From Factoring to Phase Estimation

A discussion of Shor's algorithm

Emanuel Knill, Raymond Laflamme, Howard N. Barnum, Diego A. Dalvit, Jacek J. Dziarmaga, James E. Gubernatis, Leonid Gurvits, Gerardo Ortiz, Lorenza Viola, and Wojciech H. Zurek

The publication of Shor's quantum algorithm for efficiently factoring numbers (1994 and 1997) was the key event that stimulated many theoretical and experimental investigations of quantum computation. One of the reasons why this algorithm is so important is that the security of widely used public-key cryptographic protocols relies on the conjectured difficulty of factoring large numbers. An elementary overview of these protocols and the quantum algorithm for breaking them is provided in Artur Ekert (1998).[1] Here, we outline the relationship between factoring and the powerful technique of phase estimation. This relationship helps in understanding many of the existing quantum algorithms and was first explained in Richard Cleve et al. (1998). This explanation was motivated by Alexei Kitaev's version (1995) of the factoring algorithm.

The factoring problem requires writing a whole number N as a product of primes. (Primes are whole numbers greater than 1 that are divisible without remainder only by 1 and themselves.) Shor's algorithm solves this problem by reducing it to instances of the order-finding problem, which will be defined below. The reduction is based on basic number theory and involves efficient classical computation. At the core of Shor's algorithm is a quantum algorithm that solves the order-finding problem efficiently. In this case, an algorithm is considered efficient if it uses resources bounded by a polynomial in the number of digits of N. For more information on the requisite number theory, see any textbook on number theory (Bolker 1970, Hardy and Wright 1979).

We begin by showing that factoring reduces to order finding. The first observation is that, to factor a whole number, it is sufficient to solve the factor-finding problem, whose statement is, "Given a whole number N, find a proper factor of N if one exists. A factor of N is a whole number f that satisfies $N = fg$ for some whole number g. The factor f is proper if $f \neq 1$ and $f \neq N$. For example, if $N = 15$, then 3 and 5 are its proper factors. For some numbers, it is easy to find proper factors. For example, you can tell that N is even from the least significant digit (in decimal or binary), in which case, 2 is a proper factor (unless $N = 2$, a prime). But many numbers are not so easy. As an example, you can try to find a factor of $N = 149{,}573$ by hand.[2] You can complete the factorization of a whole number by recursively applying an algorithm for the factor-finding problem to all the proper factors found.

Before we continue the reduction of factoring to order finding, we will briefly explain modular arithmetic, which both simplifies the discussion and is necessary to avoid computing with numbers that have exponential numbers of digits. We say that a and b are equal modulo N, written as $a = b \bmod N$, if $a - b$ is divisible by N (without remainder). For example, $3 = 18 \bmod 15 = 33 \bmod 15$. Equality modulo N is well behaved with respect to addition and multiplication. That is, if $a = b \bmod N$ and $c = d \bmod N$, then $a + c = b + d \bmod N$, and $ac = bd \bmod N$. For factoring N, we will be look-

[1]All the citations in this article have been referenced on pages 31 to 33 of the main article, "Quantum Information Processing."

[2]149,573 = 373 * 401

ing for whole numbers a that are divisible by a proper factor of N. If a has this property, then so does any b with $b = a \bmod N$. We therefore perform all arithmetic modulo N. One way to think of all this is that we use only whole numbers a that satisfy $0 \le a \le N - 1$. We can implement each arithmetic operation modulo N by applying the operation in the usual way and then computing the remainder after division by N. For example, to obtain $ab \bmod N$, we first compute ab. The unique c such that $0 \le c \le N - 1$ and $c = ab \bmod N$ is the remainder after division of ab by N. Thus, c is the result of multiplying a by b modulo N. Consistent with this procedure, we can think of the expression $a \bmod N$ as referring to the remainder of a after division by N.

The second observation in the reduction of factoring to order finding is that it is sufficient to find a whole number r with the property that $r^2 - 1$ is a multiple of N, but $r - 1$ and $r + 1$ are not. Using the language of modular arithmetic, the property is expressed as $r^2 = 1 \bmod N$, but $r \ne 1 \bmod N$ and $r \ne -1 \bmod N$. Because $1 \bmod N$ and $-1 \bmod N$ are the obvious square roots of $1 \bmod N$, we say that r is a nontrivial square root of unity (modulo N). For such an r, one can write $r^2 - 1 = (r - 1)(r + 1) = mN$ for some whole number m. This implies that every prime factor p of N divides either $(r - 1)$ or $(r + 1)$ so that either $(r - 1)$ or $(r + 1)$ is or shares a factor with N. Suppose that $r - 1$ is or shares such a factor. Because $r - 1$ is not a multiple of N, the greatest common divisor of $r - 1$ and N is a factor of N. Since an efficient classical algorithm (the Euclidean algorithm) exists for finding the greatest common divisor, we can easily find the desired proper factor.

The examples of $N = 15$ and $N = 21$ serve to illustrate the key features of the algorithm. For $N = 15$, possible choices for r are $r = 4$ ($4^2 - 1 = 1 * 15$), and $r = 11$ ($11^2 - 1 = 120 = 8 * 15$). For the first choice, the proper factors emerge immediately: $4 - 1 = 3$, and $4 + 1 = 5$. For the second, it is necessary to determine the greatest common divisors (or gcd). Let $\gcd(x, y)$ stand for the greatest common divisor of x and y. The proper factors are $\gcd(11 - 1, 15) = \gcd(10, 15) = 5$, and $\gcd(11 + 1, 15) = \gcd(12, 15) = 3$. For $N = 21$, one can take $r = 8$ as $8^2 - 1 = 63 = 3 * 21$. In this case, $8 - 1 = 7$ is a proper factor, and $\gcd(8 + 1, 21) = 3$ is another.

For N even or a power of a prime, it is not always possible to find a nontrivial square root of unity. Because both cases can be handled efficiently by known classical algorithms, we can exclude them. In every other case, such numbers r exist. One way to find such an r is to start from any whole number q, with $1 < q < N$. If $\gcd(q, N) = 1$, then according to a basic result in number theory, there is a smallest whole number $k > 1$ such that $q^k - 1 = 0 \bmod N$. The number k is called the order of q modulo N. If k is even, say, $k = 2l$, then $(q^l)^2 = 1 \bmod N$, so q^l is a (possibly trivial) square root of unity. For the example of $N = 15$, we can try $q = 2$. The order of 2 modulo 15 is 4, which gives $r = 2^2 = 4$, the first of the two choices in the previous paragraph. For $N = 21$, again with $q = 2$, the order is 6: $2^6 - 1 = 63 = 3 * 21$. Thus, $r = 2^3 = 8$. We can also try $q = 11$, in which case, with foresight, it turns out that $11^6 - 1$ is divisible by 21. A possible problem appears, namely, the powers q^k, which we want to compute, are extremely large. But modular arithmetic can help us avoid this problem. For example, to find the order of 11 modulo 21 by direct search, we can perform the following computation: In general, such a direct search for the order of q modulo N is very inefficient, but as we will see,

$$\begin{array}{lllllllll}
11^2 & = & 121 & = & 5*21+16 & = & 16 \bmod 21 & & \\
11^3 & = & 11*11^2 & = & & & 11*16 \bmod 21 & = & 11*(-5) \bmod 21 \\
 & & & = & & & -55 \bmod 21 & = & -3*21+8 \bmod 21 = 8 \bmod 21 \\
11^4 & = & 11*11^3 & = & & & 11*8 \bmod 21 & = & 4*21+4 \bmod 21 = 4 \bmod 21 \\
11^5 & = & 11*11^4 & = & & & 11*4 \bmod 21 & = & 2 \bmod 21 \\
11^6 & = & 11*11^5 & = & & & 11*2 \bmod 21 & = & 1 \bmod 21
\end{array} \tag{1}$$

there is an efficient quantum algorithm that can determine the order.

A factor-finding algorithm based on the above observations is the following:

```
FACTORFIND($N$)

**Input:** A positive, nonprime whole number $N$

**Output:** A proper factor $f$ of $N$, that is, $f$ is a whole number such that $1 < f < N$ and
$N = fg$ for some whole number $g$.

1. If $N$ is even, return $f = 2$.
2. If $N = p^k$ for $p$ prime, return $p$.
3. Randomly pick $1 < q < N - 1$.
   a. If $f = \gcd(q, N) > 1$, return $f$.
4. Determine the order $k$ of $q$ modulo $N$ using the quantum order-finding algorithm.
   a. If $k$ is not even, repeat at step 3.
5. Write $k = 2l$ and determine $r = q^l \bmod N$ with $1 < r < N$.
   a. If $1 < f = \gcd(r - 1, N) < N$, return $f$.
   b. If $1 < f = \gcd(r + 1, N) < N$, return $f$.
   c. If we failed to find a proper factor, repeat at step 3.
```

The efficiency of this algorithm depends on the probability that a randomly chosen q at step 3 results in finding a factor. An analysis of the group of numbers q that satisfy $\gcd(q, N) = 1$ shows that this probability is sufficiently large.

The main problem left to be solved is finding the order of q mod N. A direct search for the order of q mod N involves computing the sequence

$$1 \rightarrow q \rightarrow q^2 \bmod N \rightarrow \ldots \rightarrow q^{k-1} \bmod N \rightarrow 1 = q^k \bmod N \ . \tag{2}$$

This sequence can be conveniently visualized as a cycle whose length is the order $q \bmod N$ (refer to Figure 1).

To introduce the quantum algorithm, we first associate the logical quantum states $|0\rangle$, $|1\rangle, \ldots |N-1\rangle$ with the numbers $0, 1, \ldots, N-1$. The map f that takes each number on the cycle to the next number along the cycle is given by $f(x) = qx \bmod N$. For q satisfying $\gcd(q, N) = 1$, the map f permutes not only the numbers on the cycle but all the numbers modulo N. As a result, the linear operator $\hat{f}$ defined by $\hat{f}|x\rangle = |f(x)\rangle = |qx \bmod N\rangle$ is unitary. The quantum algorithm deduces the length of the cycle for q by making measurements to determine the properties of the action of $\hat{f}$ on superpositions of the states $|q^s \bmod N\rangle$. To illustrate the basic ideas, we work out the example of $N = 15$ and $q = 8$. The action of $\hat{f}$ on the states $|1\rangle$, $|8\rangle$, $|4\rangle$, and $|2\rangle$ in the cycle of $8 \bmod 15$ is

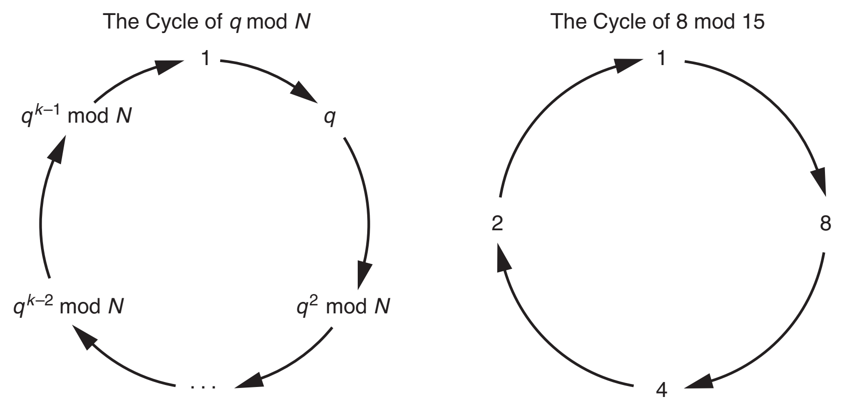


Figure 1. Multiplicative Cycles of q mod N
Each number on a cycle is obtained from the previous one by multiplication by q mod N.

completely determined by the eigenstates and eigenvalues of $\hat{f}$. For cyclicly acting permutations, a basis of eigenstates is given by the Fourier basis for the space spanned by the states in a cycle. For the cycle of interest, the Fourier basis consists of the states

$$\begin{aligned} \left|\psi_0\right\rangle &= \frac{1}{2}\left(\left|1\right\rangle + \left|8\right\rangle + \left|4\right\rangle + \left|2\right\rangle\right) , \\ \left|\psi_1\right\rangle &= \frac{1}{2}\left(\left|1\right\rangle + i\left|8\right\rangle - \left|4\right\rangle - i\left|2\right\rangle\right) , \\ \left|\psi_2\right\rangle &= \frac{1}{2}\left(\left|1\right\rangle - \left|8\right\rangle + \left|4\right\rangle - \left|2\right\rangle\right) , \text{ and} \\ \left|\psi_3\right\rangle &= \frac{1}{2}\left(\left|1\right\rangle - i\left|8\right\rangle - \left|4\right\rangle + i\left|2\right\rangle\right) . \end{aligned} \tag{3}$$

The phases of the lth state of the cycle occurring in the sum for $|\psi_m\rangle$ can be written as i^{lm}. It follows that $\hat{f}\,|\psi_m\rangle = i^m|\psi_m\rangle$, that is, the eigenvalue of $\hat{f}$ for $|\psi_m\rangle$ is i^m. Note that, in complex numbers, the powers of i are all the fourth roots of unity. In general, the Fourier basis for the cycle $\ldots \rightarrow |q^l \text{ mod } \mathrm{N}\rangle \rightarrow \ldots$ consists of the states $|\psi_m\rangle = \Sigma_l \omega^{lm}|q^l \text{ mod } N\rangle$, where $\omega = e^{i2\pi/k}$ is a primitive kth root of unity. (The complex number x is a primitive kth root of unity if k is the smallest whole number $k > 0$ such that $x^k = 1$. For example, both –1 and i are fourth roots of unity, but only i is primitive.)

It is, of course, possible to express the logical state $|1\rangle$ using the Fourier basis

$$\left|1\right\rangle = \frac{1}{2}\left(\left|\psi_0\right\rangle + \left|\psi_1\right\rangle + \left|\psi_2\right\rangle + \left|\psi_3\right\rangle\right) . \tag{4}$$

The key step of the quantum algorithm for order finding consists of a measurement to estimate a random eigenvalue of $\hat{f}$, whose associated eigenstate occurs in the expression for $|1\rangle$ in terms of the Fourier basis. If the eigenvalue found is a kth root of unity, we infer that the cycle length is divisible by k and check (using a classical algorithm) whether this is the order of q. In the example, the random eigenvalues are 1 (the only primitive first root of unity), i and $-i$ (primitive fourth roots of unity), and –1 (the primitive second root of unity). The order is found if the random eigenvalue is a fourth root of unity, which happens with probability 1/2 in this case.

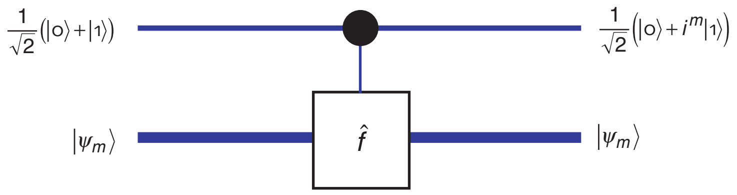

$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + i^m|1\rangle)$
$|\psi_m\rangle$
$\hat{f}$
$|\psi_m\rangle$

Figure 2. Phase Estimation with One Qubit
The input is a product state on one ancilla qubit and on a second quantum system, as shown. The state $|\psi_m\rangle$ on the second system is an eigenstate of $\hat{f}$. For the example provided in Equation (3), the eigenvalue is i^m. A controlled-$\hat{f}$ operation is applied to the input, that is, $\hat{f}$ is applied to the second system conditional on $|1\rangle$ for the ancilla qubit. In the bra-ket notation, the total operation can be written as $|0\rangle\langle 0| + |1\rangle\langle 1|\hat{f}$ (system labels have been omitted). Because $\hat{f}$ changes only the phase of its input, the second system is unchanged, but the phase modifies the ancilla qubit's superposition as shown.

The quantum algorithm for obtaining an eigenvalue is called the phase estimation algorithm, and it exploits a more general version of the phase kickback we encountered in the solution of the parity problem. The phase kickback transfers the eigenvalue of an eigenstate of $\hat{f}$ to a Fourier basis on a number of additional qubits called helper or ancilla qubits. Which Fourier state results is then determined by a subroutine called the measured quantum Fourier transform. We introduce these elements in the next paragraphs. Their combination for solving the general order-finding problem is illustrated on page 45.

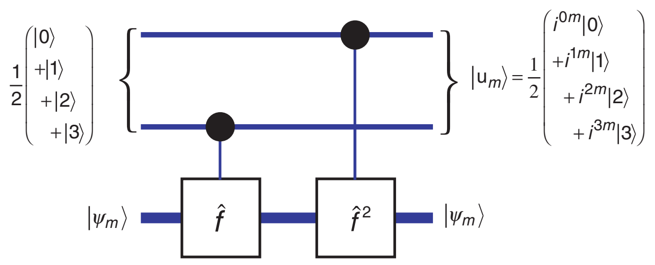
Figure 2 shows how to kick back the eigenvalue of an eigenstate of $\hat{f}$ using a network implementing the controlled-$\hat{f}$ operation. The network in Figure 2 can be used with input $|1\rangle$ on the second system. From Equation (4) and the superposition principle, it follows that the output correlates the different phase kickback states with the four eigenvectors $|\psi_m\rangle$. That is, the network implements the following transformation:

$$\frac{1}{2\sqrt{2}}\left(|0\rangle+|1\rangle\right)\begin{pmatrix} |\psi_0\rangle \\ +|\psi_1\rangle \\ +|\psi_2\rangle \\ +|\psi_3\rangle \end{pmatrix} \rightarrow \frac{1}{2\sqrt{2}}\begin{pmatrix} \left(|0\rangle + i^0|1\rangle\right)|\psi_0\rangle \\ +\left(|0\rangle + i^1|1\rangle\right)|\psi_1\rangle \\ +\left(|0\rangle + i^2|1\rangle\right)|\psi_2\rangle \\ +\left(|0\rangle + i^3|1\rangle\right)|\psi_3\rangle \end{pmatrix}. \tag{5}$$

The hope is that a measurement of the first qubit can distinguish between the four possible phases that can be kicked back. However, because the four states are not mutually orthogonal, they are not unambiguously distinguishable by a measurement. To solve this problem, we use a second qubit and a controlled-$\hat{f}^2$ as shown in Figure 3.

The four possible states $|u_m\rangle$ that appear on the ancilla qubits in the network of Figure 3 are the Fourier basis for the cycle $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0$ and are therefore orthonormal. If we apply the network of Figure 3 with $|1\rangle$ instead of $|\psi_m\rangle$ at the lower input, the output correlates the four $|\psi_m\rangle$ in the superposition with the $|u_m\rangle$, which makes the information about the eigenvalues of $\hat{f}$ available in the Fourier basis of the two ancilla qubits. This approach has the advantage that the states are known, whereas in the Fourier basis for the cycle of q mod N, the states depend on the numbers in the cycle, which are not known in advance (except in very simple cases, such as the example we are working with).

To learn one of the eigenvalues of $\hat{f}$, the last step is to make a measurement in the Fourier basis. For one qubit representing the binary numbers 0 and 1, the Fourier basis is $1/\sqrt{2}(|0\rangle + |1\rangle)$ and $1/\sqrt{2}(|0\rangle - |1\rangle)$, which is constructed as discussed after



$\frac{1}{2}\begin{pmatrix}|0\rangle\\+|1\rangle\\+|2\rangle\\+|3\rangle\end{pmatrix}$
$|u_m\rangle=\frac{1}{2}\begin{pmatrix}i^{0m}|0\rangle\\+i^{1m}|1\rangle\\+i^{2m}|2\rangle\\+i^{3m}|3\rangle\end{pmatrix}$
$|\psi_m\rangle$
$\hat{f}$
$\hat{f}^2$
$|\psi_m\rangle$

Figure 3. Phase Estimation with Two Qubits
Using two qubits ensures distinguishability of the eigenvalues of $\hat{f}$ for the states $|\psi_m\rangle$. The states of the input qubits are used to represent the numbers from 0 to 3 in binary. The most significant bit (the two's digit in binary representation) is carried by the top qubit. That is, we make the following identification: $|0\rangle = |00\rangle$, $|1\rangle = |01\rangle$, $|2\rangle = |10\rangle$, and $|3\rangle = |11\rangle$. It follows that the network has the effect of applying $\hat{f}^m$ conditional on the input qubits' logical state being $|m\rangle$.

Equation (3) but using the square root of unity $\omega = -1$ instead of the fourth root i. To make a measurement that determines which of the two basis vectors is present, it suffices to apply the Hadamard transform **H** and make a standard measurement, just as we did twice in the network of Figure 2 in the article "Quantum Information Processing" on page 23. A more complicated network works with two qubits representing the binary numbers from 0 to 3. Such a network is shown in Figure 4.

To see how the network extracts the bits in the index of $|u_a\rangle$, we can follow the states as the network is executed. The input state at checkpoint 1 in Figure 4 is given by

$$|\phi_1\rangle = |u_a\rangle = \frac{1}{2}\begin{pmatrix} i^{0*a}|0\rangle \\ +i^{1*a}|1\rangle \\ +i^{2*a}|2\rangle \\ +i^{3*a}|3\rangle \end{pmatrix} = \frac{1}{2}\begin{pmatrix} i^{(0*2^1+0*2^0)(a_1*2^1+a_0*2^0)}|00\rangle \\ +i^{(0*2^1+1*2^0)(a_1*2^1+a_0*2^0)}|01\rangle \\ +i^{(1*2^1+0*2^0)(a_1*2^1+a_0*2^0)}|10\rangle \\ +i^{(1*2^1+1*2^0)(a_1*2^1+a_0*2^0)}|11\rangle \end{pmatrix}. \tag{6}$$

In the last sum, the relevant numbers have been fully expanded in terms of their binary digits to give a flavor of the general principles underlying the measured Fourier transform. The next step of the network applies a Hadamard gate to the qubit carrying the most significant digit. To understand how it succeeds in extracting a_0, the least signifi-

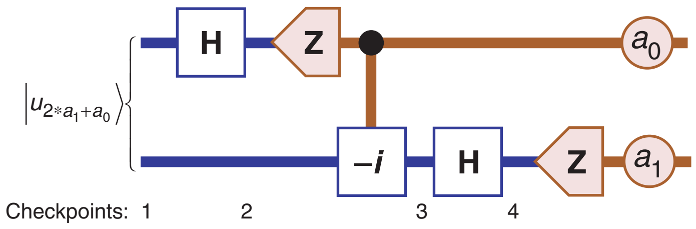


Figure 4. Measured Quantum Fourier Transform on Two Qubits
The two qubits represent the numbers 0, 1, 2, and 3 . If the input is one of the Fourier states $|u_a\rangle$, where the binary digits of a are determined by $a = 2 * a_1 + a_0$, then the measurement outcomes are a_0 and a_1, as shown. The numbers under the network are checkpoints used for analysis. [For details on the measured Fourier transform, see Griffiths and Niu (1996).]

cant bit of a, let b with binary digits b_0 and b_1 represent one of the logical states of the two qubits. As before, the most significant bit b_1 is represented by the top/first qubit that the first Hadamard gate is applied to. The phase of $|b\rangle$ in Equation (6) is given by $i^{(b_1 * 2^1 + b_0 * 2^0)(a_1 * 2^1 + a_0 * 2^0)}$. Next, we determine how the phase depends on b_1:

$$\begin{aligned} i^{\left(b_1 * 2^1 + b_0 * 2^0\right)\left(a_1 * 2^1 + a_0 * 2^0\right)} &= i^{b_1 * 2^1 * \left(a_1 * 2^1 + a_0 * 2^0\right)}\; i^{b_0 * 2^0 * \left(a_1 * 2^1 + a_0 * 2^0\right)} \\ &= i^{b_1 * a_1 * 2^2}\; i^{b_1 * a_0 * 2^1}\; i^{b_0 * 2^0 * \left(a_1 * 2^1 + a_0 * 2^0\right)} \\ &= \left(i^4\right)^{b_1 * a_1} \left(i^2\right)^{b_1 * a_0}\; i^{b_0 * 2^0 * \left(a_1 * 2^1 + a_0 * 2^0\right)} \\ &= (-1)^{b_1 * a_0}\; i^{b_0 * 2^0 * \left(a_1 * 2^1 + a_0 * 2^0\right)} \quad . \end{aligned} \tag{7}$$

It follows that, if $a_0 = 0$, the phase does not depend on b_1, and if $a_0 = 1$, it changes sign with b_1. This sign change can be detected by performing the Hadamard transform and measuring, as can be seen explicitly by computing the state after the Hadamard transform at checkpoint 2:

$$\begin{aligned} |\phi_2\rangle &= \frac{1}{\sqrt{2}}\left(i^{0 * 2^0 * \left(a_1 * 2^1 + a_0 * 2^0\right)} |a_0\rangle|0\rangle + i^{1 * 2^0 * \left(a_1 * 2^1 + a_0 * 2^0\right)} |a_0\rangle|1\rangle \right) \\ &= |a_0\rangle \frac{1}{\sqrt{2}}\left(i^{0 * 2^0 * \left(a_1 * 2^1 + a_0 * 2^0\right)} |0\rangle + i^{1 * 2^0 * \left(a_1 * 2^1 + a_0 * 2^0\right)} |1\rangle \right) . \end{aligned} \tag{8}$$

The phases still show a dependence on a_0 via the terms $i^{b_0 * 2^0 * a_0 * 2^0} = i^{b_0 a_0}$. The purpose of the phase-shift gate conditioned on the measurement outcome is to remove that dependence. The result is the following state on the remaining qubit at checkpoint 3:

$$\begin{aligned} |\phi_3\rangle &= \frac{1}{\sqrt{2}}\left(i^{0 * 2^0 * a_1 * 2^1} |0\rangle + i^{1 * 2^0 * a_1 * 2^1} |1\rangle \right) \\ &= \frac{1}{\sqrt{2}}\left((-1)^{0 * a_1} |0\rangle + (-1)^{1 * a_1} |1\rangle \right) \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{a_1} |1\rangle \right) . \end{aligned} \tag{9}$$

The final Hadamard transform followed by a measurement therefore results in the bit a_1, as desired.

The elements that we used to determine the order of 8 modulo 15 can be combined and generalized to determine the order of any q modulo N with gcd(q, N) = 1. The general network is shown in Figure 5. Two features of the generalization are not apparent from the example. First, in order for the quantum network to be efficient, an efficient implementation of the controlled $\hat{f}^{2^l}$ operation is required. To obtain such an implementation, first note that to calculate $f^{2^l}(x) = q^{2^l} x \bmod N$, it suffices to square q repeatedly

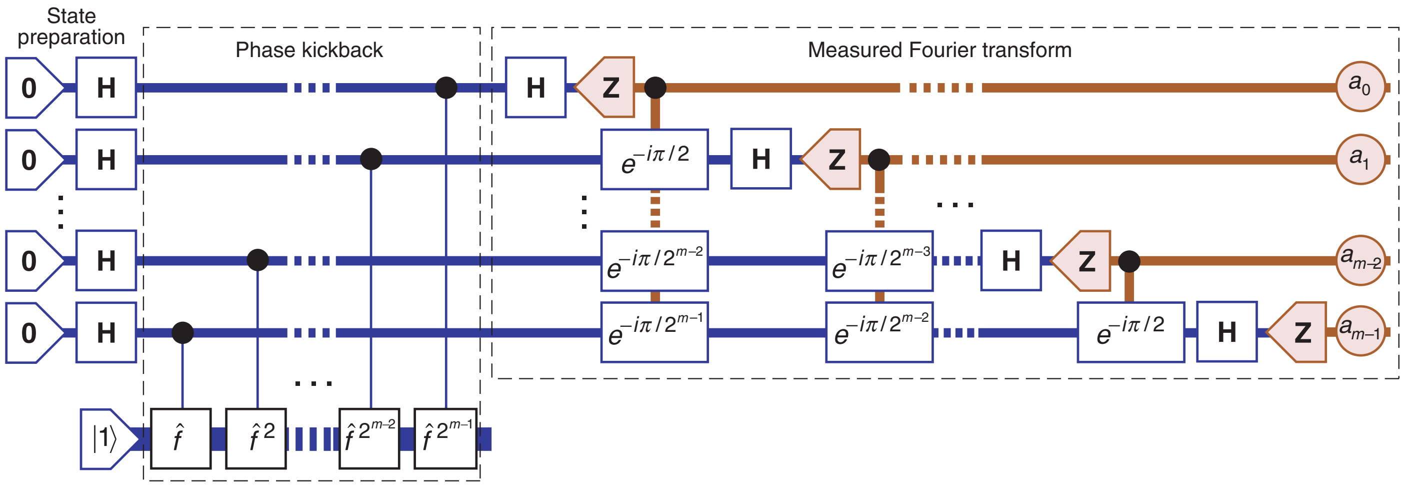


Figure 5. Network for Quantum Order Finding and Phase Estimation

The number m of qubits used for the phase kickback has to be chosen such that $m > 2 * \log_2(k_u)$, where k_u is a known upper bound on the order k of q mod N. Because $N > k$, one can set $m = 2\lceil \log_2(N) \rceil$, where $\lceil x \rceil$ is the least whole number $s \geq x$. There is an eigenvalue $\lambda_l = e^{i2l\pi/k}$ of one of the Fourier eigenvectors associated with the cycle of q mod N such that the number a, whose binary digits are the measurement outcomes, satisfies $e^{i\pi a/2^{m-1}} \approx e^{i2\pi l/k}$. More precisely, with probability above .405, there exists l such that $|a/2^m - l/k| \leq 1/2^{m+1}$ (Cleve et al. 1998). Because any two distinct rational numbers with denominator at most k_u differ by at least $1/k_u^2 > 2/2^{m+1}$, the theory of rational approximations guarantees that we can uniquely determine the number l/k. There is an efficient classical algorithm based on continued fractions that computes r and s with $r/s = l/k$ and $s = k/gcd(l, k)$. The probability that gcd(l, k) = 1 is at least 1/ ($\log_2(k)$ + 1), in which case we learn that $s = k$ and this is the order of q mod N. Note that the complexity of the network depends on the complexity of implementing the controlled $\hat{f}^{2^l}$ operations. Because these operations can be implemented efficiently, the network and hence the determination of the order of q mod N are efficient in the sense that, on average, polynomial resources in $\log_2(N)$ suffice.

modulo N using $(q^{2^m})^2$ mod N = $q^{2^{m+1}}$ mod N until we obtain q^{2^l} mod N. The result is then multiplied by x mod N. This computation is efficient. For any given q, the computation can be converted to an efficient network consisting of Toffoli gates and controlled-not gates acting on the binary representation of x. The conversion can be accomplished with standard techniques from the theory of reversible classical computation. The result is an efficient network for $\hat{f}^{2^l}$. Basic network theory can then be used to implement the controlled version of this operation (Barenco et al. 1995).

To understand the second feature, note that we were lucky to anticipate that the order of 8 modulo 15 was a power of 2, which nicely matched the measured Fourier transform we constructed on two qubits. The measured Fourier transform on m ancilla qubits can detect exactly only eigenvalues that are powers of the 2^mth root of unity $e^{i\pi/2^{m-1}}$. The phase kicked back by the controlled operations corresponds to a kth root of unity. Given a Fourier state on the cycle of q mod N, the resulting state on the ancilla qubits has phases that go as powers of a kth root of unity. Fortunately, the ancilla's Fourier basis is such that the measured Fourier transform picks up primarily those basis states whose generating phase is close to the kickback phase. Thus, we are likely to detect a nearby $\omega = e^{il\pi/2^{m-1}}$. It is still necessary to infer (a divisor of) k from knowledge of such an ω. Because we know that the order k is bounded by N, the number of possible phases kicked back that are near the measured ω is limited. To ensure that there is only one possible such phase, it is necessary to choose m such that $2^m > N^2$. (See also Figure 5.) ■