SHOR'S ALGORITHM

Quantum Fourier Transform

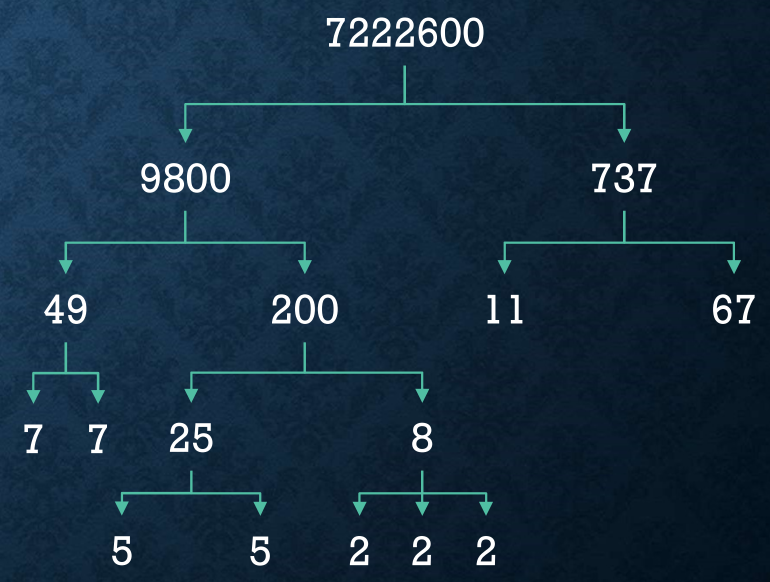
&

Quantum Phase Estimation

PROBLEM DESCRIPTION (1/2)

Integer factorization: Given a random number ***N*** decompose this number into a product of smaller integers.

Prime factorization: Given a random number ***N*** decompose this number into a product of prime numbers.



Example of integer factorization

Example of prime factorization

PROBLEM DESCRIPTION (2/2)

There is not yet any ‘classical’ algorithm that can factor all integers in polynomial time.

Polynomial time complexity definition: $O(b^k)$

$\boldsymbol{b}$: The number of bits used to represent the number N

$\boldsymbol{k}$: Some constant

Unsolved problem in computer science:
Can integer factorization be solved in polynomial time on a classical computer?

It is suspected that this problem is in class *NP*

Current best algorithm runtime: $e^{\left(\sqrt[3]{\frac{64}{9}}+O(1)\right)\cdot(\ln N)^{1/3}\cdot(\ln \mathrm{l} \qquad)^{2/3}}$

RSA-200:
27997833911221327870829467638722601621070446786955428537560009929326128400107
60934567105295536085606182235191095136578863710595448200657677509858055761357
909873495014417886317894629518723786922182398

The CPU time spent on finding RSA-200 factors is equivalent to 75 CPU years

A MATHEMATICAL OBSERVATION

For any number a that does not share any factors with N (a and N are coprime) we define $r = order_N(a)$ as the smallest integer such that $a^r \equiv 1\ (mod\ N)$

$$\text{Given that } r = order_N(a)$$
$$a^r \equiv 1\ (mod\ N)$$
$$a^r - 1 \equiv 0\ (mod\ N)$$
$$\left(a^{r/2} - 1\right) \cdot \left(a^{r/2} + 1\right) \equiv 0\ (mod\ N)$$
$$\left(a^{r/2} - 1\right) \cdot \left(a^{r/2} + 1\right) = k \cdot N$$

Problems:

1. r might be odd
2. $\left(a^{r/2} + 1\right)$ might be a multiple of N
3. r is extremely difficult to compute in a classical computer

It turns out that if a is picked uniformly at random in the range $[2, N - 1]$ then the probability of problems 1 and 2 not happening is about $\frac{3}{8} \approx 0.375$

After repeating this process 10 times the probability of failure is $\left(1 - \frac{3}{8}\right)^{10} \approx 0.009 \approx 1\%$

SHOR'S ALGORITHM

Procedure:

1. Pick a number a uniformly at random $a \in [2, N-1]$
2. Compute $K = gcd(a, N)$ using Euclidean algorithm
3. If $K \neq 1$ then K is a factor of N and we are done (very unlikely for large numbers)
4. **Use the quantum period finding subroutine to find $\mathbf{r = order_N(a)}$**
5. If r is odd, then repeat the process from step 1
6. If $\left(a^{r/2} \pm 1\right) \equiv 0 \; (mod\ N)$ then repeat again the process from step 1.
7. The factors of N are $gcd(\left(a^{r/2} + 1\right), N)$ and $gcd(\left(a^{r/2} - 1\right), N)$

Steps 1,2,3 are classically preprocessing steps

Steps 5, 6, 7 are classically post processing steps

Only step 4 runs on quantum computer

QUANTUM FOURIER TRANSFORM (1/10)

In quantum computing the Quantum Fourier Transform (QFT) is a linear transformation on qubits, and is the quantum analogue of the discrete Fourier transform

The DFT maps a vector $[x_0 \quad x_1 \quad \cdots \quad x_{N-1}]$ into another vector $[y_0 \quad y_1 \quad \cdots \quad y_{N-1}]$

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{2\pi i \frac{jk}{N}}$$

The quantum Fourier transform acts on a quantum state $|X\rangle$ and maps it to the quantum state $|Y\rangle$

$$|X\rangle = \sum_{j=0}^{N-1} x_j \cdot |j\rangle \qquad |Y\rangle = \sum_{k=0}^{N-1} y_k \cdot |k\rangle \qquad y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{2\pi i \frac{jk}{N}}$$

QUANTUM FOURIER TRANSFORM (2/10)

The unitary operator of the QFT is: $U_{QFT} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} e^{2\pi i \frac{jk}{N}} \cdot |k\rangle\langle j|$

$$U_{QFT} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix} \qquad \omega = e^{\frac{2\pi i}{N}}$$

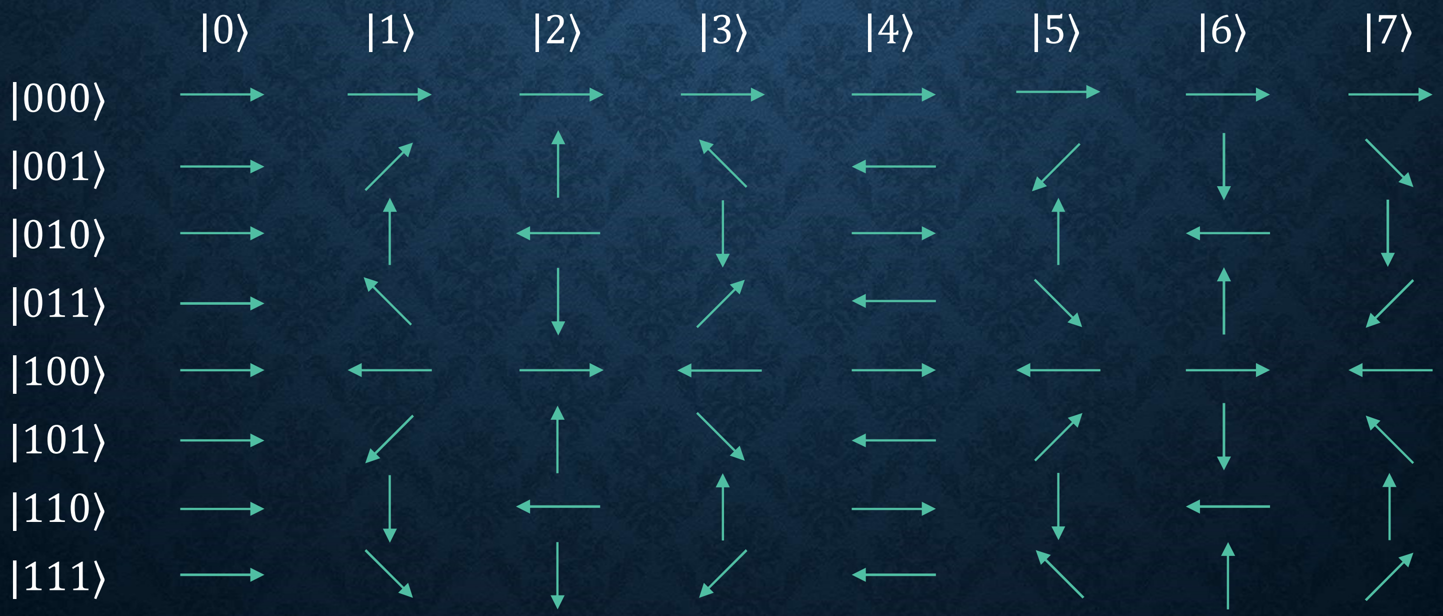
$$U_{QFT}|x\rangle = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^x & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2x} & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3x} & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{x(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \rightarrow |x\rangle \\ \vdots \\ 0 \end{bmatrix} = \frac{1}{\sqrt{N}} \cdot \begin{bmatrix} 1 \\ \omega^x \\ \omega^{2x} \\ \omega^{3x} \\ \vdots \\ \omega^{x(N-1)} \end{bmatrix}$$

The probabilities of each state in the Fourier basis are the same and equal to $\frac{1}{N}$

QUANTUM FOURIER TRANSFORM (3/10)

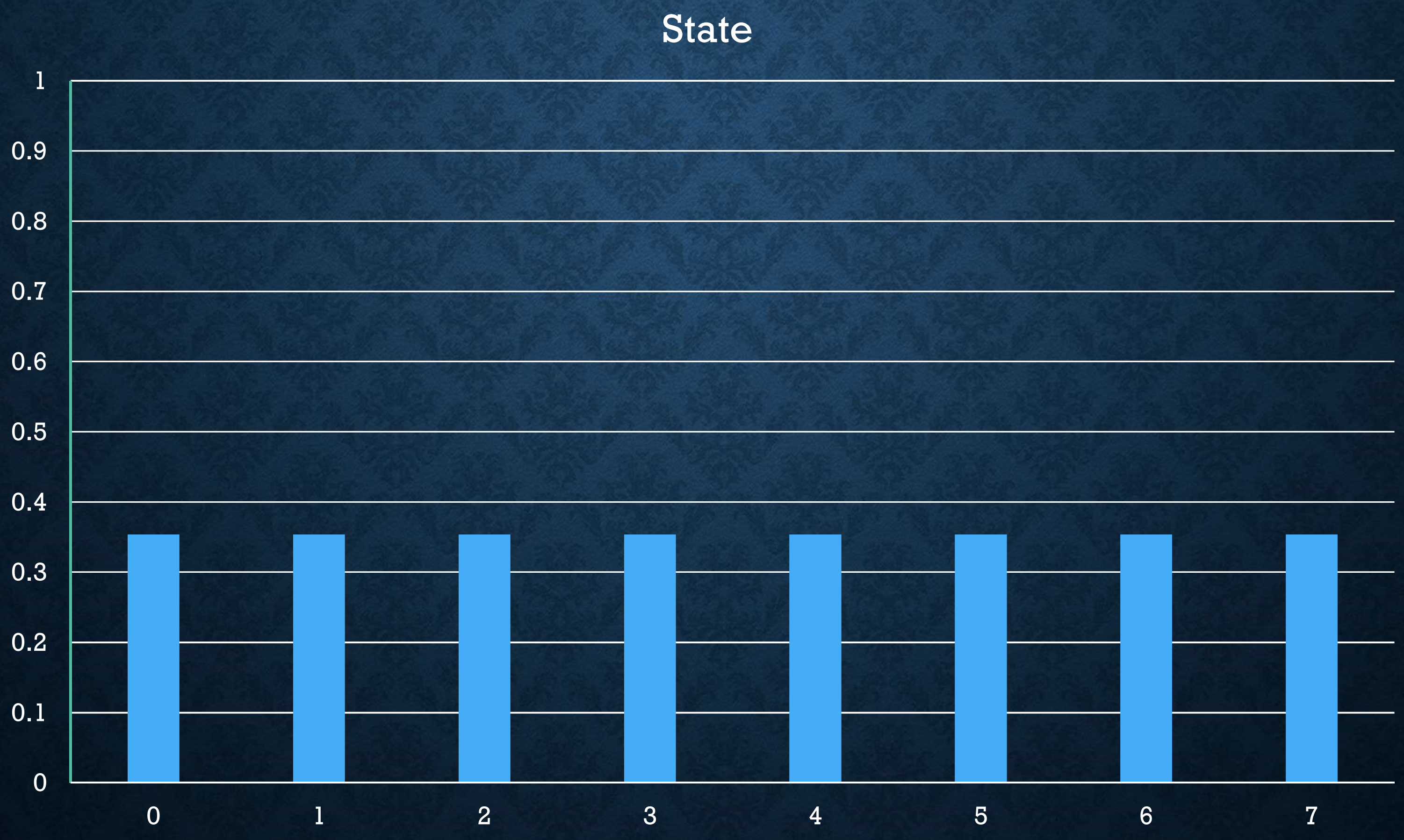
Example of QFT for 3 qubits

$$U_{QFT} = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & e^{\frac{i\pi}{4}} & i & e^{\frac{3i\pi}{4}} & -1 & e^{\frac{5i\pi}{4}} & -i & e^{\frac{7i\pi}{4}} \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & e^{\frac{3i\pi}{4}} & -i & e^{\frac{i\pi}{4}} & -1 & e^{\frac{7i\pi}{4}} & i & e^{\frac{5i\pi}{4}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & e^{\frac{5i\pi}{4}} & i & e^{\frac{7i\pi}{4}} & -1 & e^{\frac{i\pi}{4}} & -i & e^{\frac{3i\pi}{4}} \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & e^{\frac{7i\pi}{4}} & -i & e^{\frac{5i\pi}{4}} & -1 & e^{\frac{3i\pi}{4}} & i & e^{\frac{i\pi}{4}} \end{bmatrix}$$



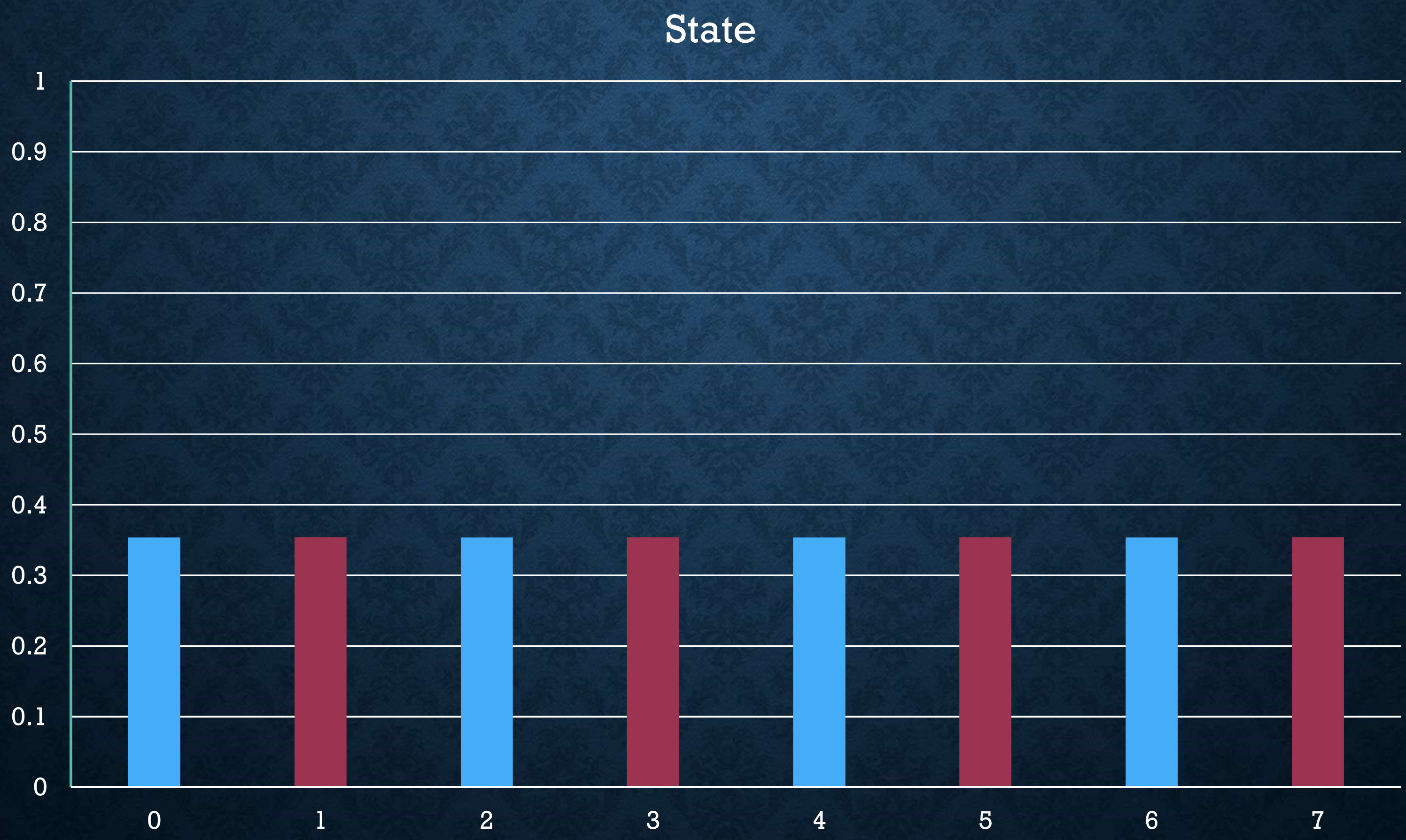
QUANTUM FOURIER TRANSFORM (4/10)

The state vector of $QFT|0\rangle$



QUANTUM FOURIER TRANSFORM (5/10)

The state vector of $QFT|4\rangle$



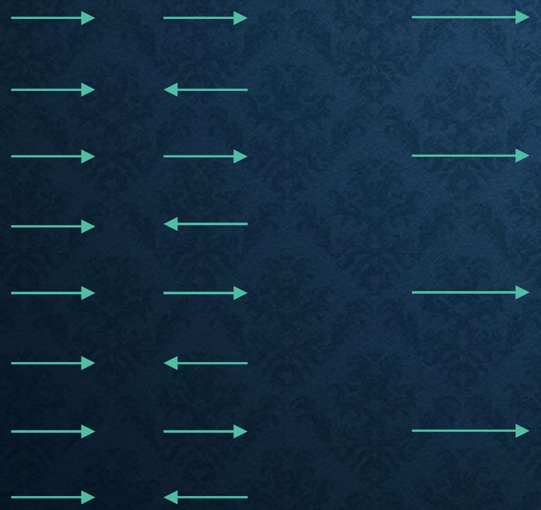
QUANTUM FOURIER TRANSFORM (6/10)

An important property of the Quantum Fourier Transform

$$QFT(|x\rangle + |y\rangle) = QFT|x\rangle + QFT|y\rangle$$

Example: $QFT(|0\rangle + |4\rangle)$

$|0\rangle$ $|4\rangle$ $|0\rangle + |4\rangle$

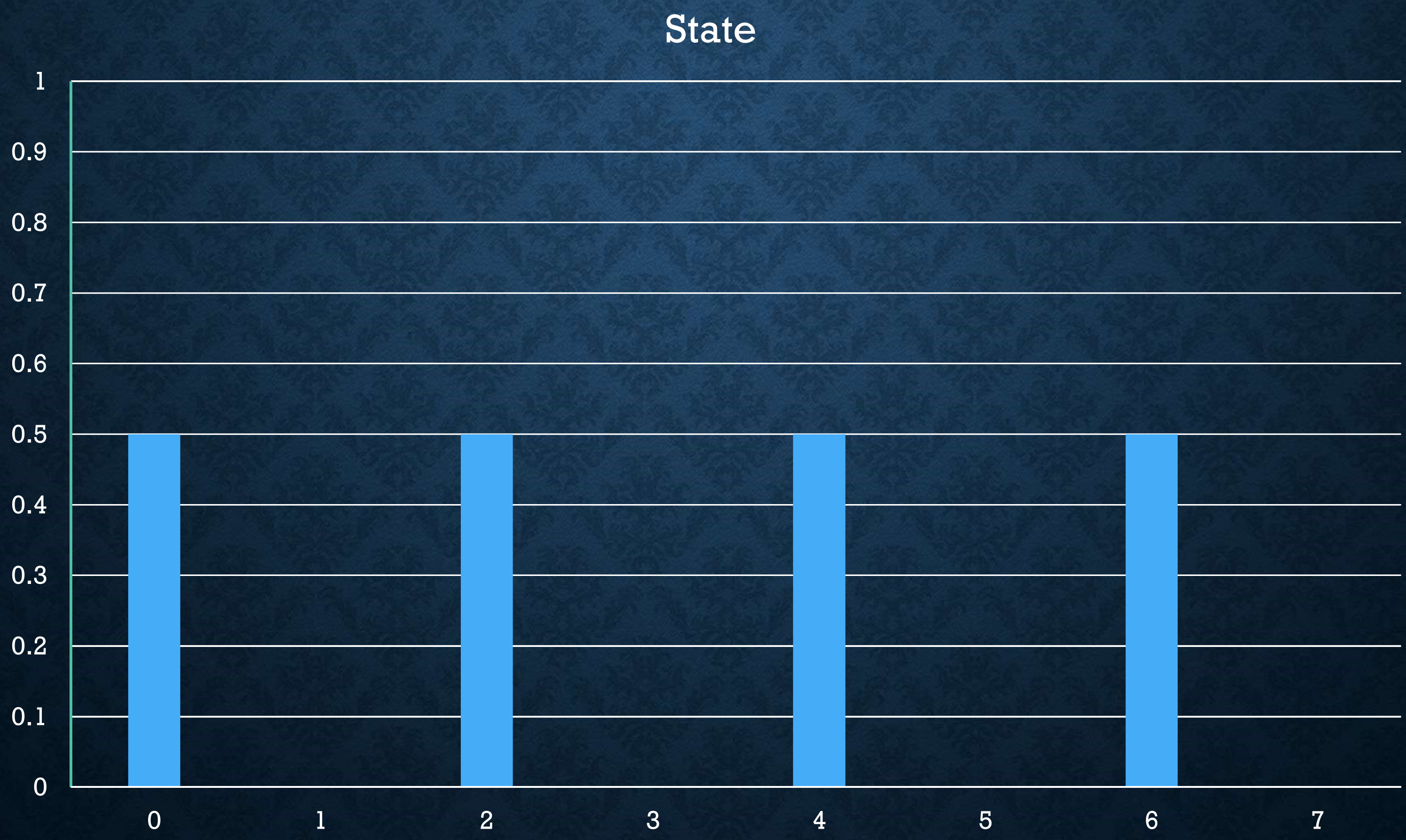


Constructive Interference: States 0, 2, 4, 6

Destructive Interference: States 1, 3, 5, 7

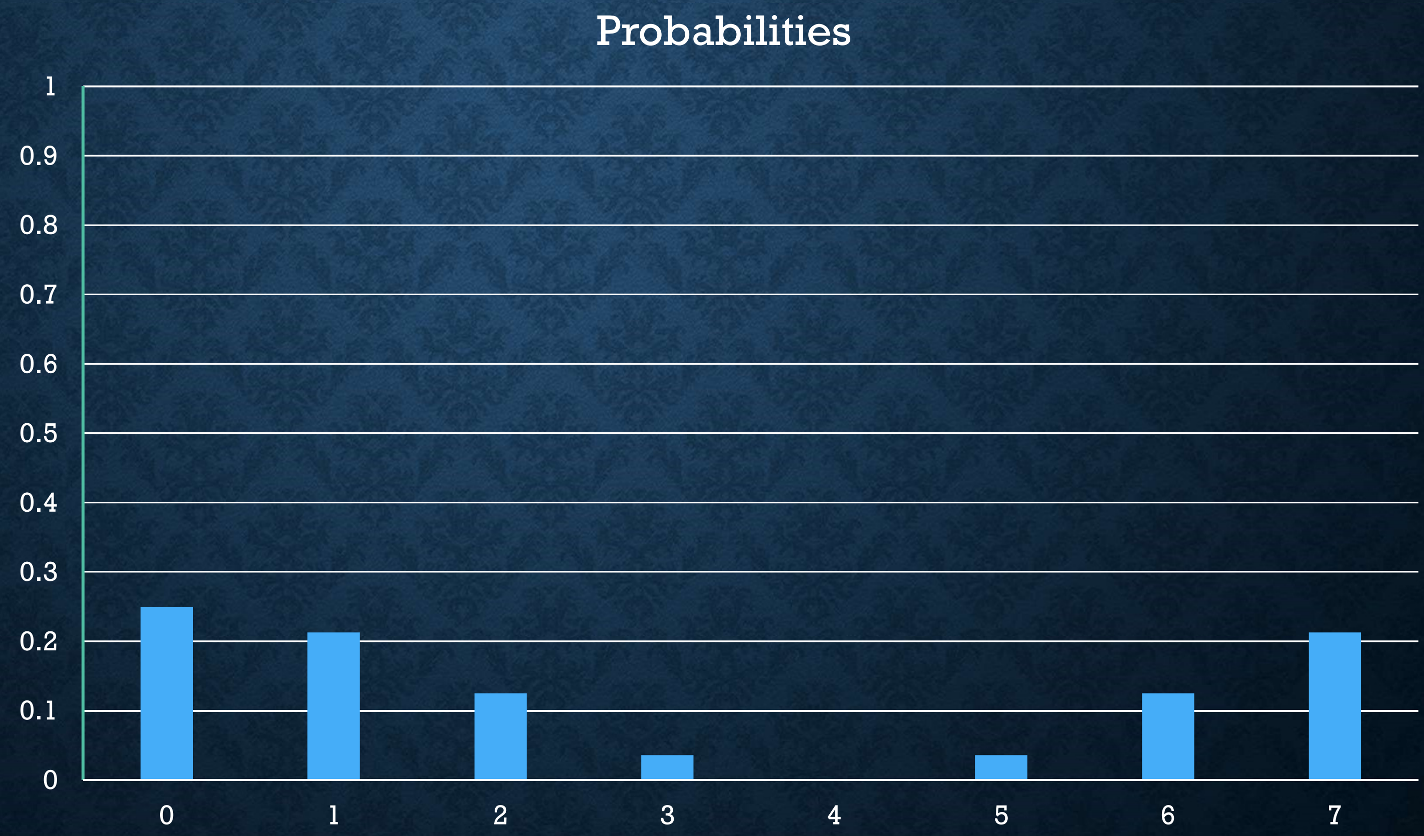
QUANTUM FOURIER TRANSFORM (7/10)

The state vector of the $QFT(|0\rangle + |4\rangle)$



QUANTUM FOURIER TRANSFORM (8/10)

The state vector of the $QFT(|0\rangle + |1\rangle)$



QUANTUM FOURIER TRANSFORM (9/10)

The QFT circuit is classically computable in $O(n^2)$ time, where n is the number of qubits

The total number of elementary quantum gates that are used to implement the QFT circuit is $O(n^2)$

The classical FFT runs in $O(NlogN)$, where N is 2^n

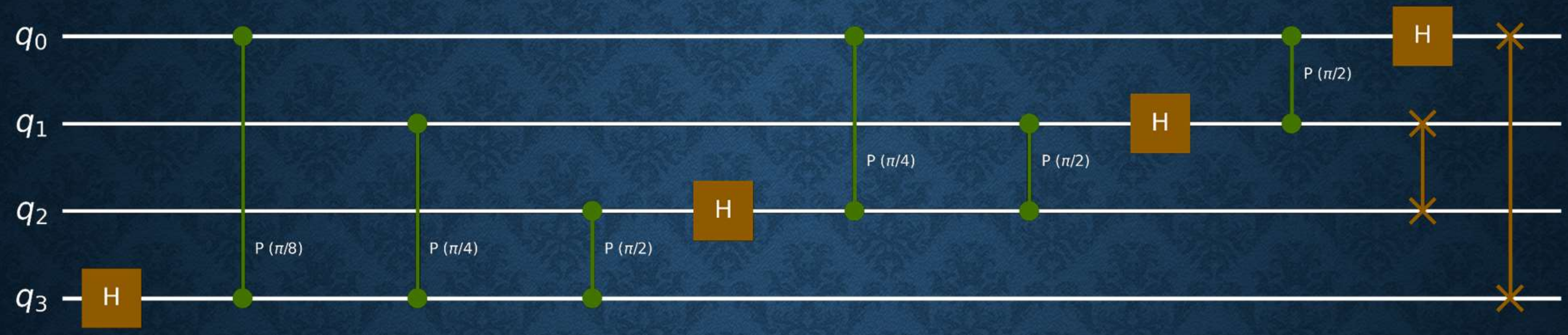
This is an exponational speedup in time complexity!! However…

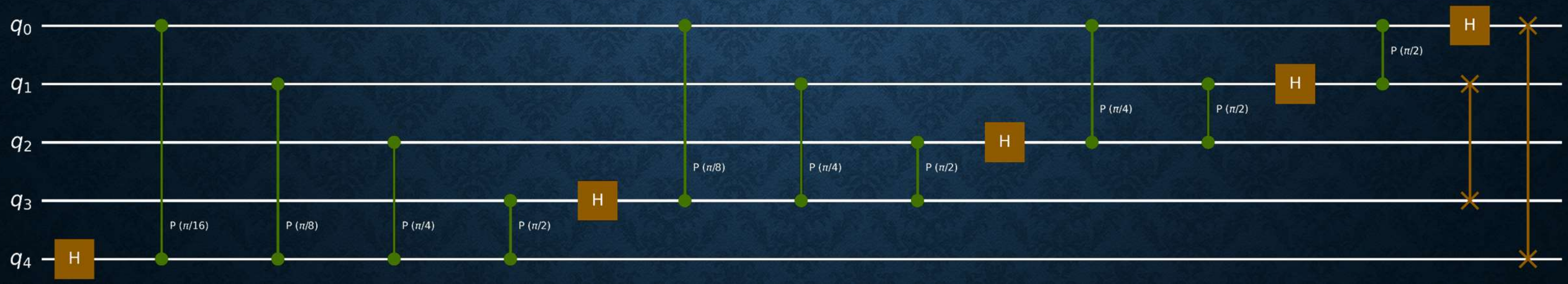
Measuring the output state collapses superposition

Time complexity if we want to parse the information of the QFT state: $O(Nn^2) = O(Nlog(N)^2)$

QUANTUM FOURIER TRANSFORM (10/10)

The circuit



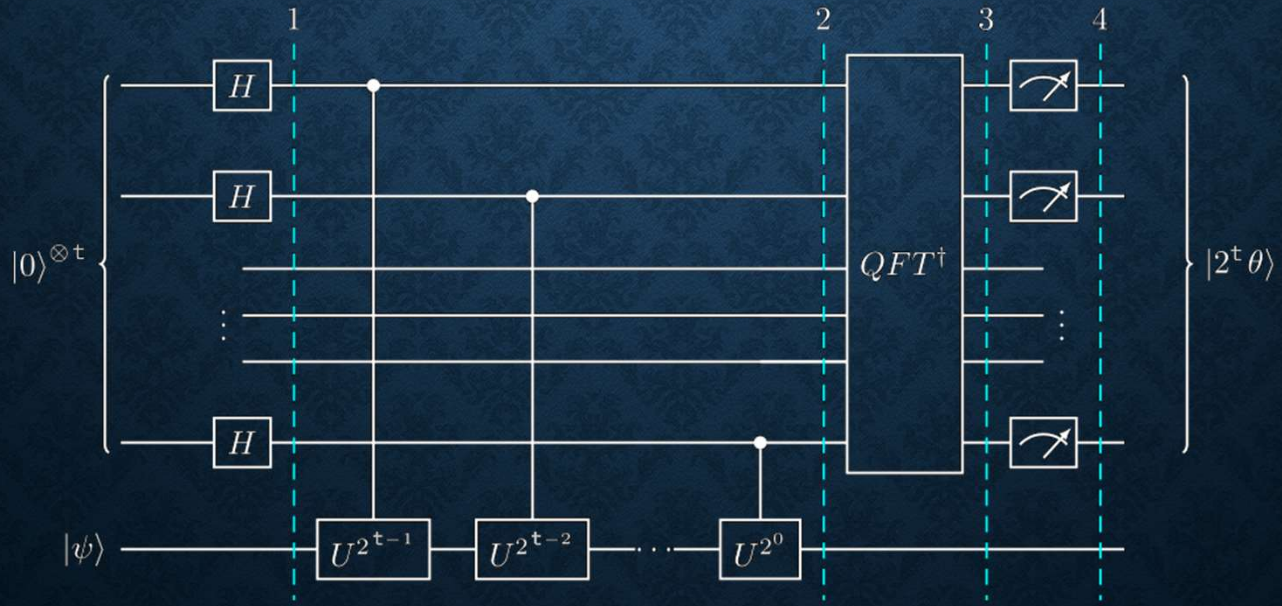


QUANTUM PHASE ESTIMATION (1/8)

Quantum phase estimation is one of the most important subroutines in quantum computation

Given a unitary operator U the algorithm estimates θ in $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$

The algorithm uses phase kickback to write the phase of U (in the Fourier basis) to the t counting registers.



QUANTUM PHASE ESTIMATION (2/8)

Applying the controlled U operation many times results in:

$$U^{2^j}|\psi\rangle = U^{2^j-1}U|\psi\rangle = U^{2^j-1}e^{2\pi i\theta}|\psi\rangle$$

It can be proven by induction that: $U^{2^j}|\psi\rangle = e^{2\pi i\theta 2^j}|\psi\rangle$

The controlled U operation results in:

$$|0\rangle \otimes |\psi\rangle + |1\rangle \otimes e^{2\pi i\theta}|\psi\rangle = \left(|0\rangle + e^{2\pi i\theta}|1\rangle\right) \cdot |\psi\rangle$$

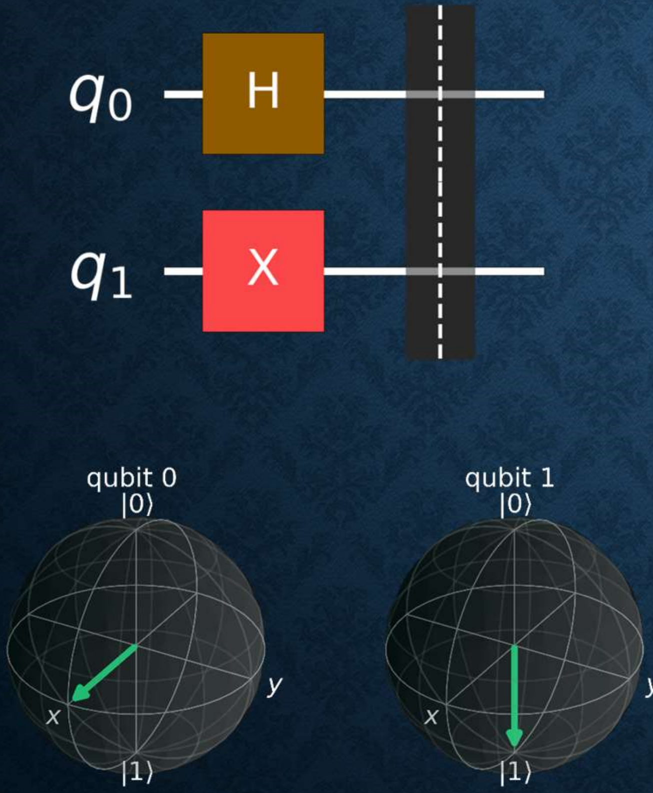
Applying all the controlled U operations results in the state:

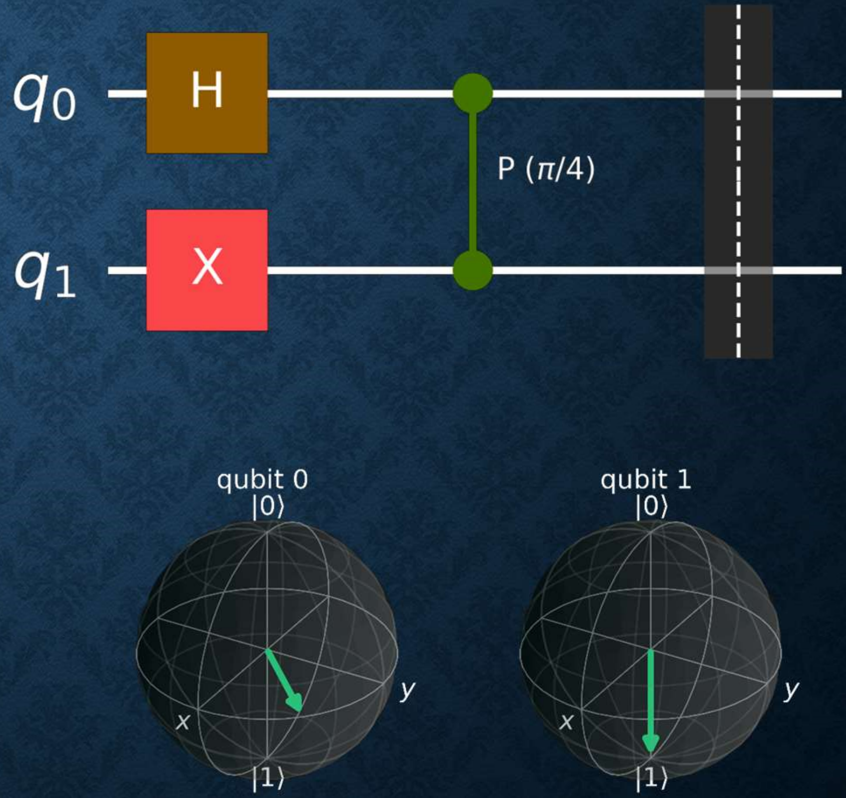
$$\frac{1}{\sqrt[2]{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i\theta k}|k\rangle$$

This looks a lot like Fourier Transform.

QUANTUM PHASE ESTIMATION (3/8)

Remember… Phase Kickback





QUANTUM PHASE ESTIMATION (4/8)

The number of counting qubits t that we use depends on the accuracy of the measurement we want.

Example of 3 bit precision

T gate: $\begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}$

Notice that $T|1\rangle = e^{\frac{i\pi}{4}}|1\rangle$

$\theta = \frac{1}{8} = 0.001$ in binary.

3 counting qubits for maximum precision

Example of 2 bit precision

S gate: $\begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{bmatrix}$

Notice that $S|1\rangle = e^{\frac{i\pi}{2}}|1\rangle$

$\theta = \frac{1}{4} = 0.01$ in binary.

2 counting qubits for maximum precision

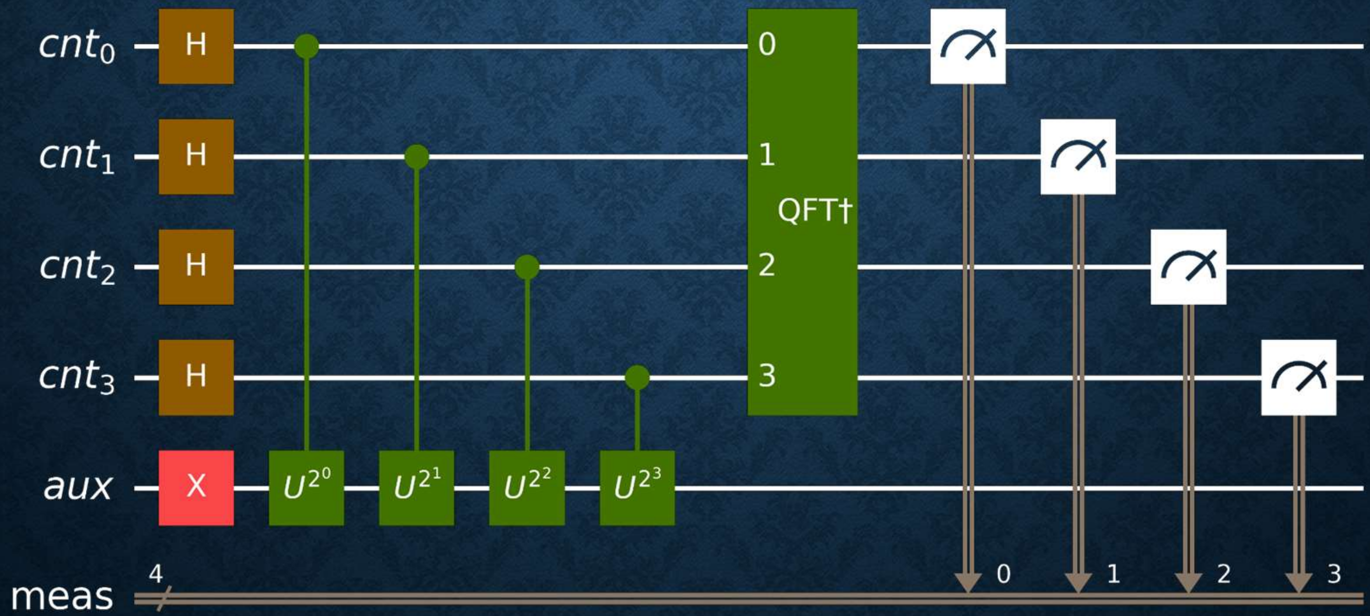
Notice: if θ MANTISA cannot store all the information the measurement will always be somewhat imprecise. Example $\theta = \frac{1}{3}$

QUANTUM PHASE ESTIMATION (5/8)

Estimating $\theta = \frac{5}{16}$

Notice that $\theta = \frac{5}{16} = 0.3125 = 0.0101$ (in binary)

This means that 4 qubits are needed to have maximum precision



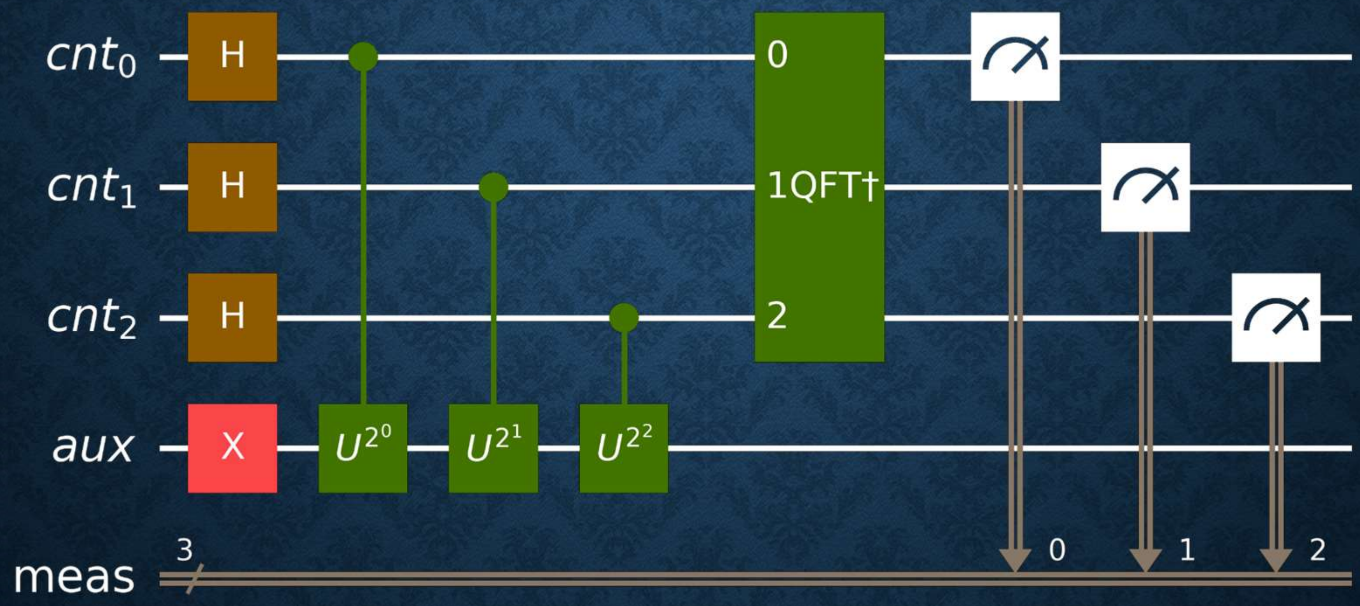
QUANTUM PHASE ESTIMATION (6/8)

Estimating $\theta = \frac{5}{16}$ using 4 counting qubits.

QUANTUM PHASE ESTIMATION (7/8)

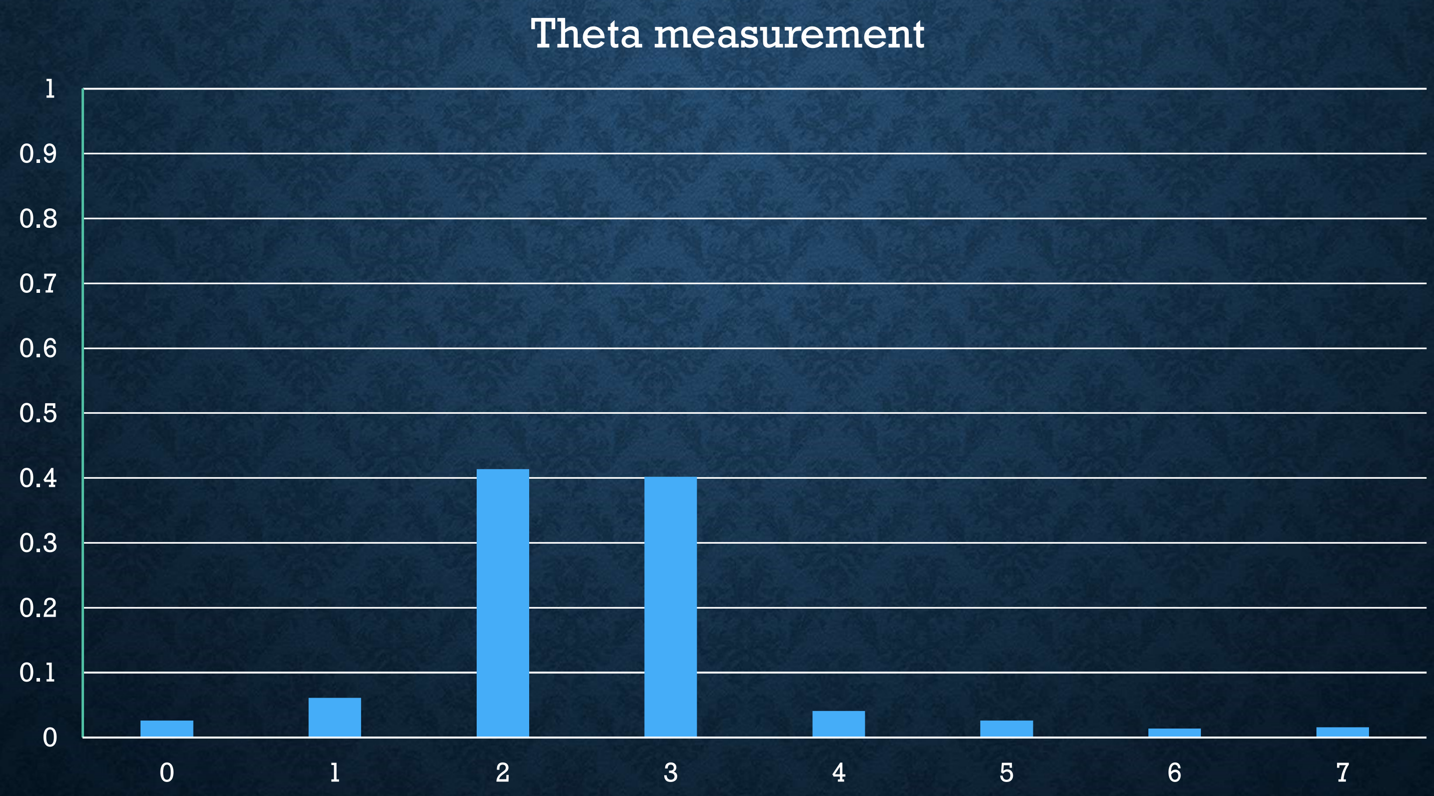
Estimating $\theta = \frac{5}{16}$ using 3 counting qubits.

This is the same as $\theta = \frac{2.5}{8}$



QUANTUM PHASE ESTIMATION (8/8)

Estimating $\theta = \frac{5}{16}$ using 3 counting qubits.



BACK TO SHOR'S ALGORITHM

Remember… Our goal is to find $\mathbf{r} = \mathbf{order_N(a)}$ where N is the number we want to factor and a is a random guess

We need to calculate $f(x) = a^x \; mod \; N$

This function is periodic

For a random number s we have: $f(s) = a^s \; mod \; N$

If we plug r into the function f we have: $f(r) = 1$

$$f(s) \cdot f(r) = a^{s+r} \; mod \; N = a^s \; mod \; N$$

Since s was picked at random, the function is periodic.

BACK TO SHOR'S ALGORITHM

We start by creating a superposition of all the possible powers (these are the counting qubits)

$$|x\rangle = |0\rangle + |1\rangle + |2\rangle + \cdots + |2^{2n} - 2\rangle + |2^{2n} - 1\rangle$$

The actual state is $|x\rangle = \frac{1}{\sqrt{2^{2n}}} \sum_{j=0}^{2^{2n}} |j\rangle$

Then we create an entangled state of the power $|x\rangle$ and the remainder $|a^x modN\rangle$

Since the function $a^x modN$ is periodic, measuring the remainder collapses the superposition of the powers into a superposition of only the powers that would generate the measured remainder.

Since the function f is periodic the states of the superposition after the measurement are r states apart, where r is the order modulo N

But we don't have to do this…

BACK TO SHOR'S ALGORITHM

It turns out that using the quantum phase estimation algorithm the period r is kicked back in the counting registers.

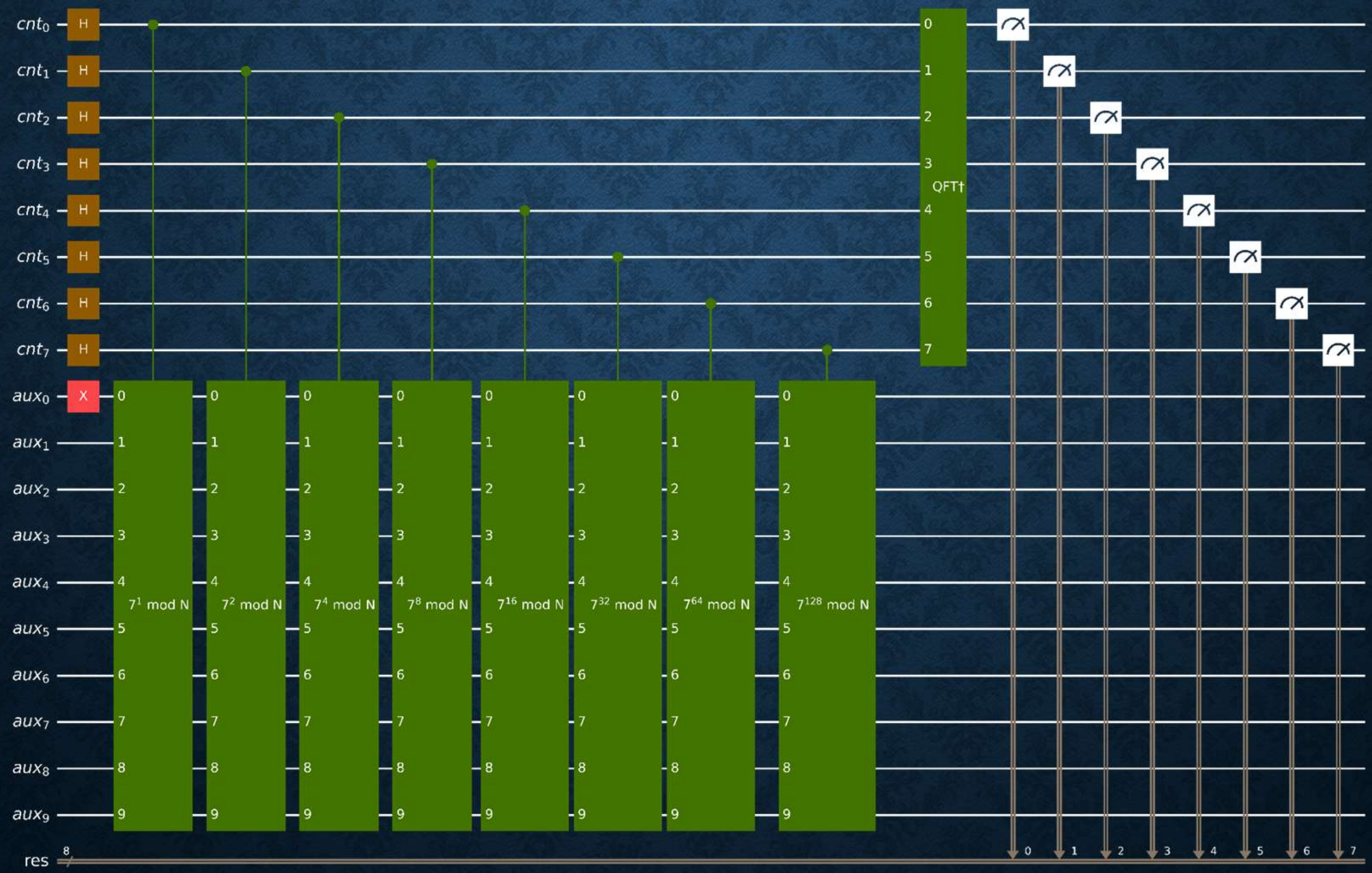
We perform inverse Quantum Fourier Transform to find that period.

After the inverse QFT the state of the counting qubits is left in a superposition of multiples of $|\frac{1}{r}\rangle$

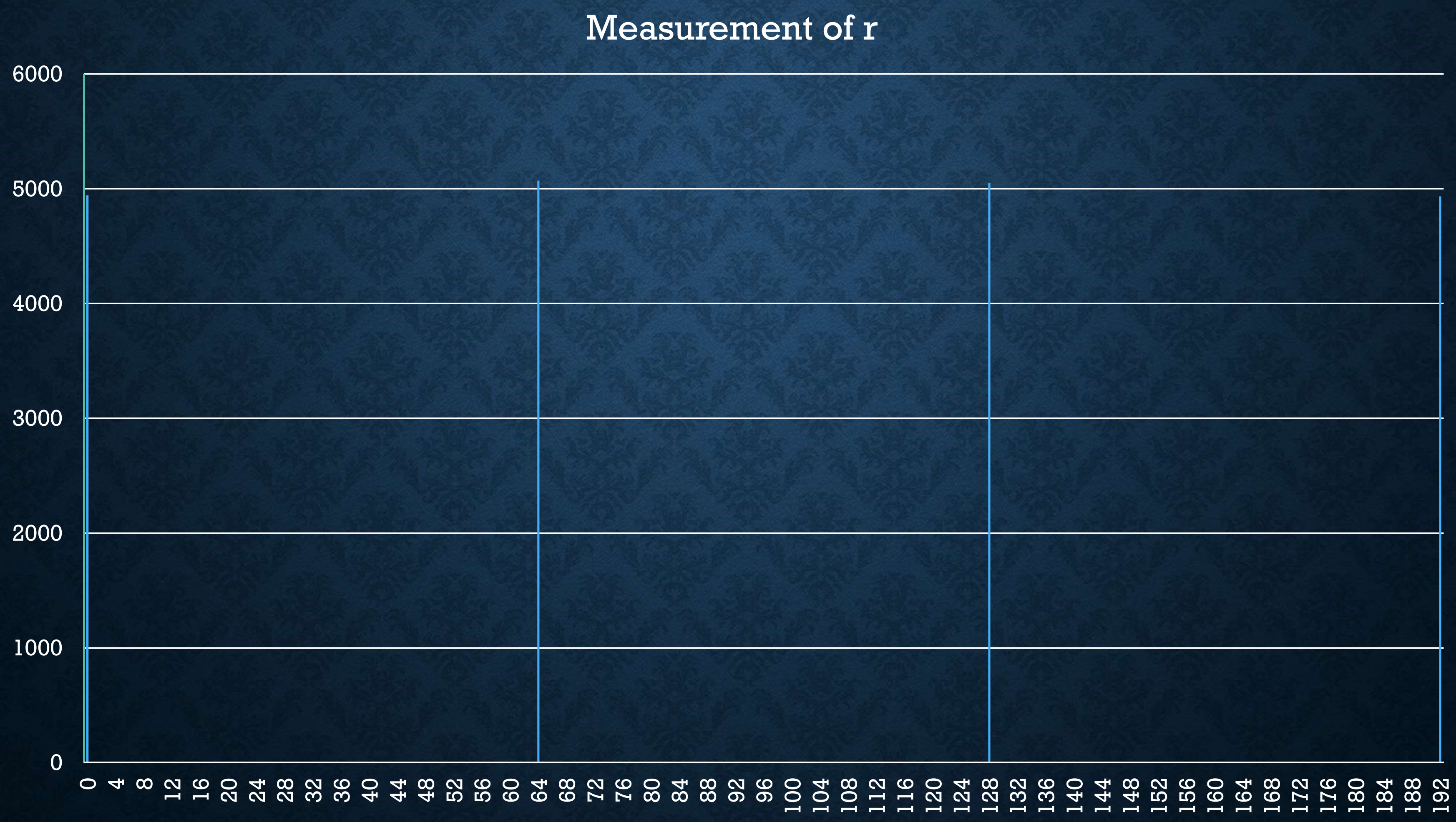
EXAMPLE $N = 15$

First step is to randomly pick a number a

Let's say $a = 7$



EXAMPLE $N = 15$ AND $a = 7$



EXAMPLE $N = 15$ AND $a = 7$

The result should be multiples of $\frac{1}{r}$ with 8 bit precision.

Measurements of the final state: $\frac{0}{256}, \frac{64}{256}, \frac{128}{256}, \frac{192}{256}$

Simplifying the fractions we get: $\frac{0}{1}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$

Therefore $r = 4$

Better solutions that may share factors with N are:

$$p' = 7^2 + 1 = 50$$

$$q' = 7^2 - 1 = 48$$

Performing Euclidian algorithm of these improved guesses with the initial number we get:

$$p = gcd(50, 15) = 5$$

$$q = gcd(48, 15) = 3$$

EXAMPLE $N = 35$

First step is to randomly pick a number a

Let's say $a = 33$

FAST SQUARING ALGORITHM

Example for calculating $33^{2048}\ mod\ 35$

$33^1 \equiv 33\ (mod\ 35)$

$33^2 \equiv 4\ (mod\ 35)$

$33^4 \equiv 16\ (mod\ 35)$

$33^8 \equiv 11\ (mod\ 35)$

$33^{16} \equiv 16\ (mod\ 35)$

$33^{32} \equiv 11\ (mod\ 35)$

$33^{64} \equiv 16\ (mod\ 35)$

$33^{128} \equiv 11\ (mod\ 35)$

$33^{256} \equiv 16\ (mod\ 35)$

$33^{512} \equiv 11\ (mod\ 35)$

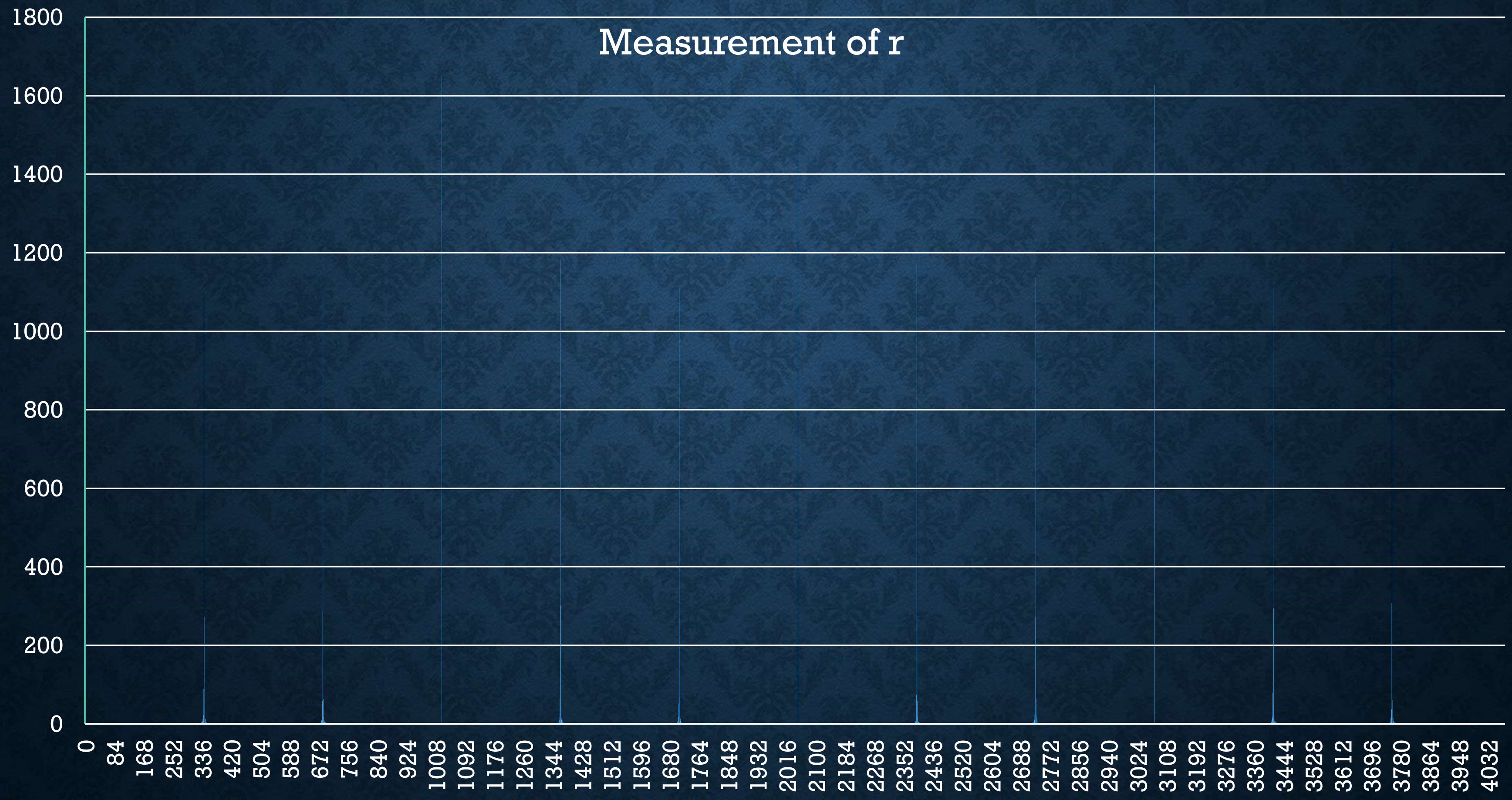
$33^{1024} \equiv 16\ (mod\ 35)$

$33^{2048} \equiv 11\ (mod\ 35)$

Time complexity $O(loglogN)$ if $N = 33^{2048}$

This is very efficient

EXAMPLE $N = 35$ AND $a = 33$



EXAMPLE $N = 35$ AND $a = 33$

The result should be multiples of $\frac{1}{r}$ with 12 bit precision.

Measured value of $r = 12$

Indeed $33^{12} \equiv 1 \ (mod\ 35)$

Better solutions that may share factors with N are:

$$p' = 33^6 + 1 = 1291467970$$

$$q' = 33^6 - 1 = 1291467968$$

Performing Euclidian algorithm of these improved guesses with the initial number we get:

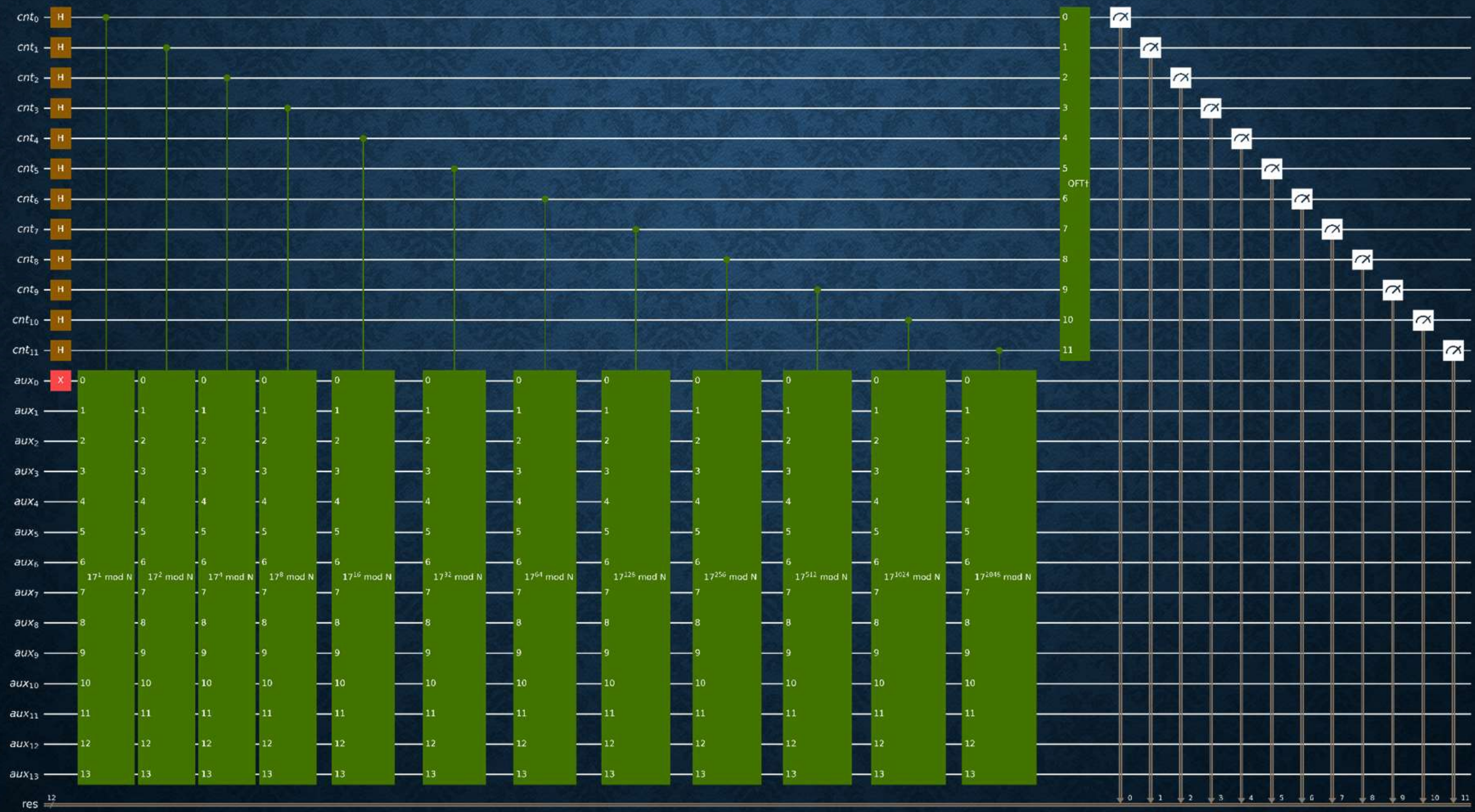
$$p = gcd(1291467970, 35) = 5$$

$$q = gcd(1291467968, 35) = 7$$

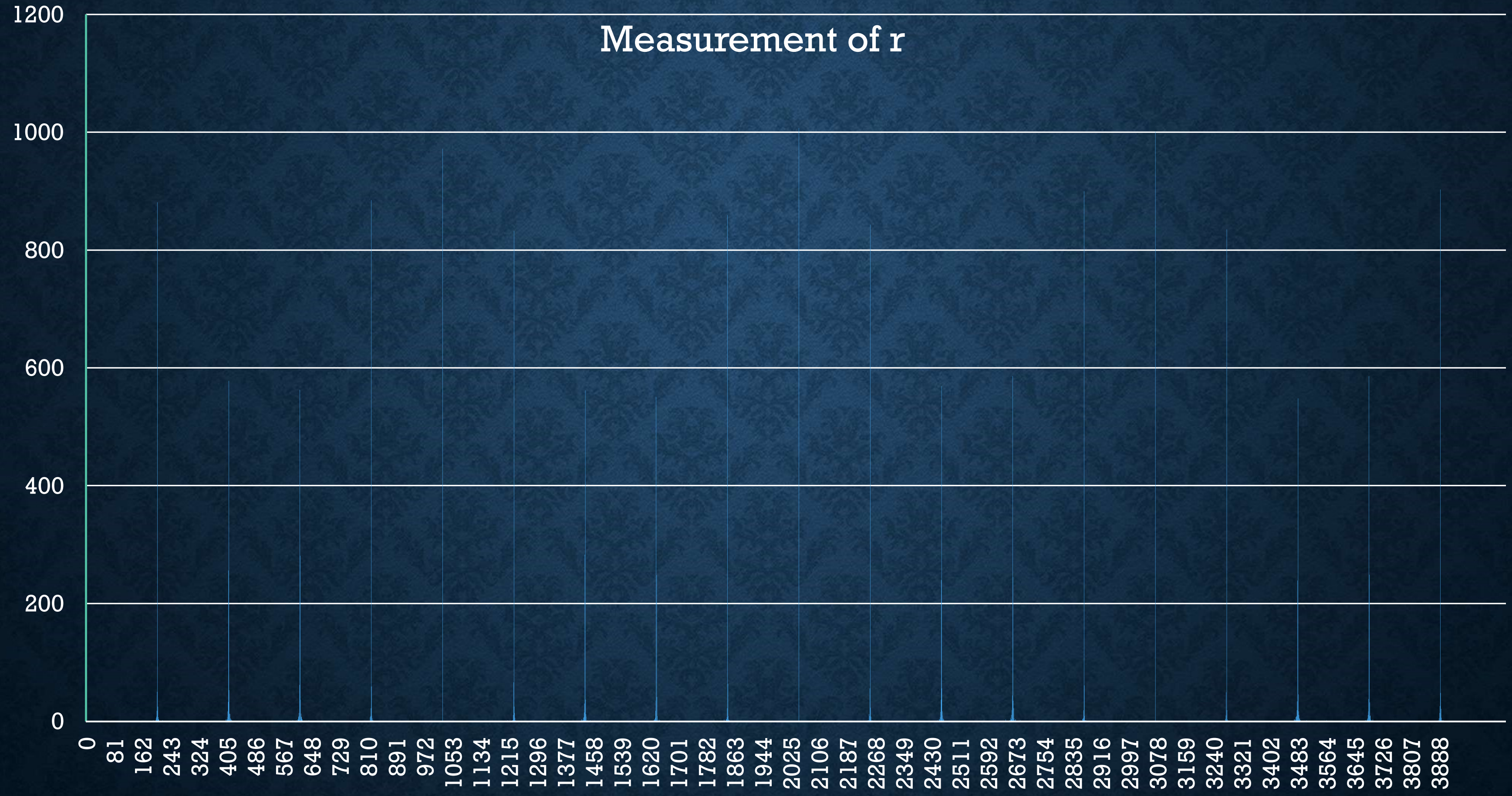
EXAMPLE $N = 55$

First step is to randomly pick a number a

Let's say $a = 17$



EXAMPLE $N = 55$ AND $a = 17$



COMPLEXITY

The circuit is classically computable in polynomial time

The runtime of the quantum circuit is $O(n^3)$ where n is the number of elementary quantum gates.

The space complexity is $O(n)$ where n is the number of bits we need to represent the number N we want to factor.

We need $4n + 2$ qubits to run shor's algorithm for an n bit number N

A more efficient solution has been implemented where the space complexity is $2n + 3$ qubits.

REFERENCES

Generic circuit implementation: https://arxiv.org/abs/quant-ph/0205095v3

Python implementation: https://github.com/astratakis/shors-algorithm