in X\}$	$f(gh, x) = f(g, f(h, x))$ $f(gs, x) = f(g, x)$

Shorのアルゴリズムで解ける問題

http://bit.ly/2Ww2Kua

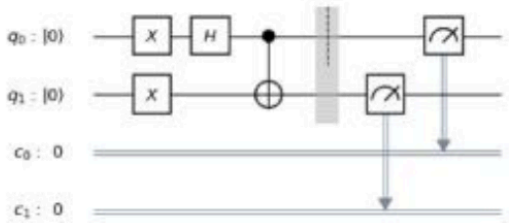
お知らせ

IBM Qiskit、python

量子コンピュータで学ぶ
量子プログラミング入門

好評につき
リピート開催！

講師
丸山不二夫氏
連続講演会マルレク主宰



HOSTED BY
角川アスキー総合研究所
KADOKAWA ASCII Research Laboratories, Inc.

2019
7.6
（土曜）
東京・市ヶ谷
13:00
-
19:00
プログラミング+
http://ascii.jp/programming/

https://lab-kadokawa82.peatix.com/