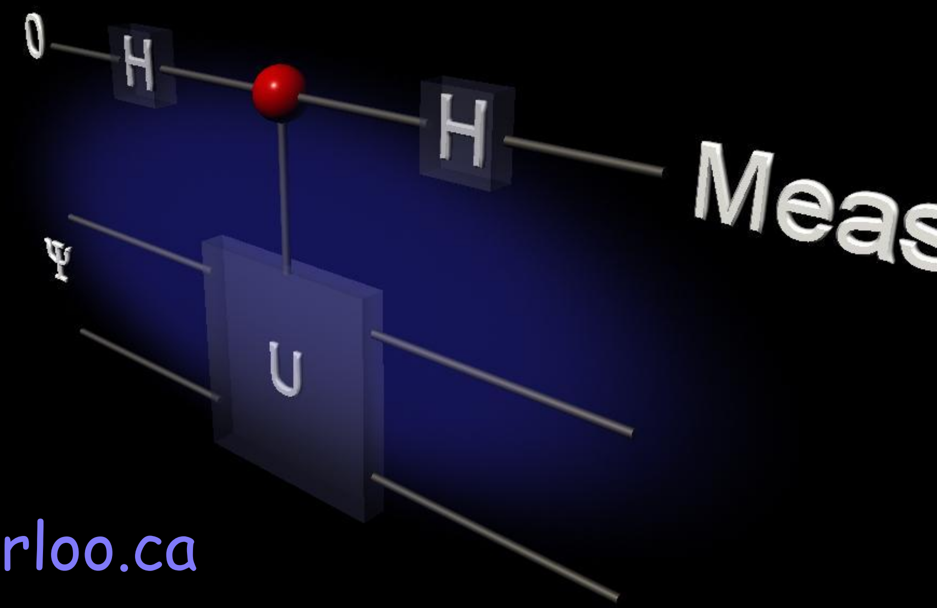
Introduction to Quantum Information Processing



CO481 CS467 PHYS467

Michele Mosca mmosca@iqc.uwaterloo.ca

Tuesdays and Thursdays 10am-11:15am

IQC Institute for Quantum Computing

UNIVERSITY OF WATERLOO

PERIMETER INSTITUTE FOR THEORETICAL PHYSICS

Reading

Chapter 6, sections 7.1.1, 7.1.3, 7.2, 7.3.1, 7.3.2, 7.3.3

Recall: Multi-qubit Hadamard

$$|x\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_y (-1)^{x \cdot y} |y\rangle$$

$$\frac{1}{\sqrt{2^n}} \sum_y (-1)^{x \cdot y} |y\rangle \xrightarrow{H^{\otimes n}} |x\rangle$$

Quantum algorithms

- The algorithms we have seen have been computing “classical” functions on quantum superpositions

- This encoded information in the phases of the basis states: measuring basis states would provide little useful information

- But a simple quantum transformation translated the phase information into information that was measurable in the computational basis

Quantum factoring

- The security of many public key cryptosystems used in industry today relies on the difficulty of factoring large numbers into smaller factors.

- Factoring the integer N into smaller factors can be reduced to the following task:

Given integer a, find the smallest positive integer r so that $a^r \equiv 1 \bmod N$

Complexity

- The best known <u>rigorous classical</u> algorithms use

$$\boxed{e^{O(\sqrt{\log(N)\log\log(N)})}}$$

operations

- The best known <u>heuristic classical</u> algorithms use

$$\boxed{e^{O((\log(N)^{\frac{1}{3}}\log\log(N)^{\frac{2}{3}})}}$$

operations

(Aside: how does factoring reduce to order-finding ?)

- The most common approach for factoring integers is the difference of squares technique:
  - “Randomly” find two integers *x* and *y* satisfying

$$x^2 = y^2 \bmod N$$

  - So *N* divides $x^2 - y^2 = (x - y)(x + y)$
  - Hope that $\gcd(N, x - y)$ is non-trivial

- If r is even, then let

$$x = a^{r/2} \bmod N$$

so that

$$x^2 = 1^2 \bmod N$$

Quantum factoring

Since we know how to efficiently multiply by a mod *N*, we can efficiently implement

$$U_a|x\rangle = |ax\rangle$$

Note that

$$U_{a^r}|x\rangle = |a^r x\rangle = |x\rangle$$

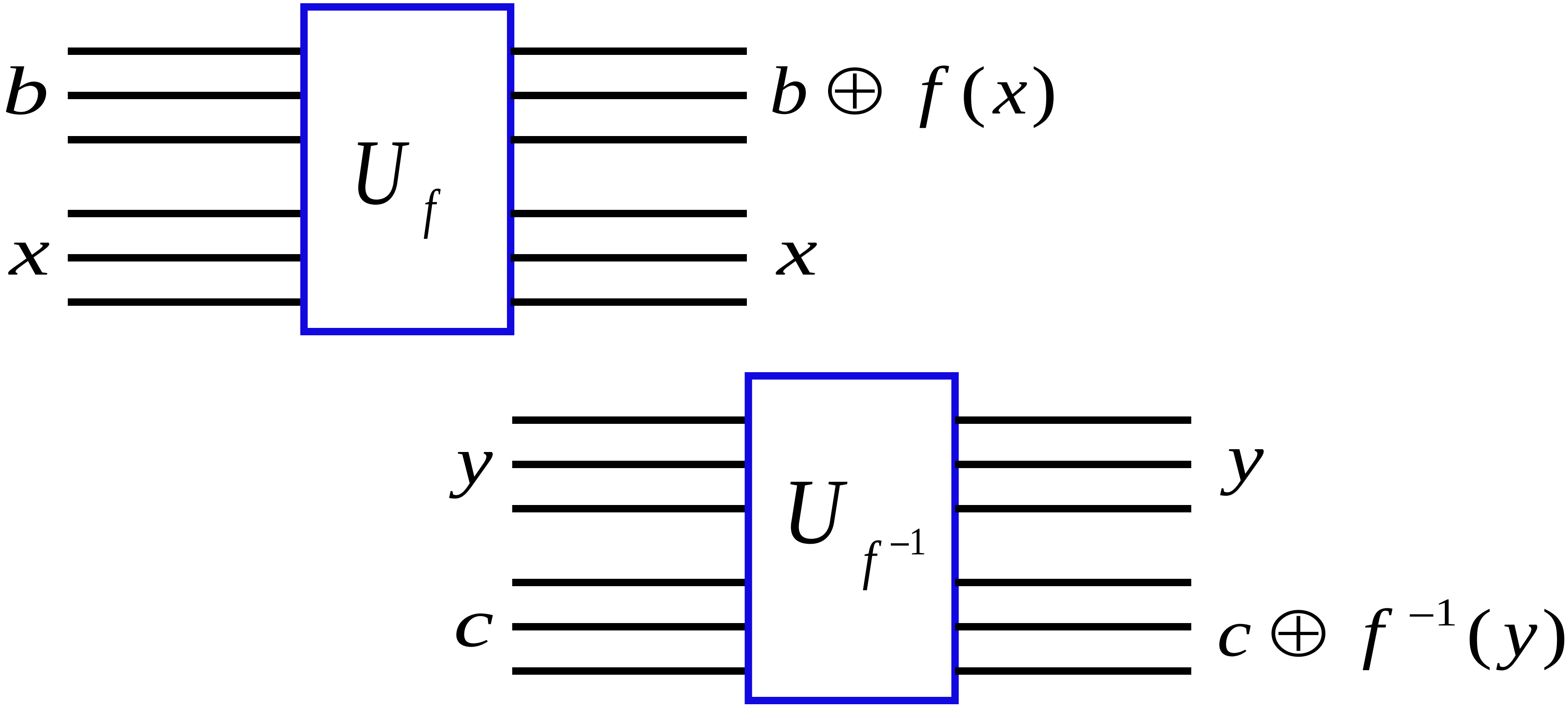
i.e.

$$U_a^r = \mathrm{I}$$

Remember that $|x\rangle$ represents the state corresponding to the binary representation of x (e.g. for four qubits, $|2\rangle$ represents $|0010\rangle$)

(Aside : more on reversible computing)

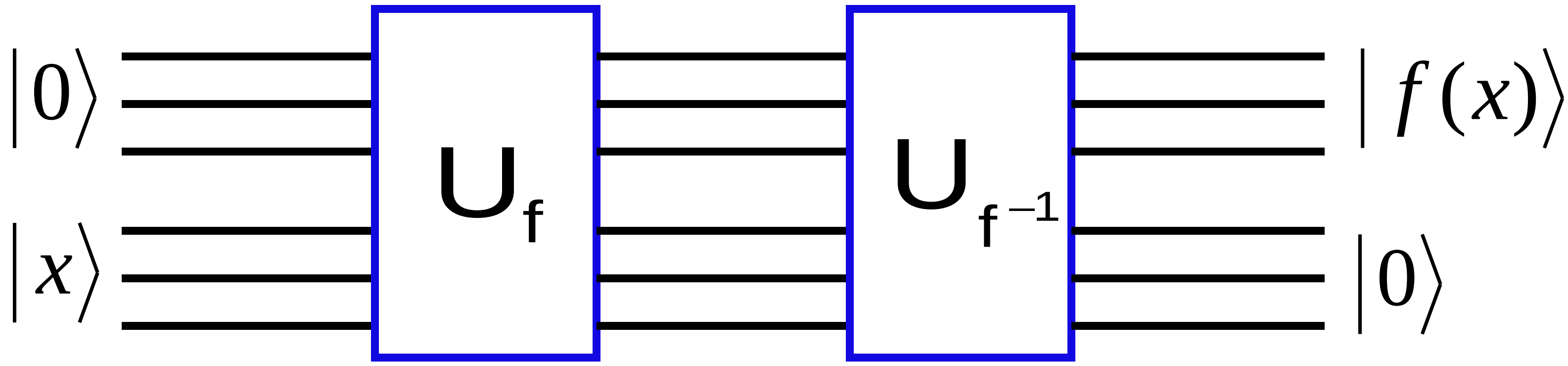
If we know how to efficiently compute ***f*** and ***f*** $^{-1}$ then we can efficiently and reversibly map



(Aside : more on reversible computing)

And therefore, for such invertible f, we can efficiently map

$$|x\rangle \rightarrow |f(x)\rangle$$



Finding r

For most integers k, a good estimate of $\frac{k}{r}$ (with error at most $\frac{1}{2r^2}$) allows us to determine r (even if we don't know k).

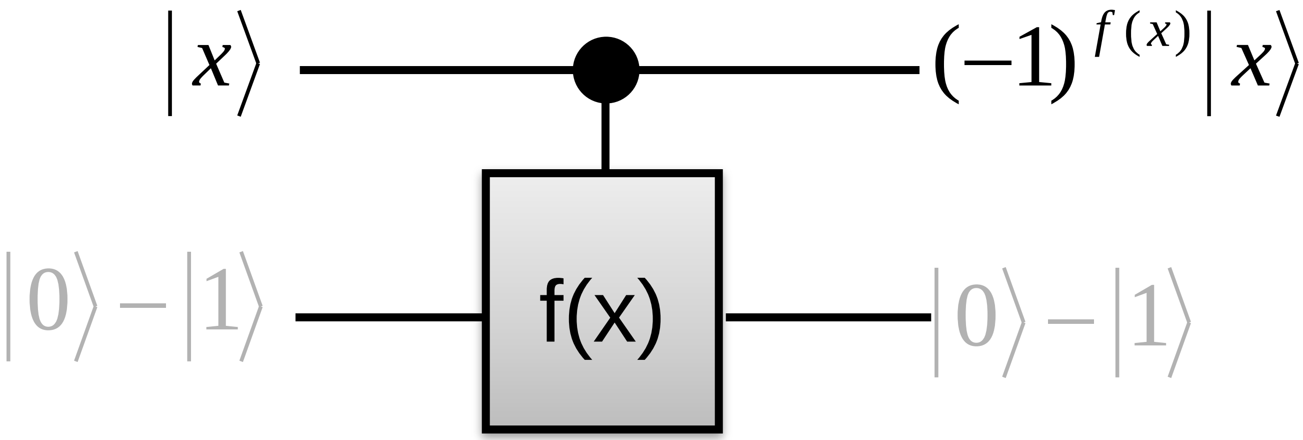
(using continued fractions)

So what?

How would we convert the eigenvalues into something measurable?

Eigenvalue kick-back

- Recall the "trick":



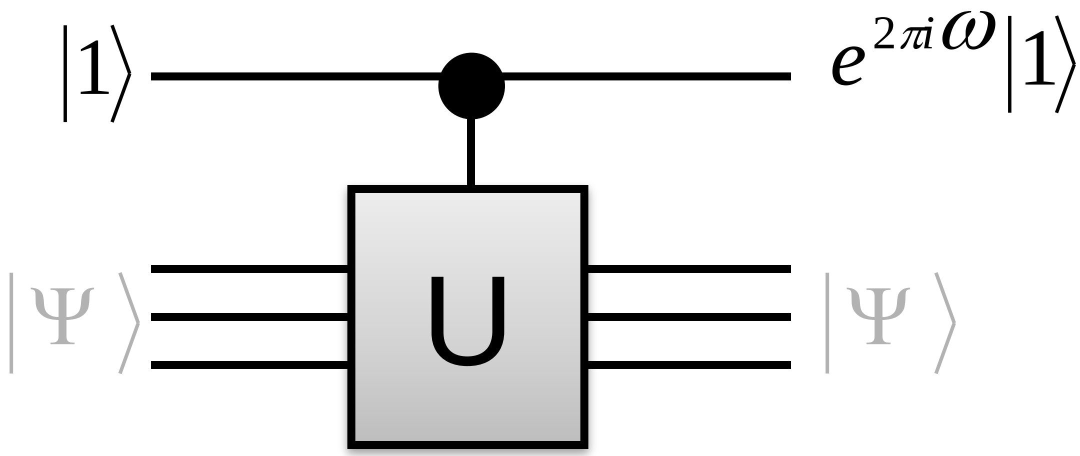
$$|x\rangle(|0\rangle-|1\rangle)\rightarrow|x\rangle(|f(x)\rangle-|f(x)\oplus 1\rangle)$$

$$=|x\rangle(-1)^{f(x)}(|0\rangle-|1\rangle)$$

$$=(-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle)$$

Eigenvalue kick-back (Kitaev)

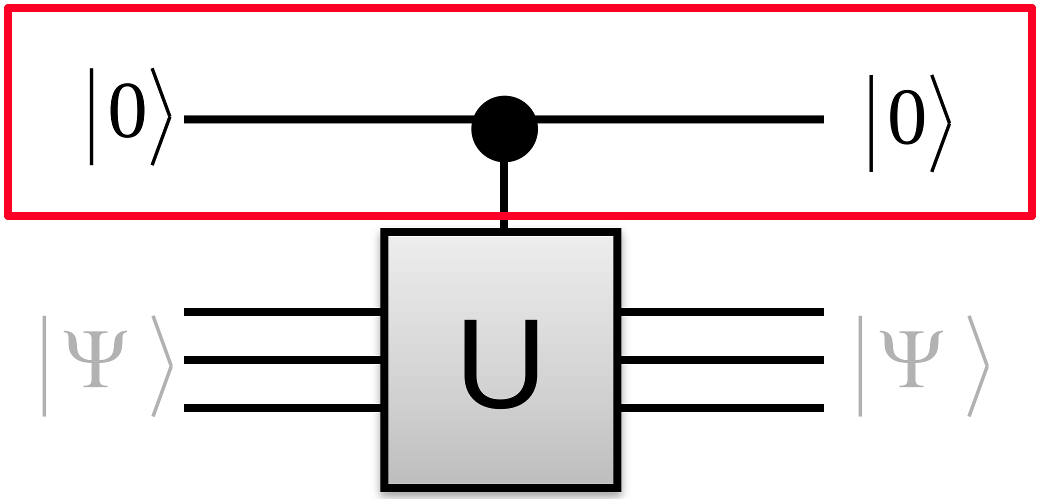
- Consider a unitary operation **U** with eigenvalue $e^{2\pi i\omega}$ and eigenvector $|\Psi\rangle$



$$|1\rangle|\Psi\rangle \to |1\rangle U|\Psi\rangle = |1\rangle e^{2\pi i\omega}|\Psi\rangle$$

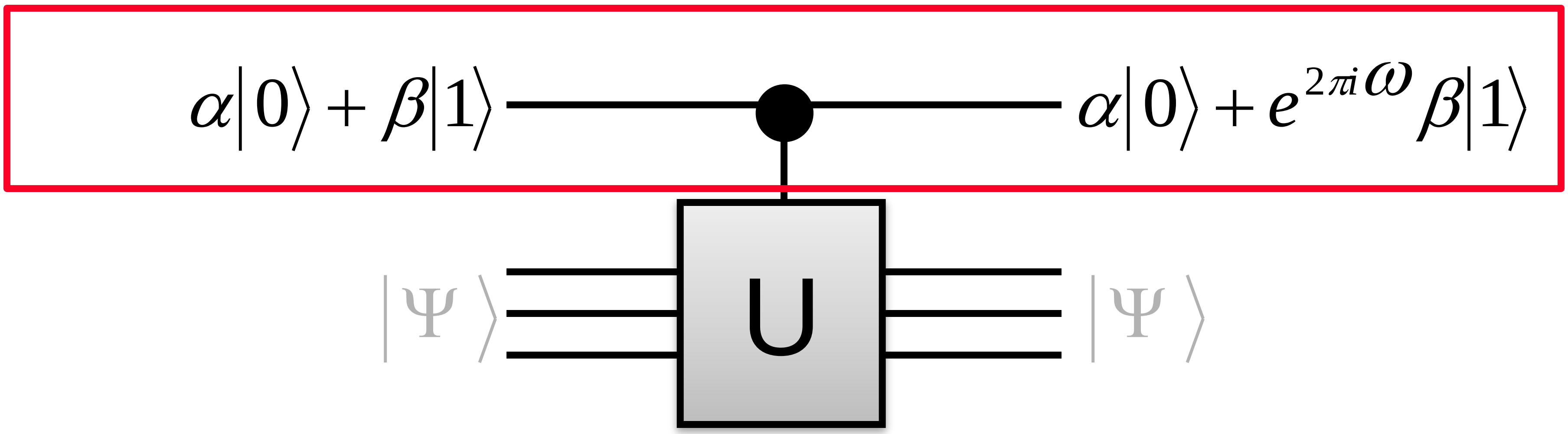
$$= e^{2\pi i\omega}|1\rangle|\Psi\rangle$$

Eigenvalue kick-back



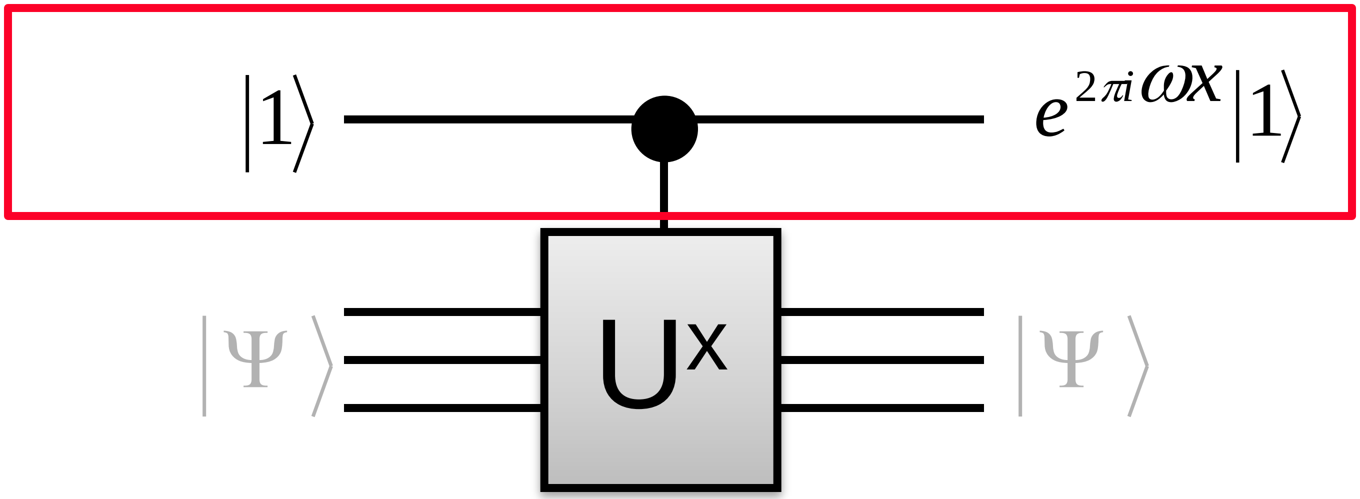
Eigenvalue kick-back

- As a relative phase, $e^{2\pi i\omega}$ becomes measurable

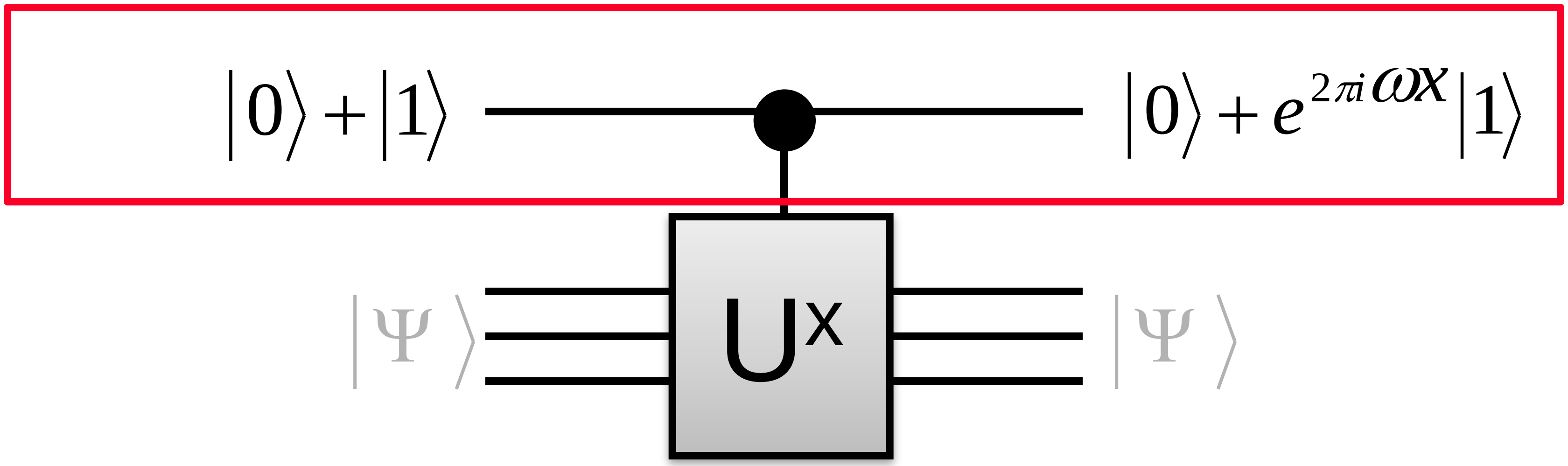


Eigenvalue kick-back

- If we exponentiate **U**, we get multiples of ω

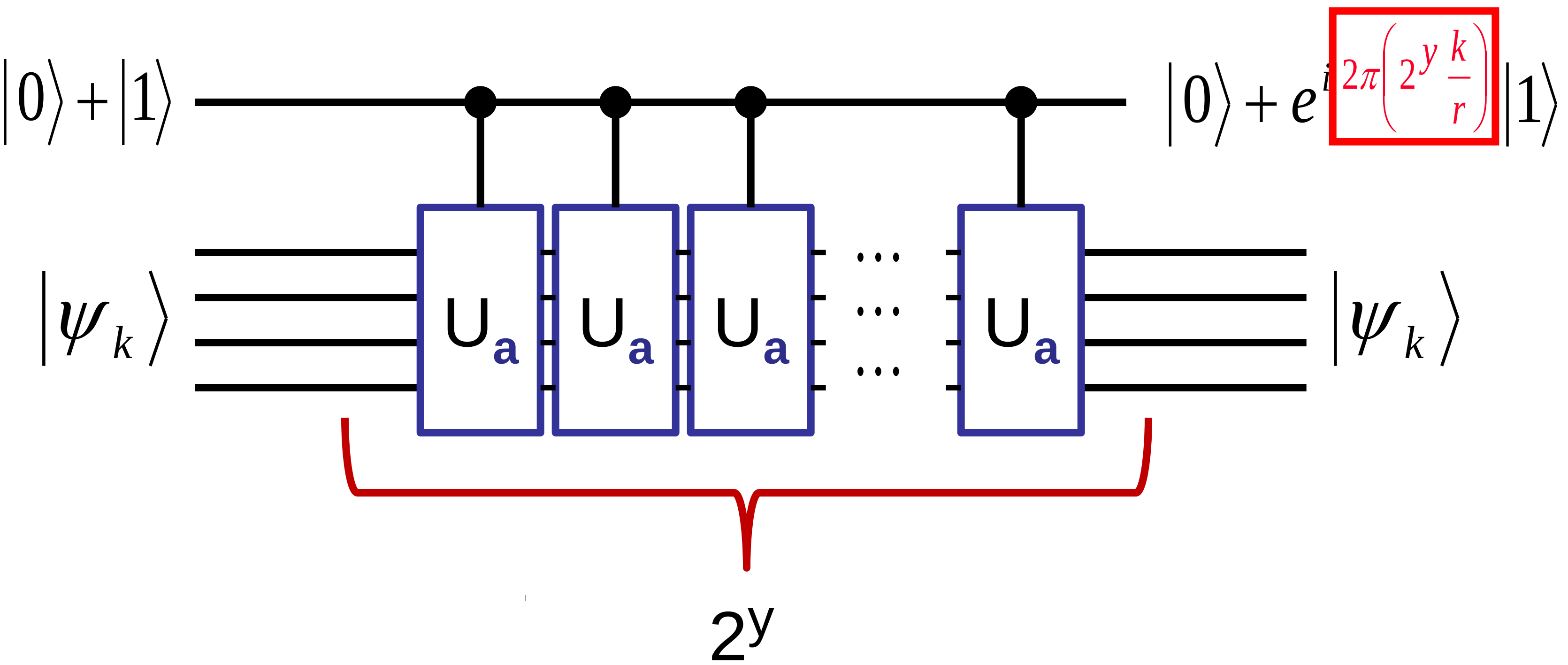


Eigenvalue kick-back



Inefficient exponentiation

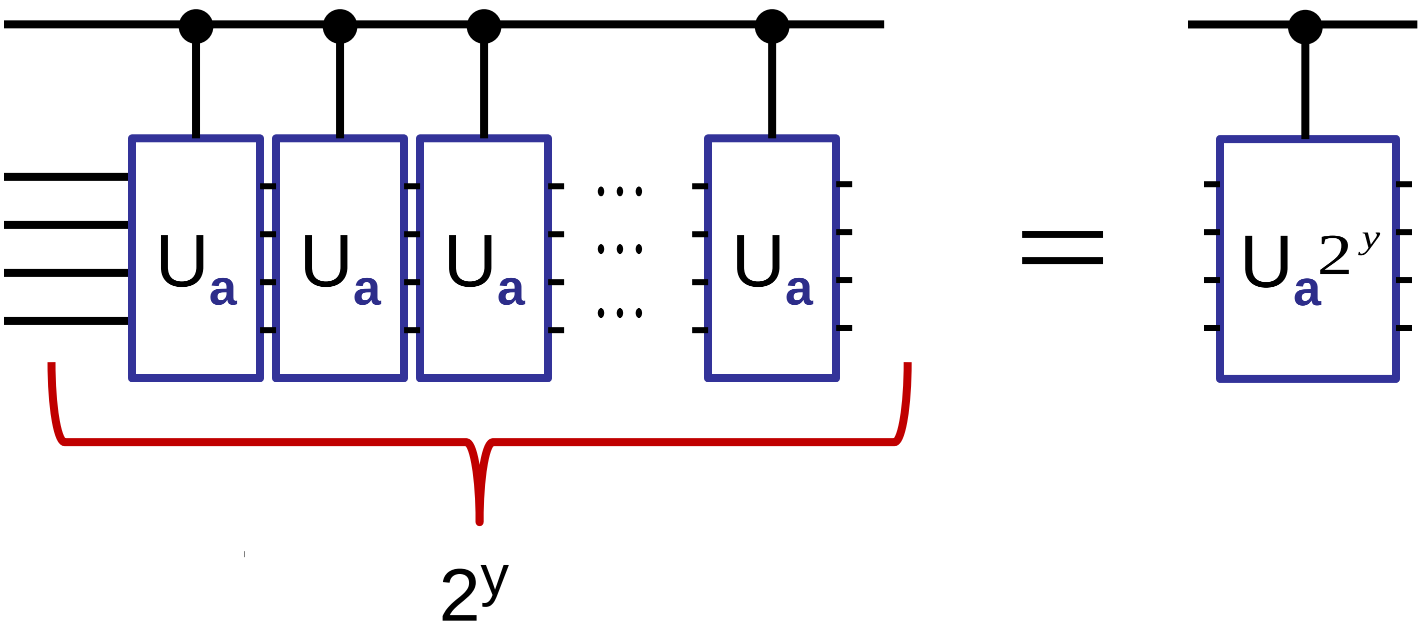
We can effect a relative phase shift of $e^{i\boxed{2\pi\left(2^{y}\frac{k}{r}\right)}}$



Efficient exponentiation

But we can also do it **efficiently** by noticing that

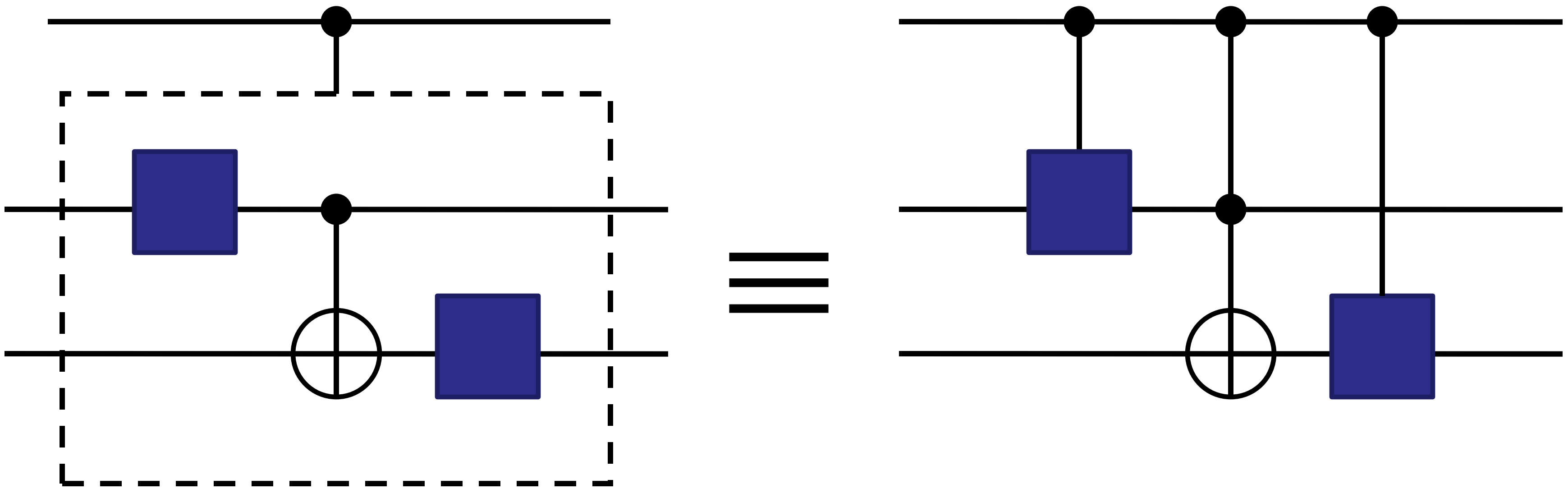
$$U_a^{2^y} = U_{a^{2^y}}$$



How do we implement c-U ?

Replace every gate G in the circuit with a c-G.

For example,



Next step?

- We thus know how, given an eigenvector with eigenvalue $e^{2\pi i\left(2^{y}\frac{k}{r}\right)}$,
  to construct

$$=\left(|0\rangle+e^{2\pi i(2^{n-1}\frac{k}{r})}|1\rangle\right)\otimes\left(|0\rangle+e^{2\pi i(2^{n-2}\frac{k}{r})}|1\rangle\right)\otimes\cdots\otimes\left(|0\rangle+e^{2\pi i(\frac{k}{r})}|1\rangle\right)$$

Useful identity

- We can show that

$$\left(|0\rangle + e^{2\pi i(2^{n-1}\omega)}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i(2^{n-2}\omega)}|1\rangle\right)\otimes\cdots\otimes\left(|0\rangle + e^{2\pi i(\omega)}|1\rangle\right)$$

$$=\sum_{y=0}^{2^n-1} e^{2\pi i\omega y}\,|y\rangle$$

Quantum phase estimation

- Suppose we wish to estimate a number $\omega \in [0,1)$ given the quantum state

$$\sum_{y=0}^{2^n-1} e^{2\pi i \omega y} |y\rangle$$

- Note that in binary we can express

$$\omega = 0.x_1 x_2 x_3 \ldots$$

$$2\omega = x_1.x_2 x_3 \ldots$$

$$2^{n-1}\omega = x_1 x_2 x_3 \ldots x_{n-1}.x_n x_{n+1} \ldots$$

Quantum phase estimation

- Since $e^{2\pi ik} = 1$ for any integer *k*, we have

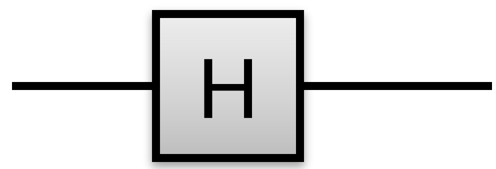
$$e^{2\pi i(2\omega)} = e^{2\pi i(x_1.x_2x_3\ldots)} = e^{2\pi i x_1} e^{2\pi i(0.x_2x_3\ldots)} = e^{2\pi i(0.x_2x_3\ldots)}$$

$$e^{2\pi i(2^k\omega)} = e^{2\pi i(0.x_{k+1}x_{k+2}\ldots)}$$

Quantum phase estimation

- If $\omega = 0.x_1$ then we can do the following

$$\frac{|0\rangle + e^{2\pi i (0.x_1)}|1\rangle}{\sqrt{2}} = \frac{|0\rangle + (-1)^{x_1}|1\rangle}{\sqrt{2}}$$



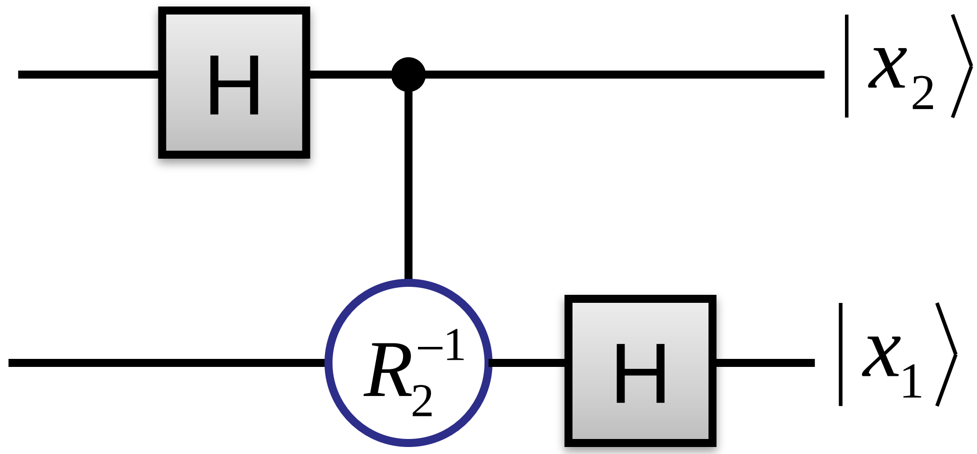
$|x_1\rangle$

Quantum phase estimation

- So if $\omega = 0.x_1x_2$ then we can do the following

$$\frac{|0\rangle + e^{2\pi i(0.x_2)}|1\rangle}{\sqrt{2}}$$

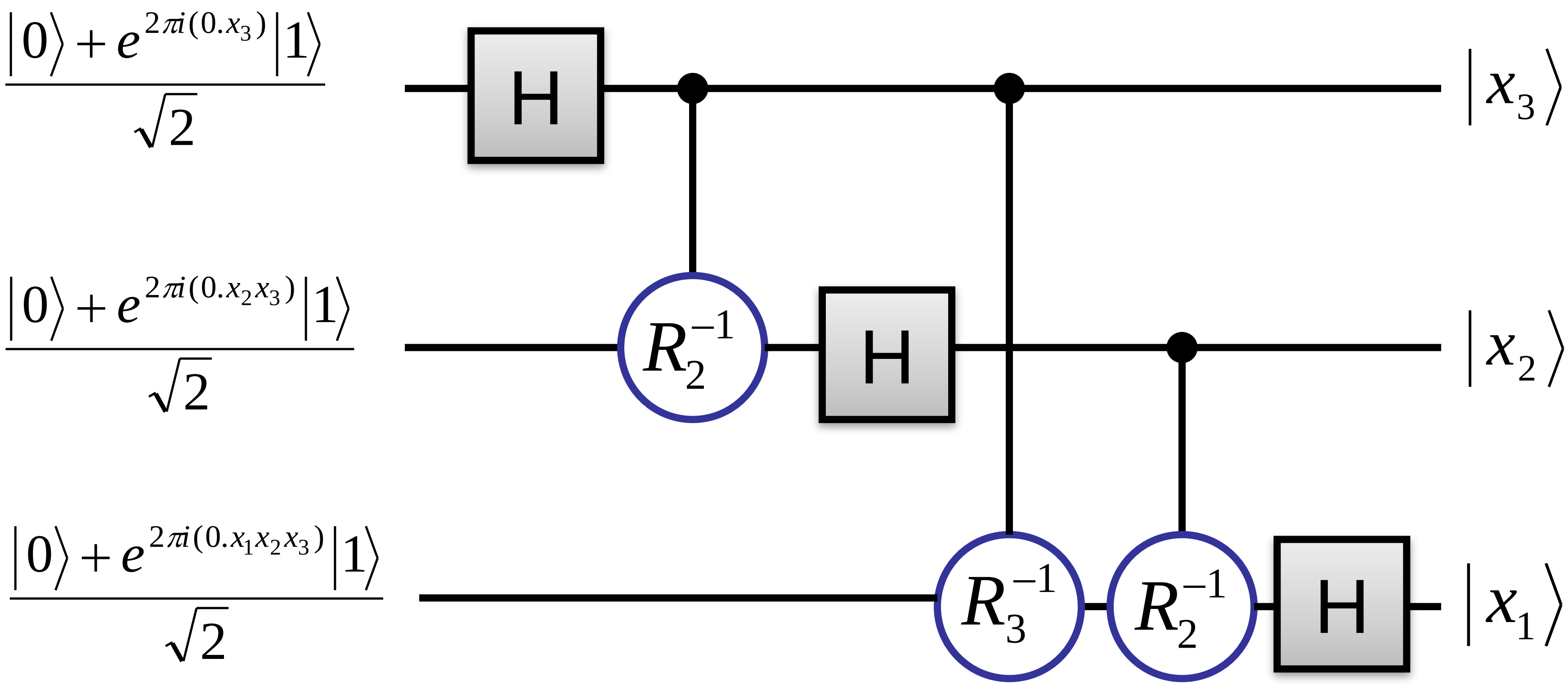
$$\frac{|0\rangle + e^{2\pi i(0.x_1x_2)}|1\rangle}{\sqrt{2}}$$



$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$$

Quantum phase estimation

- So if $\omega = 0.x_1x_2x_3$ then we can do the following



Quantum phase estimation

- Generalizing this network (and reversing the order of the qubits at the end) gives us a network with $O(n^2)$ gates that implements

$$\sum_{y=0}^{2^n-1} e^{2\pi i \frac{x}{2^n} y} |y\rangle \mapsto |x\rangle$$

Discrete Fourier transform

- The discrete Fourier transform maps vectors of dimension *N* by transforming the xth elementary vector according to

$$(0,0,\ldots,0,1,0,\ldots 0) \mapsto (1, e^{2\pi i \frac{x}{N}}, e^{2\pi i \frac{2x}{N}}, \ldots, e^{2\pi i \frac{(N-1)x}{N}})$$

- The quantum Fourier transform maps vectors in a Hilbert space of dimension *N* according to

$$|x\rangle \mapsto \sum_{y=0}^{N-1} e^{2\pi i \frac{x}{N} y} |y\rangle$$

Discrete Fourier transform

- Thus we have illustrated how to implement (the inverse of) the quantum Fourier transform in a Hilbert space of dimension 2^n

Estimating arbitrary ω ϵ [0,1)

- What if ω is not necessarily of the form $\frac{x}{2^n}$ for some integer *x* ?

- The QFT will map

$$\sum_{x=0}^{2^n-1} e^{2\pi i \omega z} |z\rangle$$

to a superposition

$$|\tilde{\omega}\rangle = \sum_y \alpha_y |y\rangle$$

where

$$\Pr ob\left(\left|\frac{y}{N} - \omega\right| \leq \frac{1}{N}\right) \geq \frac{8}{\pi^2} \qquad |\alpha_y| \in O\left(\frac{1}{\left|\frac{y}{N} - \omega\right|}\right)$$

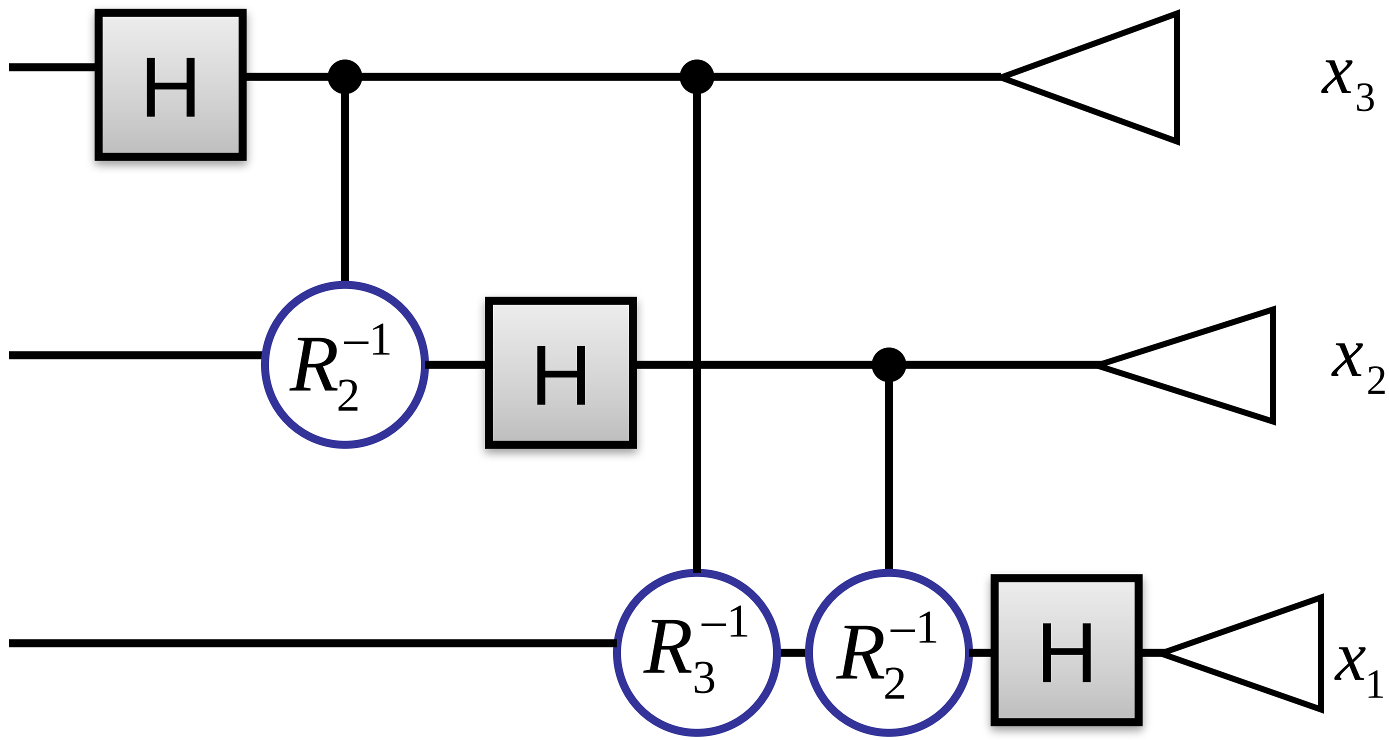
Quantum phase estimation

- For any real $\omega \in [0,1)$

$$\frac{|0\rangle + e^{2\pi i(4\omega)}|1\rangle}{\sqrt{2}}$$

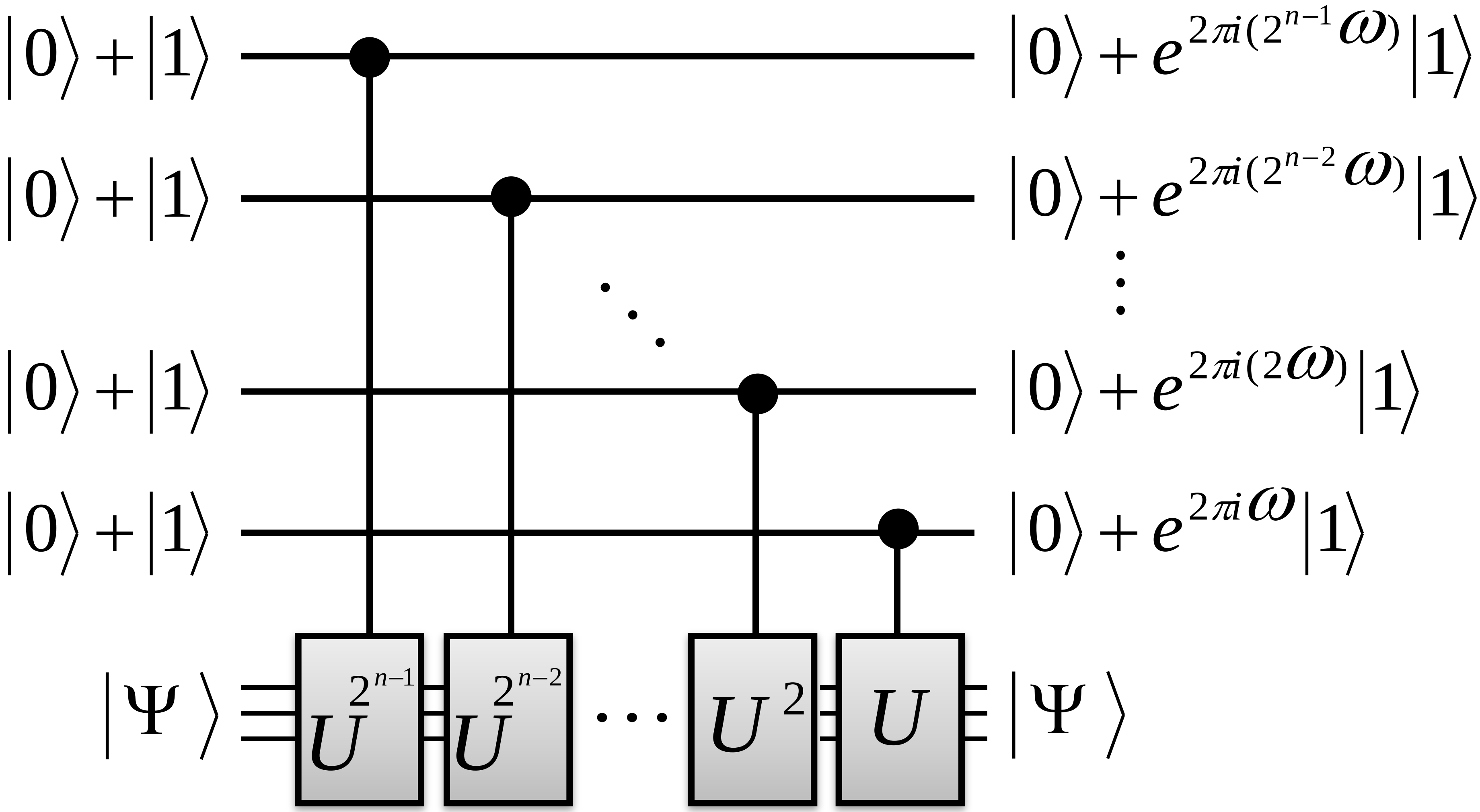
$$\frac{|0\rangle + e^{2\pi i(2\omega)}|1\rangle}{\sqrt{2}}$$

$$\frac{|0\rangle + e^{2\pi i(\omega)}|1\rangle}{\sqrt{2}}$$

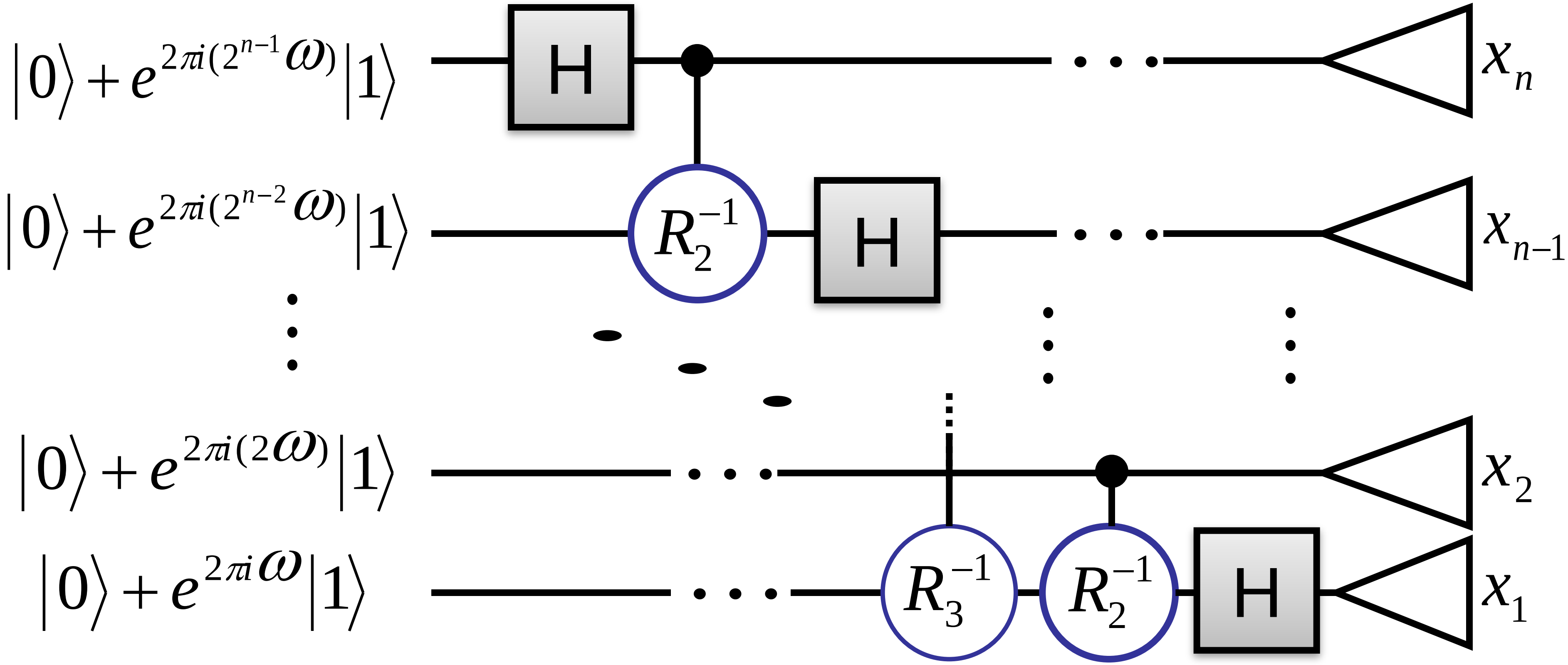


- With high probability $\frac{4x_1 + 2x_2 + x_3}{8} \approx \omega$

Eigenvalue kick-back

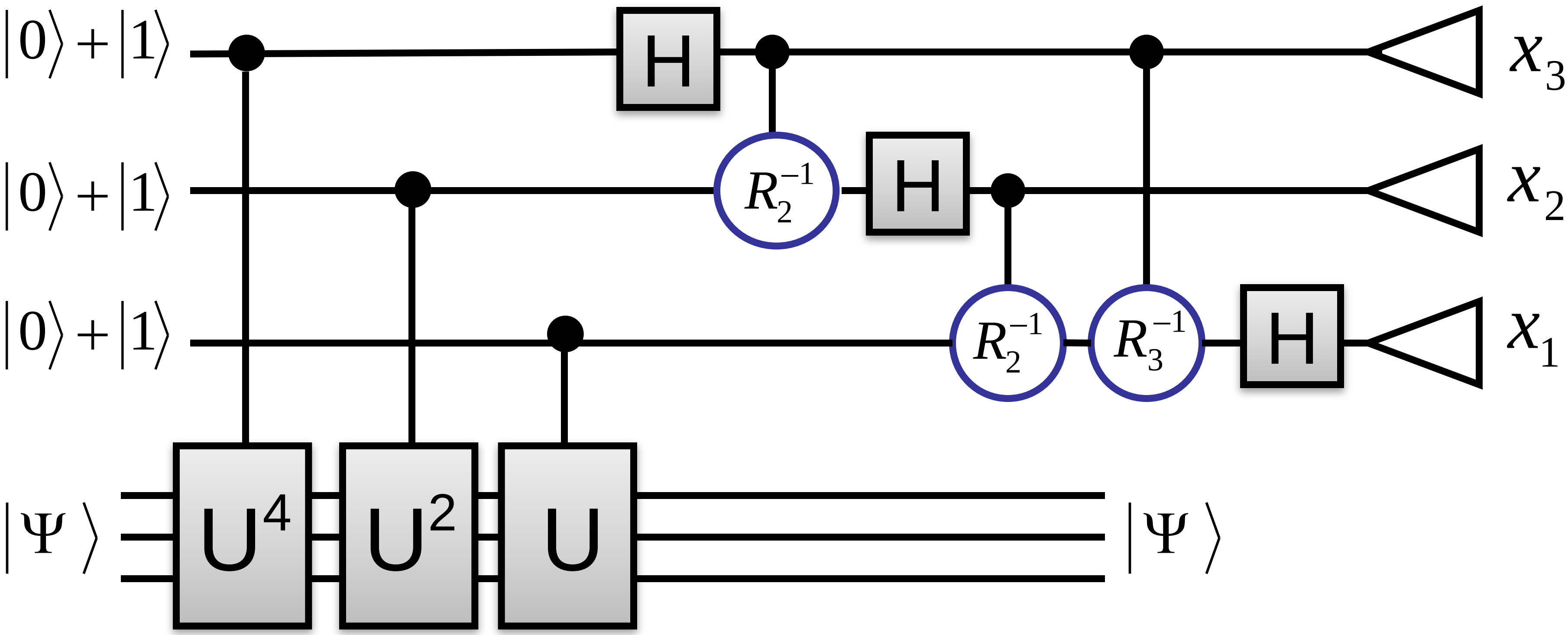


Phase estimation

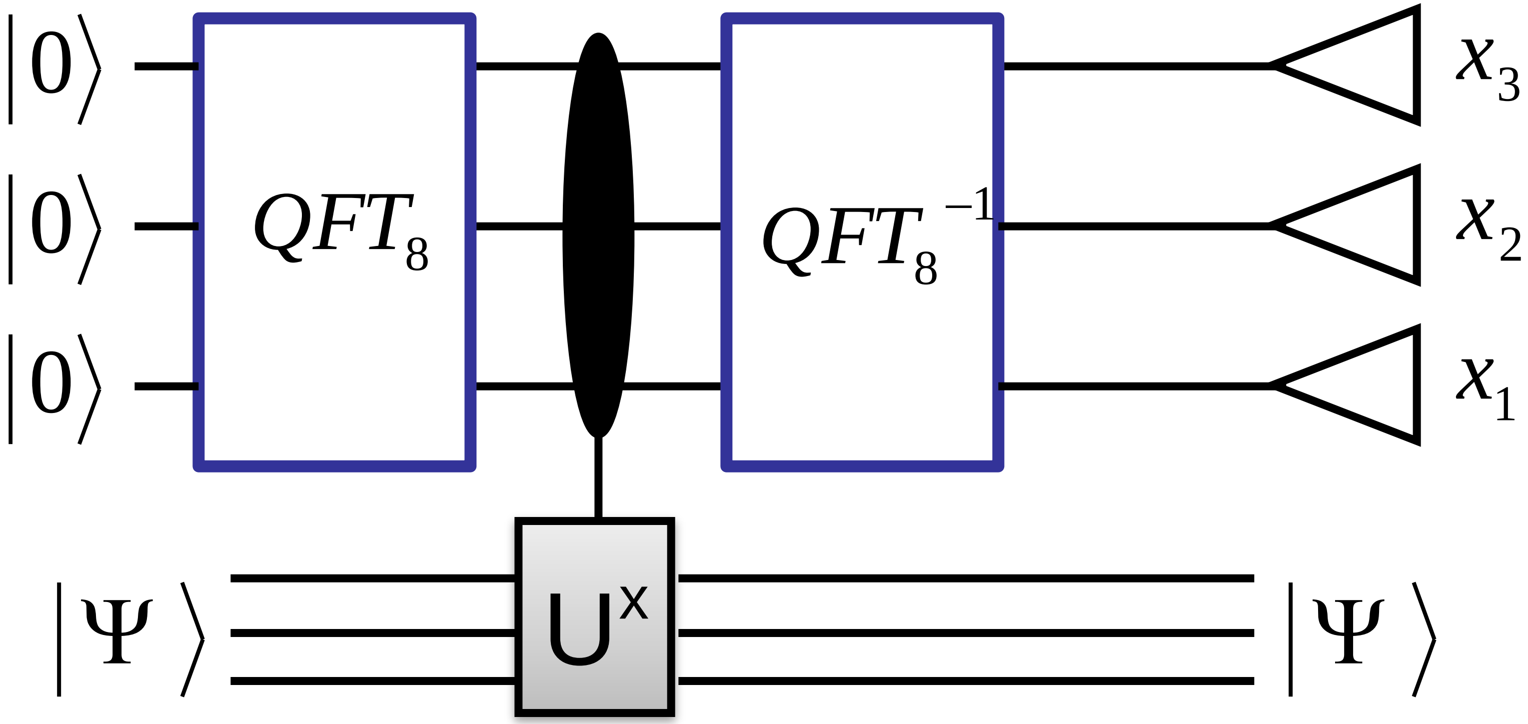


$$\frac{2^{n-1}x_1 + 2^{n-2}x_2 + \cdots x_n}{2^n} \approx \omega$$

Eigenvalue estimation



Eigenvalue estimation



Eigenvalue estimation

- Given **U** with eigenvector $|\Psi\rangle$ and eigenvalue $e^{2\pi i\omega}$, we thus have an algorithm that maps

$$|0\rangle|\Psi\rangle \rightarrow |\tilde{\omega}\rangle|\Psi\rangle$$

Eigenvalue estimation

- Given **U** with eigenvectors $|\Psi_k\rangle$ and respective eigenvalues $e^{2\pi i\omega_k}$ we thus have an algorithm that maps

$$|0\rangle|\Psi_k\rangle \to |\tilde{\omega}_k\rangle|\Psi_k\rangle$$

and therefore

$$|0\rangle\sum_k \alpha_k|\Psi_k\rangle = \sum_k \alpha_k|0\rangle|\Psi_k\rangle \to \sum_k \alpha_k|\tilde{\omega}_k\rangle|\Psi_k\rangle$$

Eigenvalue estimation

- Measuring the first register of

$$\sum_k \alpha_k \left|\tilde{\omega}_k\right\rangle \left|\Psi_k\right\rangle$$

is equivalent to measuring $\left|\tilde{\omega}_k\right\rangle$ with probability $\left|\alpha_k\right|^2$.

Eigenvectors

We know the eigenvalues of U_a are of the form

$$e^{2\pi i \frac{k}{r}}$$

$$U_a|\psi_k\rangle = \mathrm{e}^{\mathrm{i}2\pi\frac{\mathrm{k}}{r}}|\psi_k\rangle$$

$$|\psi_k\rangle = \sum_{j=0}^{r-1} e^{-i2\pi j\frac{k}{r}}|a^j\rangle$$

Checking the eigenvalues

$$U_a|\psi_k\rangle = \sum_{j=0}^{r-1} e^{-i2\pi j\frac{k}{r}} U_a|a^j\rangle$$

$$= \sum_{j=0}^{r-1} e^{-i2\pi j\frac{k}{r}}|a^{j+1}\rangle = e^{i2\pi\frac{k}{r}}\left(\sum_{j=1}^{r} e^{-i2\pi j\frac{k}{r}}|a^j\rangle\right)$$

$$= \mathrm{e}^{i2\pi\frac{k}{r}}\left(\sum_{j=0}^{r-1} e^{-i2\pi j\frac{k}{r}}|a^j\rangle\right) = e^{i2\pi\frac{k}{r}}|\psi_k\rangle$$

Note that

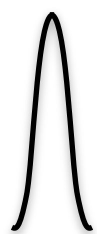
$$|1\rangle = \sum_{k=0}^{r-1} \frac{1}{\sqrt{r}} |\psi_k\rangle$$

Eigenvalue Estimation
Factoring Algorithm

$$|0\rangle|1\rangle \mapsto \sum_{k=0}^{r-1}\sum_{x}|x\rangle|\psi_k\rangle$$

$$\mapsto \sum_{k=0}^{r-1}\sum_{x} e^{2\pi i k x/r}|x\rangle|\psi_k\rangle$$

$$\Rightarrow \sum_{k} \Bigg(\qquad \Bigg) |\psi_k\rangle$$



$$\frac{k}{r}$$

Circuit for Eigenvalue Estimation Factoring Algorithm