

Fast parallel circuits for the quantum Fourier transform

Richard Cleve* John Watrous†
University of Calgary‡

June 1, 2000

Abstract

We give new bounds on the circuit complexity of the quantum Fourier transform (QFT). We give an upper bound of $O(\log n + \log \log(1/\varepsilon))$ on the circuit depth for computing an approximation of the QFT with respect to the modulus 2^n with error bounded by ε . Thus, even for exponentially small error, our circuits have depth $O(\log n)$. The best previous depth bound was $O(n)$, even for approximations with constant error. Moreover, our circuits have size $O(n \log(n/\varepsilon))$. We also give an upper bound of $O(n(\log n)^2 \log \log n)$ on the circuit size of the exact QFT modulo 2^n , for which the best previous bound was $O(n^2)$.

As an application of the above depth bound, we show that Shor's factoring algorithm may be based on quantum circuits with depth only $O(\log n)$ and polynomial-size, in combination with classical polynomial-time pre- and post-processing. In the language of computational complexity, this implies that factoring is in the complexity class ZPP^{BQNC} , where BQNC is the class of problems computable with bounded-error probability by quantum circuits with poly-logarithmic depth and polynomial size.

Finally, we prove an $\Omega(\log n)$ lower bound on the depth complexity of approximations of the QFT with constant error. This implies that the above upper bound is asymptotically optimal (for a reasonable range of values of ε).

1 Introduction and summary of results

In this paper we consider the quantum circuit complexity of the *quantum Fourier transform (QFT)*. The quantum Fourier transform is the key quantum operation at the heart of Shor's quantum algorithms for factoring and computing discrete logarithms [35] and the known extensions and variants of these algorithms (see, e.g., Kitaev [25], Boneh and Lipton [7], Grigoriev [20], and Cleve, Ekert, Macchiavello, and Mosca [10]). The quantum Fourier transform also plays a key role in extensions of Grover's quantum searching technique [21], due to Brassard, Høyer, and Tapp [8] and Mosca [29].

In order to discuss the quantum Fourier transform in greater detail we recall the *discrete Fourier transform (DFT)*; for a given dimension m the discrete Fourier transform is a linear operator on

*Email: cleve@cpsc.ucalgary.ca

†Email: jwatrous@cpsc.ucalgary.ca

‡Department of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4. Research partially supported by Canada's NSERC.

\mathbb{C}^m mapping $(a_0, a_1, \dots, a_{m-1})$ to $(b_0, b_1, \dots, b_{m-1})$, where

$$b_x = \sum_{y=0}^{m-1} (e^{2\pi i/m})^{x \cdot y} a_y. \quad (1)$$

The discrete Fourier transform has many important applications in classical computing, essentially due to the efficiency of the *fast Fourier transform (FFT)*, which is an algorithm that computes the DFT with $O(m \log m)$ arithmetic operations, as opposed to the obvious $O(m^2)$ method. The FFT algorithm was proposed by Cooley and Tukey in 1965 [13], though its origins can be traced back to Gauss in 1866 [17]. The FFT plays an important role in digital signal processing, and it has been suggested [16] as a contender for the second most important nontrivial algorithm in practice, after fast sorting. The DFT (and the FFT algorithm) generalize to certain algebraic structures, such as rings containing primitive m^{th} roots of unity (which can play the role of $e^{2\pi i/m}$ in Eq. 1). This more abstract type of FFT is a principal component in Schönhage and Strassen’s fast multiplication algorithm [33], which can be expressed as circuits of size $O(n \log n \log \log n)$ for multiplying n -bit integers. For more applications—of which there are many—and historical information, see [27, 12, 23].

The *quantum Fourier transform (QFT)* is a unitary operation that essentially performs the DFT on the amplitude vector of a quantum state—the QFT maps the quantum state $\sum_{x=0}^{m-1} \alpha_x |x\rangle$ to the state $\sum_{x=0}^{m-1} \beta_x |x\rangle$, where

$$\beta_x = \frac{1}{\sqrt{m}} \sum_{y=0}^{m-1} (e^{2\pi i/m})^{x \cdot y} \alpha_y. \quad (2)$$

For certain values of m there are very efficient quantum algorithms for the QFT. The fact that the quantum circuit size can be polynomial in $\log m$ for some values of m was first observed by Shor [34] and is of critical importance in his polynomial-time algorithms for prime factorization and discrete logarithms. Shor’s original method may be described as a “mixed-radix” method, and is discussed further in Section 7.2. In the particular case where $m = 2^n$, there exist quantum circuits performing the quantum Fourier transform with $O(n^2)$ gates, which was proved by Coppersmith [14] (see also [9]). These circuits are based on a recursive description of the QFT that is analogous to the description of the DFT exploited by the FFT. While in some sense these quantum circuits are exponentially faster than the classical FFT, the task that they perform is quite different. The QFT does not explicitly produce any of the values $\beta_0, \beta_1, \dots, \beta_{m-1}$ as output (nor does it explicitly obtain any of the values $\alpha_0, \alpha_1, \dots, \alpha_{m-1}$ as input). Intuitively, the difference between performing a DFT and a QFT can be thought of as being analogous to the difference between computing all the probabilities that comprise a probability distribution and sampling a probability distribution—the latter task being frequently much easier.

Coppersmith [14] also proposed quantum circuits that approximate the QFT with error bounded by ε , and showed that such approximations can be computed by circuits of size $O(n \log(n/\varepsilon))$ for modulus 2^n . Such approximations can be thought of as unitary operations whose distance from the QFT (in the operator norm induced by Euclidean distance) is bounded by ε . Kitaev [25] showed how the QFT for an arbitrary modulus m can be approximated by circuits with size polynomial in $\log(m/\varepsilon)$. For most information processing purposes, it suffices to use such approximations of quantum operations (for ε ranging from constant down to $1/n^{O(1)}$). Indeed, since it seems rather

implausible to physically implement quantum gates with perfect accuracy, the need to ultimately consider approximations is likely inevitable. Thus, we believe that the most relevant consideration is to approximately compute the QFT, though exact computations of the QFT are still of interest as part of the mathematical theory of quantum computation.

Moore and Nilsson [28] showed how to obtain logarithmic-depth circuits that perform encoding and decoding for standard quantum error-correcting codes. For the QFT, in both the exact and approximate case, the gates in Coppersmith’s circuits can be arranged so as to have depth $2n - 1$, as noted in [28], but not less depth than this. Similarly, the techniques of Shor and of Kitaev have polynomial depth. Our first result shows that it is possible to compute good approximations of the QFT with logarithmic-depth quantum circuits.

Theorem 1 *For any n and ε there is a quantum circuit approximating the QFT modulo 2^n with precision ε that has size $O(n \log(n/\varepsilon))$ and depth $O(\log n + \log \log(1/\varepsilon))$.*

By an approximation of a unitary operation U with *precision* ε , we mean a unitary operation V (possibly acting on additional ancilla qubits) with the following property. For any input (pure) quantum state, the Euclidean distance between applying U to the state and V to the state is at most ε (in the Hilbert space that includes the input/output qubits and the ancilla qubits). Also, whenever we refer to *circuits*, unless otherwise stated, there is an implicit assumption that the circuits belong to a logarithmic-space uniformly generated family in the usual way (via a *classical* Turing machine). In Section 7.2, we consider a different approach for parallelizing Shor’s QFT method, which gives somewhat worse bounds.

The proof of Theorem 1 follows the general approach introduced by Kitaev [25], with several efficiency improvements as well as parallelizations. In particular, we introduce a new parallel method for performing multiprecision phase estimation.

We also show that, if size rather than depth is the primary consideration, it is possible to compute the QFT *exactly* with a near-linear number of gates.

Theorem 2 *For any n there is a quantum circuit that exactly computes the QFT modulo 2^n that has size $O(n(\log n)^2 \log \log n)$ and depth $O(n)$.*

Theorem 2 is based on a nonstandard recursive description of the QFT [9] combined with an asymptotically fast multiplication algorithm [33].

There are several reasons why we believe results regarding quantum circuit complexity, such as in the above theorems, are important. First, circuit depth is likely to be particularly relevant in the quantum setting for physical reasons. Perhaps most notably, fault tolerant quantum computation necessarily requires parallelization anyway [1]—under various noise models, error correction must continually be applied in parallel to the qubits of a quantum computer, even when the qubits are doing nothing. In such models, parallelization saves not only the total amount of time, but also the total amount of work. Furthermore, informally speaking, the depth of a quantum circuit corresponds to the amount of time coherence must be preserved, so in addition to saving work, parallelization may allow for larger quantum circuits to be implemented within systems having shorter decoherence times or using less extensive error correction. A final reason is that such results suggest alternate methods for performing various operations, which may in turn suggest or shed light on quantum algorithms for other problems or more general methods for improving efficiency of quantum algorithms.

It has long been known that the main bottleneck of the quantum portion of Shor’s factoring algorithm is not the QFT, but rather is the modular exponentiation step. If it were possible to perform modular exponentiation by classical circuits with poly-logarithmic depth and polynomial size then it would be possible to implement Shor’s factoring algorithm in poly-logarithmic depth with a polynomial number of qubits. Although no such algorithm is known for modular exponentiation, we can prove the following weaker result, which nevertheless implies that quantum computers need only run for poly-logarithmic time for factoring to be feasible.

Theorem 3 *There is an algorithm for factoring n -bit integers that consists of: a classical pre-processing stage, computed by a polynomial-size classical circuit; followed by a quantum information processing stage, computed by an $O(\log n)$ -depth $O(n^5(\log n)^2)$ -size quantum circuit¹; followed by a classical post-processing stage, computed by a polynomial-size classical circuit. Furthermore, the size of the quantum circuit can be reduced if a larger depth is allowed. In particular, the size can be reduced to $O(n^3)$ if the depth is increased to $O((\log n)^2)$.*

If we define the complexity class BQNC as all computational problems that can be solved by quantum circuits with poly-logarithmic depth and polynomial size—a reasonably natural extension of previous notation (see, e.g., [11, 28])—then Theorem 3 implies that the factoring problem is in ZPP^{BQNC} .

Finally, we consider the minimum depth required for approximating the QFT. It is fairly easy to show that computing the QFT *exactly* requires depth at least $\log n$. However, this is less clear in the case of approximations—and we exhibit a problem related to the QFT whose depth complexity decreases from $\log n$ in the exact case to $O(\log \log n)$ for approximations with precision ε , whenever $\varepsilon \in 1/n^{O(1)}$. Nevertheless, we show the following.

Theorem 4 *Any quantum circuit consisting of one- and two-qubit gates that approximates the QFT with precision $\frac{1}{10}$ or smaller must have depth at least $\log n$.*

This implies that the depth upper bound in Theorem 1 is asymptotically optimal for a reasonable range of values of ε .

The remainder of this paper is organized as follows. In Section 2, we review some definitions and introduce notation that is used in subsequent sections. In Section 3 we prove the depth and size bounds for quantum circuits approximating the quantum Fourier transform for any power-of-2 modulus as claimed in Theorem 1, and in Section 4 we prove the size bound claimed in Theorem 2 for exactly computing the quantum Fourier transform. In Section 5 we prove Theorem 3 by demonstrating how Shor’s factoring algorithm can be arranged so as to require only logarithmic-depth quantum circuits. In Section 6 we prove the lower bound for the QFT in Theorem 4. In Section 7 we discuss the situation when the modulus for the quantum Fourier transform is not necessarily a power of 2, including arbitrary moduli and the special case of “smooth” moduli considered in Shor’s original method for performing quantum Fourier transform. We conclude with Section 8, which mentions some directions for future work relating to this paper.

¹In this case, the underlying circuit family is *polynomial-time* uniform rather than logarithmic-space uniform.

2 Definitions and notation

Notation for special quantum states: For an n -bit modulus m , we will identify each $x \in \mathbb{Z}_m$ with its binary representation $x_{n-1} \dots x_1 x_0 \in \{0, 1\}^n$. For $x \in \mathbb{Z}_m$, the state $|x\rangle = |x_{n-1} \dots x_1 x_0\rangle$ is called a *computational basis state*.

For $x \in \mathbb{Z}_m$, the state

$$|\psi_x\rangle = \frac{1}{\sqrt{m}} \sum_{y=0}^{m-1} (e^{2\pi i/m})^{x \cdot y} |y\rangle, \quad (3)$$

is a *Fourier basis state* with *phase parameter* x . As noted in [10], when $m = 2^n$, $|\psi_x\rangle$ can be factored as follows

$$|\psi_{x_{n-1} \dots x_1 x_0}\rangle = \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2\pi i(0 \cdot x_0)} |1\rangle) (|0\rangle + e^{2\pi i(0 \cdot x_1 x_0)} |1\rangle) \dots (|0\rangle + e^{2\pi i(0 \cdot x_{n-1} \dots x_1 x_0)} |1\rangle). \quad (4)$$

For convenience, we define the state

$$|\mu_\theta\rangle = \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i\theta} |1\rangle), \quad (5)$$

where θ is a real parameter. Using this notation, we can rewrite Eq. 4 as

$$|\psi_{x_{n-1} \dots x_1 x_0}\rangle = |\mu_{0 \cdot x_0}\rangle |\mu_{0 \cdot x_1 x_0}\rangle \dots |\mu_{0 \cdot x_{n-1} \dots x_1 x_0}\rangle. \quad (6)$$

Definition of the QFT: The *quantum Fourier transform (QFT)* is the unitary operation that maps $|x\rangle$ to $|\psi_x\rangle$ (for all $x \in \mathbb{Z}_m$).

Mappings related to the QFT: A *quantum Fourier state computation (QFS)* is any unitary operation that maps $|x\rangle|0\rangle$ to $|x\rangle|\psi_x\rangle$ (for all $x \in \mathbb{Z}_m$). When the input is a computational basis state, this computes the corresponding Fourier state, but without erasing the input. We refer to approximations of a QFS as *Fourier state estimation*. A *quantum Fourier phase computation (QFP)* is any unitary operation that maps $|\psi_x\rangle|0\rangle$ to $|\psi_x\rangle|x\rangle$ (for all $x \in \mathbb{Z}_m$). When the input is a Fourier basis state, this computes the corresponding phase parameter, but without erasing the input. We refer to approximations of a QFP as *Fourier phase estimation*. As pointed out by Kitaev [25], the QFT can be computed by composing a QFS and the inverse of a QFP: $|x\rangle|0\rangle \mapsto |x\rangle|\psi_x\rangle \mapsto |0\rangle|\psi_x\rangle$.

Quantum gates: All of the quantum circuits that we construct will be composed of three types of unitary gates. One is the one-qubit *Hadamard* gate, H , which maps $|x\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^x |1\rangle)$ (for $x \in \{0, 1\}$). Another is the one-qubit *phase shift* gate, $P(\theta)$, where θ is a parameter of the form $x/2^n$ (for $x \in \mathbb{Z}_{2^n}$). $P(\theta)$ maps $|x\rangle$ to $e^{2\pi i\theta x} |x\rangle$ (for $x \in \{0, 1\}$). Finally, we use two-qubit *controlled-phase* shift gates, controlled- $P(\theta)$ (c- $P(\theta)$ for short), which map $|x\rangle|y\rangle$ to $e^{2\pi i\theta xy} |x\rangle|y\rangle$ (for $x, y \in \{0, 1\}$). Note that this set is universal, and in particular that any (classical) reversible circuit can be composed of these gates.

3 New depth bounds for the QFT

The main purpose of this section is to prove Theorem 1.

First, we review the approach of Kitaev [25] for performing the QFT for an arbitrary modulus m . By linearity, it is sufficient to give a circuit that operates correctly on computational basis states.

Given a computational basis state $|x\rangle$, first create the Fourier basis state with phase parameter x (which can be done easily if $|x\rangle$ is not erased in the process). The system is now in the state $|x\rangle|\psi_x\rangle$. Now, by performing Fourier phase estimation, the state $|x\rangle|\psi_x\rangle$ can be approximated from the state $|0\rangle|\psi_x\rangle$. Therefore, by performing the inverse of Fourier phase estimation on the state $|x\rangle|\psi_x\rangle$, a good estimate of the state $|0\rangle|\psi_x\rangle$ is obtained.

The particular phase estimation procedure used by Kitaev does not readily parallelize, but, in the case where the modulus is a power of 2, we give a new phase estimation procedure that does parallelize. This procedure requires several copies of the Fourier basis state rather than just one. To insure that the entire process parallelizes, we must parallelize the creation of the Fourier basis state as well as the process of copying and uncopying this state.

The basic steps of our technique are as follows:

1. Creation of the Fourier basis state, which is the mapping

$$|x\rangle|0\rangle \mapsto |x\rangle|\psi_x\rangle.$$

2. Copying the Fourier basis state, which is the mapping

$$|\psi_x\rangle|0\rangle \cdots |0\rangle \mapsto |\psi_x\rangle|\psi_x\rangle \cdots |\psi_x\rangle.$$

3. Erasing the computational basis state by means of estimating the phase of the Fourier basis state, which is the mapping

$$|x\rangle|\psi_x\rangle|\psi_x\rangle \cdots |\psi_x\rangle \mapsto |0\rangle|\psi_x\rangle|\psi_x\rangle \cdots |\psi_x\rangle.$$

4. Reverse step 2, which is the mapping

$$|\psi_x\rangle|\psi_x\rangle \cdots |\psi_x\rangle \mapsto |\psi_x\rangle|0\rangle \cdots |0\rangle.$$

Each of these components is discussed in detail in the subsections that follow. Throughout we assume the modulus is $m = 2^n$.

3.1 Parallel Fourier state computation and estimation

The first step is the creation of the Fourier basis state corresponding to a given computational basis state $|x\rangle$. This corresponds to the mapping

$$|x\rangle|0\rangle \mapsto |x\rangle|\psi_x\rangle. \tag{7}$$

First let us consider a circuit that performs this transformation exactly. By Eq. 6 (equivalently, Eq. 4), it suffices to compute the states $|\mu_{0.x_0}\rangle, |\mu_{0.x_1x_0}\rangle, \dots, |\mu_{0.x_{n-1}\dots x_1x_0}\rangle$ individually.

The circuit suggested by Figure 1 performs the required transformation for $|\mu_{0.x_j\dots x_0}\rangle$. In this figure we have not labelled the controlled phase shift gates, $c-P(\theta)$ (such gates are defined in Section 2), which are the gates in the center drawn as two solid circles connected by a line. In the above case, the phase θ depends on j and on the particular qubit of $|x_{n-1}\dots x_1x_0\rangle$ on which the gate acts. The value of θ for the controlled phase shift acting on $|x_i\rangle$ is 2^{i-j-1} (for $i \in \{0, 1, \dots, j\}$).

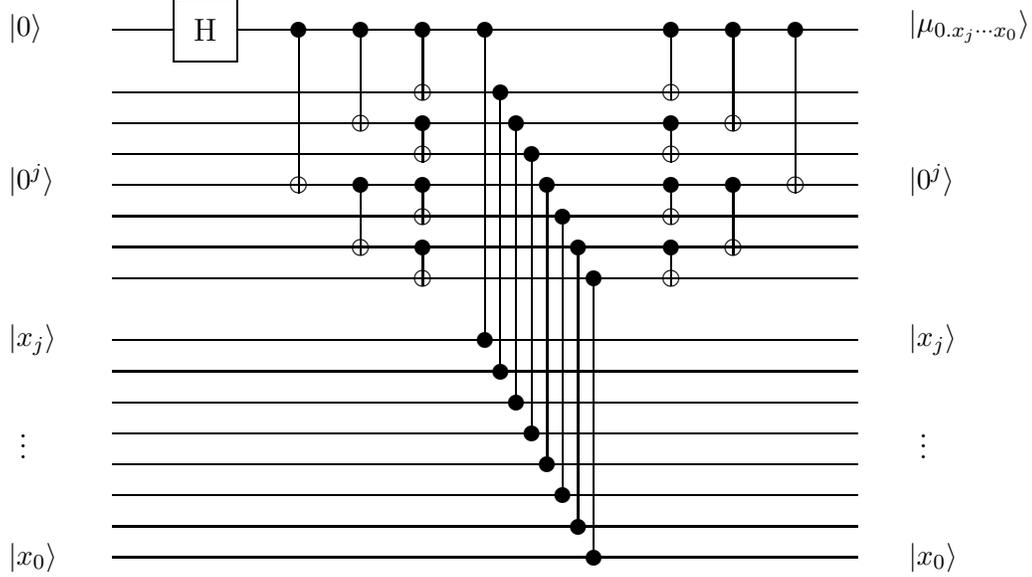


Figure 1: Quantum circuit for the exact preparation of $|\mu_{0.x_j\dots x_0}\rangle$.

From this, it may be verified that the circuit acts as indicated. The depth of this circuit is $O(\log n)$ and the size is $O(n)$.

If such a circuit is to be applied for each value of $j \in \{0, 1, \dots, n-1\}$, in order to perform the mapping (7), then the qubits $|x_{n-1}\rangle, \dots, |x_1\rangle, |x_0\rangle$ must first be copied several times ($n-i$ times for $|x_i\rangle$) to allow the controlled phase shift gates to operate in parallel. This may be performed (and inverted appropriately) in size $O(n^2)$ and depth $O(\log n)$ in the most obvious way. We conclude that the transformation (7) can be performed by circuits of size $O(n^2)$ and depth $O(\log n)$ in the exact case.

In order to reduce the size of the circuit in the approximate case, we use a similar procedure, except we only perform the controlled phase shifts when the phase θ is significant. An illustration of such a circuit is given in Figure 2. Here k denotes the number of significant phase shift gates that are used. The condition $\| |\mu_{0.x_j\dots x_0}\rangle - |\mu_{0.x_j\dots x_{j-k+1}}\rangle \| \in (\varepsilon/n)^{O(1)}$ occurs when $k \in O(\log(n/\varepsilon))$. With such a setting of k , the precision of the approximation of $|\mu_{0.x_{n-1}\dots x_0}\rangle \cdots |\mu_{0.x_0}\rangle$ can be $O(\varepsilon)$. Note that the size of the resulting circuit is $O(n \log(n/\varepsilon))$ and the depth is $O(\log \log(n/\varepsilon))$.

3.2 Copying a Fourier state in parallel

In this section, we show how to efficiently produce k copies of an n -qubit Fourier state from one copy. This is a unitary operation that acts on k n -qubit registers (thus kn qubits in all) and maps $|\psi_x\rangle|0^n\rangle \cdots |0^n\rangle$ to $|\psi_x\rangle|\psi_x\rangle \cdots |\psi_x\rangle$ for all $x \in \{0, 1\}^n$. The copying circuit will be exact and have size $O(kn)$ and depth $O(\log(kn))$. The setting of k will be $O(\log(n/\varepsilon))$.

Let us begin by considering the problem of producing two copies of a Fourier state from one. First, define the (*reversible*) *addition* and (*reversible*) *subtraction* operations as the mappings

$$\begin{aligned} |x\rangle|y\rangle &\mapsto |x\rangle|y+x\rangle \\ |x\rangle|y\rangle &\mapsto |x\rangle|y-x\rangle \end{aligned}$$

The resulting state will be $|\psi_x\rangle \cdots |\psi_x\rangle$.

Now, to implement the prefix addition and telescoping subtraction, note that they are inverses of each other. This means that it is sufficient to implement each one efficiently by a classical (non-reversible) circuit, and then combine these to produce a reversible circuit by standard techniques in reversible computing [5]. The telescoping subtraction clearly consists of $k - 1$ subtractions that can be performed in parallel, so the nonreversible size and depth bounds are $O(kn)$ and $O(\log n)$ respectively.

The prefix addition is a little more complicated. It relies on a combination of well-known tools in classical circuit design. One of them is the following general result of Ladner and Fischer [26] about parallel prefix computations.

Theorem 5 (Ladner and Fischer) *For any associative binary operation \circ , the mapping*

$$(x_1, x_2, \dots, x_k) \mapsto (x_1, x_1 \circ x_2, \dots, x_1 \circ x_2 \circ \cdots \circ x_k) \quad (8)$$

can be computed by a circuit consisting of $(x, y) \mapsto (x, x \circ y)$ gates that has size $O(k)$ and depth $O(\log k)$.

Another tool is the so-called *three-two adder*, which is a circuit that takes three n -bit integers x, y, z as input and produces two n -bit integers s, c as output, such that $x + y + z = s + c$ (recall that addition is in modulo 2^n arithmetic). It is remarkable that a three-two adder can be implemented with *constant depth* and size $O(n)$. By combining two three-two adders, one can implement a size $O(n)$ and depth $O(1)$ *four-two adder*, that performs the mapping $(x, y, z, w) \mapsto (x, y, s, c)$, where $x + y + z + w = s + c$. Now, consider the *pairwise representation* of each n -bit integer z as a pair of two n -bit integers (z', z'') such that $z = z' + z''$. This representation is not unique, but it is easy to convert to and from the pairwise representation: the respective mappings are $z \mapsto (z, 0^n)$ and $(z', z'') \mapsto z' + z''$. The useful observation is that the four-two adder performs integer addition in the pairwise representation scheme, and it does so in constant depth and size $O(n)$.

Now, the following procedure computes prefix addition in size $O(kn)$ and depth $O(\log k + \log n) = O(\log(kn))$. The input is (x_1, x_2, \dots, x_k) .

1. Convert the k integers into their pairwise representation.
2. Apply the parallel prefix circuit of Theorem 5 to perform the prefix additions in the pairwise representation scheme.
3. Convert the k integers from their pairwise representation to their standard form.

The output will be $(x_1, x_1 + x_2, \dots, x_1 + x_2 + \cdots + x_k)$, as required.

Note that step 4 of the main algorithm has a circuit of identical size and depth to the one just described, as it is simply its inverse.

3.3 Estimating the phase of a Fourier state in parallel

Finally, we will discuss the third step of the main algorithm, which corresponds to the mapping

$$|\psi_x\rangle|\psi_x\rangle \cdots |\psi_x\rangle|x\rangle \mapsto |\psi_x\rangle|\psi_x\rangle \cdots |\psi_x\rangle|0\rangle \quad (9)$$

for $x \in \{0, 1\}^n$. The number of copies of $|\psi_x\rangle$ required for this step depends on the error bound ε ; we will require $k \in O(\log(n/\varepsilon))$ copies. As discussed in subsection 3.1, any Fourier basis state $|\psi_x\rangle$ may be decomposed as $|\psi_x\rangle = |\mu_{x2^{-1}}\rangle |\mu_{x2^{-2}}\rangle \cdots |\mu_{x2^{-n}}\rangle$. Thus, we may assume that we have k copies of each of the states $|\mu_{x2^{-j}}\rangle$.

First, for each $j = 1, \dots, n$, the circuit will simulate measurements of the k copies of $|\mu_{x2^{-j}}\rangle$ (in the bases discussed below) in order to obtain an approximation $l_j/4$ to the fractional part of $2^{-j}x$. The approximation is with respect to the function $|\cdot|_1$ defined as

$$|y|_1 = \min \{z \in [0, 1) : \text{either } y - z \in \mathbb{Z} \text{ or } y + z \in \mathbb{Z}\}$$

for $y \in \mathbb{R}$ (i.e., “modulo 1” distance). With high probability the approximations will result in l_1, \dots, l_n satisfying $|l_j/4 - 2^{-j}x|_1 < \frac{1}{4}$ for each j . The (simulated) measurements can be performed in parallel, and each l_j will be determined by considering the mode of the outcomes of the measurements and thus can be computed in parallel as well. Next the circuit will reconstruct an approximation \tilde{x} to x (in parallel) from l_1, \dots, l_n . The circuit then XORs this value of \tilde{x} to the register containing x , thereby “erasing” it with high probability. Finally, the circuit inverts the computation of this \tilde{x} to clean up any garbage from the computation. As in subsection 3.2, standard techniques may be used to implement these computations as reversible circuits. We now describe each of the above steps in more detail.

Let us first recall the following fact from probability theory (see, e.g., Goldreich [18]). If X_1, \dots, X_t are independent Bernoulli trials with probability p_X of success and Y_1, \dots, Y_t are independent Bernoulli trials with p_Y of success, where $p_X < p_Y$, then

$$\Pr \left[\sum_{i=1}^t X_i \geq \sum_{i=1}^t Y_i \right] < 2e^{-(p_Y - p_X)^2 t/2}.$$

Now, define

$$\begin{aligned} |b_0\rangle &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = |\mu_0\rangle, & |b_1\rangle &= \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle = |\mu_{\frac{1}{4}}\rangle, \\ |b_2\rangle &= \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = |\mu_{\frac{1}{2}}\rangle, & |b_3\rangle &= \frac{1}{\sqrt{2}}|0\rangle - \frac{i}{\sqrt{2}}|1\rangle = |\mu_{\frac{3}{4}}\rangle, \end{aligned}$$

and consider measurements of the states $|\mu_{x2^{-j}}\rangle$ in the bases $\{|b_0\rangle, |b_2\rangle\}$ and $\{|b_1\rangle, |b_3\rangle\}$ (these measurements correspond to measurements of the Pauli operators σ_x and σ_y , respectively). In particular, given that we have k copies of each $|\mu_{x2^{-j}}\rangle$, we suppose that each of the above two measurements is performed independently on $k/2$ of the copies. Let $l_j \in \{0, 1, 2, 3\}$ represent the basis state that occurs with the highest frequency in these measurements for each j , breaking ties arbitrarily. We claim that the inequality $|l_j/4 - 2^{-j}x|_1 < \frac{1}{4}$ is satisfied with high probability:

$$\Pr \left[|l_j/4 - 2^{-j}x|_1 \geq \frac{1}{4} \right] < 4e^{-k/8}. \quad (10)$$

To prove that the inequality (10) holds, let us suppose that x and j are fixed, and let us define

$$p_0 = |\langle b_0 | \mu_{x2^{-j}} \rangle|^2, \quad p_1 = |\langle b_1 | \mu_{x2^{-j}} \rangle|^2, \quad p_2 = |\langle b_2 | \mu_{x2^{-j}} \rangle|^2, \quad p_3 = |\langle b_3 | \mu_{x2^{-j}} \rangle|^2.$$

These are the probabilities associated with the above measurements, meaning that the probability that a measurement of $|\mu_{x2^{-j}}\rangle$ in the $\{|b_0\rangle, |b_2\rangle\}$ basis yields 0 is p_0 , the probability that the measurement yields 2 is p_2 , and similar for p_1 and p_3 when the measurement in the $\{|b_1\rangle, |b_3\rangle\}$ basis

is performed. Now, note the following two facts: (i) it must be that $\max\{p_0, p_1, p_2, p_3\} \geq 1/2 + \sqrt{2}/4$ (for any choice of x and j), and (ii) if $|l/4 - 2^{-j}x|_1 \geq \frac{1}{4}$, then we must have $p_l \leq 1/2$. Therefore, if the inequality is not satisfied for some j (i.e., if $|l_j/4 - 2^{-j}x|_1 \geq \frac{1}{4}$), then it must be the case that $p_{l'} - p_{l_j} \geq \sqrt{2}/4$ for some different value of $l' \neq l_j$. Based on the inequalities above, we conclude that a very improbable event has taken place: the probability of the result l_j appearing more frequently than l' is at most $2e^{-k/8}$. Unless $|2^{-j}x|_1 \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$ there are at most two values of l_j that give $|l_j/4 - 2^{-j}x|_1 \geq \frac{1}{4}$, and so in this case we conclude that (10) holds. (In the special case $|2^{-j}x|_1 \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$, the inequality (10) follows trivially.)

From (10) we determine that $|l_j/4 - 2^{-j}x|_1 < \frac{1}{4}$ holds for all values of j with probability at least $1 - 4ne^{-k/8}$.

Now consider the following problem:

Input: $l_1, \dots, l_n \in \{0, 1, 2, 3\}$.
 Promise: There exists $x \in \{0, 1\}^n$ such that $|l_j/4 - 2^{-j}x|_1 < \frac{1}{4}$ for $j = 1, \dots, n$.
 Output: x satisfying the promise.

The following algorithm solves this problem:

1. Define

$$A_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A_1 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}.$$

2. Let $x_j = A_{l_j} A_{l_{j-1}} \cdots A_{l_1} [2, 1]$ for each j , and output $x = x_n \cdots x_1$.

Let us now demonstrate that the algorithm is correct. We note that it is straightforward to show that for a given input l_1, \dots, l_n there is at most one x satisfying the promise, and thus the solution is uniquely determined if the promise holds. To show that the algorithm computes this x correctly, we prove by induction on j that x_j is output correctly. The set $\{A_0, A_1, A_2, A_3\}$ is closed under matrix multiplication, so we must have that the first column of $A_{l_i} \cdots A_{l_1}$ is either

$$e_1 := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{or} \quad e_2 := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

for each i . Thus it suffices to prove that the first column of $A_{l_j} \cdots A_{l_1}$ is e_1 if $x_j = 0$ and is e_2 if $x_j = 1$. The base case is $j = 1$. Either $x_1 = 0$, in which case the fractional part of $2^{-1}x$ is 0, or $x_1 = 1$, in which case the fractional part of $2^{-1}x$ is $1/2$. By the promise, we must therefore have $l_1 \in \{0, 1\}$ in case $x_1 = 0$ and $l_1 \in \{2, 3\}$ in case $x_1 = 1$. Thus the first column of A_{l_1} is e_1 if $x_1 = 0$ and is e_2 if $x_1 = 1$ as required. Now suppose x_j, \dots, x_1 are output correctly. We want to show that the first column of $A_{l_{j+1}} \cdots A_{l_1}$ is e_1 if $x_{j+1} = 0$ and is e_2 if $x_{j+1} = 1$. There are four possibilities for the pair (x_{j+1}, x_j) that, along with the promise, give rise to the following implications:

$$\begin{aligned} x_{j+1} = 0, x_j = 0 &\Rightarrow l_{j+1} \in \{0, 1\} \\ x_{j+1} = 0, x_j = 1 &\Rightarrow l_{j+1} \in \{1, 2\} \\ x_{j+1} = 1, x_j = 0 &\Rightarrow l_{j+1} \in \{2, 3\} \\ x_{j+1} = 1, x_j = 1 &\Rightarrow l_{j+1} \in \{3, 0\} \end{aligned}$$

Suppose $x_j = 0$, implying that the first column of $A_{l_j} \cdots A_{l_1}$ is e_1 . If $x_{j+1} = 0$ then either $l_{j+1} = 0$ or $l_{j+1} = 1$, in either case implying that the first column of $A_{l_{j+1}} \cdots A_{l_1}$ is e_1 , as required. Similarly, if $x_{j+1} = 1$ then either $l_{j+1} = 2$ or $l_{j+1} = 3$, in either case implying that the first column of $A_{l_{j+1}} \cdots A_{l_1}$ is e_2 , as required. The case $x_j = 1$ is similar. Thus we have shown that the algorithm operates correctly.

The above algorithm lends itself well to parallelization, following from the parallel prefix method discussed in subsection 3.2; by Theorem 5 all values of $x_j = A_{l_j} A_{l_{j-1}} \cdots A_{l_1}[2, 1]$, $j = 1, \dots, n$ can be computed by a single circuit of size $O(n)$ and depth $O(\log n)$ (following from the fact that multiplication of the 2×2 matrices, in modulo 2 arithmetic, can of course be done by constant-size circuits).

It follows that the entire circuit for approximating the mapping (9) given $k \in O(\log(n/\varepsilon))$ copies of $|\psi_x\rangle$ has size $O(n \log(n/\varepsilon))$ and depth $O(\log n + \log \log(n/\varepsilon)) = O(\log n + \log \log(1/\varepsilon))$. It remains to argue that the circuit operates with error $O(\varepsilon)$. This follows from standard results based on ideas in [6] about converting quantum circuits that perform measurements and produce classical information with small error probability into unitary operations (without measurements) that can operate on data in superposition. It should be noted that a state $|\psi_x\rangle$ can be conserved throughout the computation to ensure that errors corresponding to different values of x are orthogonal.

4 New size bounds for the QFT

In this section, we prove Theorem 2. Let F_{2^n} denote the Fourier transform modulo 2^n , which acts on n qubits. The Hadamard transform is $H = F_2$.

The standard circuit construction for F_{2^n} can be described recursively as follows (where the two-qubit controlled-phase shift gates of the form $c-P(\theta)$ are defined in Section 2).

Standard recursive circuit description for F_{2^n} :

1. Apply $F_{2^{n-1}}$ to the first $n - 1$ qubits.
2. For each $j \in \{1, 2, \dots, n - 1\}$, apply $c-P(1/2^{n-j+1})$ to the j^{th} and n^{th} qubit.
3. Apply H to the n^{th} qubit.

The resulting circuit consists of $n(n - 1)/2$ two-qubit gates and n one-qubit gates.

Below is a more general recursive circuit description for F_{2^n} , parameterized by $m \in \{1, \dots, n - 1\}$. This coincides with the above circuit when $m = 1$. When $m > 1$, it can be verified that the circuit does not change very much. It has exactly the same gates, though the relative order of the two-qubit gates (which all commute with each other) changes.

Generalized recursive circuit description for F_{2^n} :

1. Apply $F_{2^{n-m}}$ to the first $n - m$ qubits.
2. For each $j \in \{1, 2, \dots, n - m\}$ and $k \in \{1, 2, \dots, m\}$, apply $c-P(1/2^{k-j+1})$ to the j^{th} and $(n - m + k)^{\text{th}}$ qubit.
3. Apply F_{2^m} to the last m qubits.

Our new quantum circuits are based on this generalized recursive construction with $m = \lfloor n/2 \rfloor$, except that they use a more efficient method for performing the transformation in Step 2. As is, Step 2 consists of $(n - m)m$ two-qubit gates, which is approximately $n^2/4$. The key observation is that Step 2 computes the mapping which, for $x \in \{0, 1\}^{n-m}$ and $y \in \{0, 1\}^m$, takes the state $|x\rangle|y\rangle$ to the state $(e^{2\pi i/2^n})^{x \cdot y}|x\rangle|y\rangle$, where $x \cdot y$ denotes the product of x and y interpreted as binary integers. From this, it can be shown that Step 2 can be computed using any classical method for integer multiplication in conjunction with some one-qubit phase shift gates (of the form $P(\theta)$, defined in Section 2).

The best asymptotic circuit size for integer multiplication, due to Schönhage and Strassen [33], is $O(n \log n \log \log n)$, which can be translated into a reversible computation of the same size that we will denote as S . For $x \in \{0, 1\}^{n-m}$ and $y \in \{0, 1\}^m$, S maps the state $|x\rangle|y\rangle|0^n\rangle$ to $|x\rangle|y\rangle|x \cdot y\rangle$. (There are $O(n)$ additional ancilla qubits that are not explicitly indicated. Each of these begins and ends in state $|0\rangle$.)

Improved Step 2 in general circuit description for F_{2^n} :

1. Apply S to the $2n$ qubits.
2. For each $k \in \{1, 2, \dots, n\}$ apply $P(1/2^k)$ to the $(n + k)^{\text{th}}$ qubit.
3. Apply S^{-1} to the $2n$ qubits.

Using this improved Step 2 in the generalized recursive circuit description for F_{2^n} results in a total number of gates that satisfies the recurrence

$$T_n = T_{\lfloor n/2 \rfloor} + T_{\lfloor n/2 \rfloor} + O(n \log n \log \log n), \tag{11}$$

which implies that $T_n \in O(n(\log n)^2 \log \log n)$. It is straightforward to also show that the circuit has depth $O(n)$ and width $O(n)$ (where ancilla qubits are counted as part of the width).

5 Factoring via logarithmic-depth quantum circuits

In this section we discuss a simple modification of Shor’s factoring algorithm that factors integers in polynomial time using logarithmic-depth quantum circuits. It is important to note that we are not claiming the existence of logarithmic-depth quantum circuits that take as input some integer n and output a non-trivial factor of n with high probability—the method will require (polynomial time) classical pre-processing and post-processing that is not known to be parallelizable. The motivation for this approach is that, under the assumption that quantum computers can be build, one may reasonably expect that quantum computation will be expensive while classical computation will be inexpensive.

The main bottleneck of the quantum portion of Shor’s factoring algorithm is the modular exponentiation. Whether or not modular exponentiation can be parallelized is a long-standing open question, and we do not address this question here. Instead, we show that sufficient classical pre-processing allows parallelization of the part of the quantum circuit associated with the modular exponentiation. Combined with logarithmic-depth circuits for quantum Fourier transform, we obtain the result claimed in Theorem 3.

In order to describe our method, let us briefly review Shor's factoring algorithm, including the reduction from factoring to order-finding. It is assumed the input is a n -bit integer N that is odd and composite.

1. (Classical) Randomly select $a \in \{2, \dots, N-1\}$. If $\gcd(a, N) > 1$ then output $\gcd(a, N)$, otherwise continue to step 2.
2. (Quantum) Attempt to find information about the order of a in \mathbb{Z}_N :
 - a. Initialize a $2n$ -qubit register and an n -qubit register to state $|0\rangle|0\rangle$.
 - b. Perform a Hadamard transform on each qubit of the first register.
 - c. (Modular exponentiation step.) Perform the unitary mapping:
$$|x\rangle|0\rangle \mapsto |x\rangle|a^x \bmod N\rangle.$$
 - c. Perform the quantum Fourier transform on the first register and measure (in the computational basis). Let y denote the result.
3. (Classical) Use the continued fraction algorithm to find relatively prime integers k and r such that $0 \leq k < r < N$ and $|y/2^m - k/r| \leq 2^{-2n}$. If $a^r \equiv 1 \pmod{N}$ then continue to step 4, otherwise repeat step 2.
4. (Classical) If r is even, compute $d = \gcd(a^{r/2} - 1, N)$ and output d if it is a nontrivial factor of N . Otherwise go to step 1.

The key observation is that much of the work required for the modular exponentiation step can be shifted to the classical computation in step 1 of the procedure. In step 1, the powers $b_0 = a$, $b_1 = a^2 \bmod N$, $b_2 = a^{2^2} \bmod N$, \dots , $b_{2n-1} = a^{2^{2n-1}} \bmod N$ can be computed in polynomial-time. With this information available in step 2, the modular exponentiation step reduces to applying a unitary operation that maps $|b_0\rangle|b_1\rangle \cdots |b_{2n-1}\rangle|x\rangle|0\rangle$ to $|b_0\rangle|b_1\rangle \cdots |b_{2n-1}\rangle|x\rangle|b_0^{x_0} \cdot b_1^{x_1} \cdots b_{2n-1}^{x_{2n-1}} \bmod N\rangle$. This is essentially an iterated multiplication problem, where one is given $2n$ n -bit integers $b_0^{x_0}, b_1^{x_1}, \dots, b_{2n-1}^{x_{2n-1}}$ as input and the goal is to compute their product. The most straightforward way to do this is to perform pairwise multiplications following the structure of a binary tree with $2n$ leaves. Each multiplication can be performed with depth $O(\log n)$ and size $O(n^2)$. The underlying binary tree has depth $\log(2n)$ and $2n-1$ internal nodes. Thus, the entire process can be performed with depth $O((\log n)^2)$ and size $O(n^3)$.

There are alternative methods for performing iterated multiplication achieving various combinations of depth and size. In particular, it was proved by Beame, Cook and Hoover [4] that a product such as we have above can be computed by $O(\log n)$ depth boolean circuits of size $O(n^5(\log n)^2)$. While $O(n^5 \log n)$ qubits may seem a high price to pay in order to save a factor of $O(\log n)$ in the circuit depth, the result has an interesting consequence regarding simulations of logarithmic-depth quantum circuits: if logarithmic-depth quantum circuits can be simulated in polynomial time, then factoring can be done in polynomial time as well. It should be noted that the circuits of Beame, Cook and Hoover are not logspace-uniform but rather are polynomial-time uniform; the best known bound on circuit depth for iterated products in the case of logspace uniform circuits is $O(\log n \log \log n)$ due to Reif [31].

6 Lower bounds

Logarithmic-depth lower bounds for *exact* computations with two-qubit gates are fairly easy to obtain, based on the fact that the state of some output qubit (usually) critically depends on every input qubit. Since, by Eq. 4, the last qubit of $|\psi_{x_{n-1}\dots x_1x_0}\rangle$ is in state $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.x_{n-1}\dots x_1x_0)}|1\rangle)$, its value depends on all n input qubits to the QFT when its input state is $|x_{n-1}\dots x_1x_0\rangle$. The depth of the circuit must be at least $\log n$ for this to be possible. This lower bound proof applies not only to the QFT, but also to QFS computations (which are defined in Section 2). This is because the output of a QFS on input $|x\rangle|0\rangle$ includes the state $|\psi_x\rangle$.

On the other hand, *approximate* computations can sometimes be performed with much lower depth than their exact counterparts. For example, in Section 3.1, it is shown that a QFS can be computed with precision ε by a quantum circuit with depth $O(\log \log(n/\varepsilon))$. Note that this is $O(\log \log n)$ whenever $\varepsilon \in 1/n^{O(1)}$. Although this suggests that it is conceivable for a sub-logarithmic-depth circuit to approximate the QFT with precision $1/n^{O(1)}$, Theorem 4 implies that this is not possible. We now prove this theorem.

Let C be a quantum circuit that approximates the *inverse* QFT with precision $\frac{1}{10}$. In this section, since we will need to consider distances between mixed states, we adopt the *trace distance* as a measure of distance (see, e.g., [15]). The trace distance between two states with respective density operators ρ and σ is given as

$$D(\rho, \sigma) = \frac{1}{2}\text{Tr}|\rho - \sigma|, \quad (12)$$

where, for an operator A , $|A| = \sqrt{A^\dagger A}$. For a pair of pure states $|\phi\rangle$ and $|\phi'\rangle$, their trace distance is $\sqrt{1 - |\langle\phi|\phi'\rangle|^2}$, which is upper bounded by their Euclidean distance.

On input $|\psi_{x_{n-1}\dots x_1x_0}\rangle$, the output state of C contains an approximation of $|x_{n-1}\dots x_1x_0\rangle$. In particular, one of the output qubits of C should be in a state that is an approximation of $|x_{n-1}\rangle$ within $\frac{1}{10}$. Let us refer to this as the *high-order* output qubit of C . If the depth of C is less than $\log n$ then the high-order output qubit of C cannot depend on all n of its input qubits. Let $k \in \{0, 1, \dots, n-1\}$ be such that the high-order output qubit does not depend on the k^{th} input qubit (where we index the input qubits right to left starting from 0). Let $r = n - k - 1$.

Set $z = 2^n - 1$, which is $11\dots 1 = 1^n$ in binary. Following Eq. 6, $|\psi_z\rangle$ can be written as

$$|\psi_z\rangle = |\mu_{0.1}\rangle|\mu_{0.11}\rangle \cdots |\mu_{0.1^n}\rangle. \quad (13)$$

Consider the state $|\psi_{z+2^r}\rangle$. Since $z + 2^r = 0^{n-r}1^r \pmod{2^n}$,

$$|\psi_{z+2^r}\rangle = |\mu_{0.1}\rangle|\mu_{0.11}\rangle \cdots |\mu_{0.1^r}\rangle|\mu_{0.01^r}\rangle|\mu_{0.001^r}\rangle \cdots |\mu_{0.0^{n-r}1^r}\rangle. \quad (14)$$

Note that, on input $|\psi_z\rangle$, the high-order output qubit of C approximates $|1\rangle$ with precision $\frac{1}{10}$; whereas, on input $|\psi_{z+2^r}\rangle$, the high-order output qubit of C approximates $|0\rangle$ with precision $\frac{1}{10}$.

Now, we consider a state $|\psi'_z\rangle$, which has an interesting relationship with both $|\psi_z\rangle$ and $|\psi_{z+2^r}\rangle$. Define

$$|\psi'_z\rangle = |\mu_{0.1}\rangle|\mu_{0.11}\rangle \cdots |\mu_{0.1^r}\rangle|\mu_{0.01^r}\rangle|\mu_{0.1^{r+2}}\rangle|\mu_{0.1^{r+3}}\rangle \cdots |\mu_{0.1^n}\rangle. \quad (15)$$

The states $|\psi'_z\rangle$ and $|\psi_z\rangle$ are identical, except in their k^{th} qubit positions (which are orthogonal: $|\mu_{0.01^r}\rangle$ vs. $|\mu_{0.1^{r+1}}\rangle$). Since the high-order output qubit of C does not depend on its k^{th} input

qubit, it is the same for input $|\psi'_z\rangle$ as for input $|\psi_z\rangle$. Therefore, the state of the high-order output qubit of C on input $|\psi'_z\rangle$ is within $\frac{1}{10}$ of $|1\rangle$.

On the other hand, the trace distance between $|\psi'_z\rangle$ and $|\psi_{z+2r}\rangle$ can be calculated to be below 0.7712, as follows. The two states are identical in qubit positions $n-1, n-2, \dots, k$. In qubit position $k-1$, the two states differ by an angle of $\frac{\pi}{4}$, in qubit position $k-2$ the two states differ by an angle of $\frac{\pi}{8}$, and so on. Therefore,

$$\begin{aligned} \langle \psi'_z | \psi_{z+2r} \rangle &= \langle \mu_{0.1r+2} | \mu_{0.001r} \rangle \langle \mu_{0.1r+3} | \mu_{0.0001r} \rangle \cdots \langle \mu_{0.1n} | \mu_{0.0^{n-r}1r} \rangle \\ &= \cos\left(\frac{\pi}{2^2}\right) \cos\left(\frac{\pi}{2^3}\right) \cdots \cos\left(\frac{\pi}{2^{n-k-1}}\right) \\ &> \cos\left(\frac{\pi}{2^2}\right) \cos\left(\frac{\pi}{2^3}\right) \cos\left(\frac{\pi}{2^4}\right) \cdots \\ &> 0.6366, \end{aligned}$$

where the numerical value for the last inequality is proved in Lemma 6 (below). This implies that the trace distance between $|\psi'_z\rangle$ and $|\psi_{z+2r}\rangle$ is less than $\sqrt{1 - (0.6366)^2} = 0.7712$. Since the trace distance is contractive, it follows that the state of the high-order output of C on input $|\psi'_z\rangle$ has trace distance less than 0.7712 from the state of high-order output of C on input $|\psi_{z+2r}\rangle$. But, by the triangle inequality, this implies that the trace distance between $|0\rangle$ and $|1\rangle$ is less than $\frac{1}{10} + 0.7712 + \frac{1}{10} < 1$, which is a contradiction, since $|0\rangle$ and $|1\rangle$ are orthogonal. This completes the proof of Theorem 4.

Lemma 6 $\cos\left(\frac{\pi}{2^2}\right) \cos\left(\frac{\pi}{2^3}\right) \cos\left(\frac{\pi}{2^4}\right) \cdots > 0.6366$.

Proof: We first lower bound the tails of the above infinite product by showing that, for any $i \geq 1$, $\cos\left(\frac{\pi}{2^{i+1}}\right) \cos\left(\frac{\pi}{2^{i+2}}\right) \cos\left(\frac{\pi}{2^{i+3}}\right) \cdots > 1 - \frac{\pi^2}{6 \cdot 4^i}$. Since, for $t > 0$, $\cos(t) > 1 - \frac{t^2}{2}$,

$$\begin{aligned} \cos\left(\frac{\pi}{2^{i+1}}\right) \cos\left(\frac{\pi}{2^{i+2}}\right) \cos\left(\frac{\pi}{2^{i+3}}\right) \cdots &> \left(1 - \frac{\pi^2}{2 \cdot 4^{i+1}}\right) \left(1 - \frac{\pi^2}{2 \cdot 4^{i+2}}\right) \left(1 - \frac{\pi^2}{2 \cdot 4^{i+3}}\right) \cdots \\ &\geq 1 - \frac{1}{2} \left(\frac{\pi^2}{4^{i+1}} + \frac{\pi^2}{4^{i+2}} + \frac{\pi^2}{4^{i+3}} + \cdots\right) \\ &= 1 - \frac{\pi^2}{6 \cdot 4^i}. \end{aligned}$$

Now it follows that, for any $i \geq 1$, $\cos\left(\frac{\pi}{2^2}\right) \cos\left(\frac{\pi}{2^3}\right) \cos\left(\frac{\pi}{2^4}\right) \cdots > \cos\left(\frac{\pi}{2^2}\right) \cdots \cos\left(\frac{\pi}{2^i}\right) \left(1 - \frac{\pi^2}{6 \cdot 4^i}\right)$. Setting $i = 8$ in this inequality gives the numerical lower bound. \blacksquare

7 Other moduli

In this section we discuss the quantum Fourier transform with respect to moduli that are not powers of 2. First we briefly sketch a method for performing (in parallel) the QFT for an arbitrary modulus that uses the QFT with a power of 2 modulus as a black box. We then discuss Shor's original method for performing the QFT with respect to a "smooth" modulus, and mention how this method may be parallelized as well.

7.1 Arbitrary moduli

Consider the QFT with respect to an arbitrary modulus m . In this subsection we note that it is possible to approximate such a QFT with high accuracy in parallel using circuits for the quantum

Fourier transform modulo 2^k for $k = \lceil \log m \rceil + O(1)$. Using the circuits for the quantum Fourier transform modulo 2^k described previously, we have that for any ε and modulus m there exists a depth $O(\log n \log \log(1/\varepsilon))$ quantum circuit that approximates the QFT modulo m to within accuracy ε for $n = \lceil \log m \rceil$. The size of the circuit is polynomial in $n + \log(1/\varepsilon)$.

The method exploits a relation between QFTs with different moduli that was used by Hales and Hallgren [22] in regard to the Fourier Sampling problem (see also Høyer [24] for an extension and simplified proof).

The basic components of the technique are as follows:

1. Create a Fourier state with modulus m , which is the mapping

$$|x\rangle|0\rangle \mapsto |x\rangle|\psi_x\rangle.$$

2. Copy the Fourier state, which is the mapping

$$|x\rangle|\psi_x\rangle|0\rangle \cdots |0\rangle \mapsto |x\rangle|\psi_x\rangle|\psi_x\rangle \cdots |\psi_x\rangle.$$

3. Apply the inverse Fourier transform modulo 2^k on each state $|\psi_x\rangle$, which is the mapping

$$|x\rangle|\psi_x\rangle \cdots |\psi_x\rangle \mapsto |x\rangle \left(F_{2^k}^\dagger |\psi_x\rangle \right) \cdots \left(F_{2^k}^\dagger |\psi_x\rangle \right).$$

4. For each (computational basis state) y occurring among the collections of qubits on which F_k^\dagger was performed, compute $\text{round}(y m 2^{-k}) \bmod m$, and compute the mode of these results. With high probability the result will be x . (A reasonably straightforward calculation shows that observation of $F_{2^k}^\dagger |\psi_x\rangle$ in the computational basis yields some y with $\text{round}(y m 2^{-k}) = x$ with probability greater than $1/2 + \delta$ for some constant δ .) XOR this result to the qubits in state $|x\rangle$, and reverse the computation of each $\text{round}(y m 2^{-k})$ and y . With high probability the mapping

$$|x\rangle \left(F_{2^k}^\dagger |\psi_x\rangle \right) \cdots \left(F_{2^k}^\dagger |\psi_x\rangle \right) \mapsto |0\rangle \left(F_{2^k}^\dagger |\psi_x\rangle \right) \cdots \left(F_{2^k}^\dagger |\psi_x\rangle \right)$$

has been performed.

5. Reverse steps 3 and 2, giving the mapping $|0\rangle \left(F_{2^k}^\dagger |\psi_x\rangle \right) \cdots \left(F_{2^k}^\dagger |\psi_x\rangle \right) \mapsto |0\rangle|\psi_x\rangle|0\rangle \cdots |0\rangle$.

Unfortunately some of the methods used in the power of 2 case (such as using three-two adders and approximating the individual qubits of the Fourier basis states) do not seem to work in this case, which results in the slightly worse depth bound. The overall size bound increases as well, but is still polynomial.

It is interesting to note that this method does not require the larger modulus to be a power of 2—effectively the method shows that the QFT modulo m for any modulus m can be efficiently approximated given a black box that approximates the QFT modulo m' for any sufficiently large m' . The technical details regarding this method will appear in the final version of this paper.

7.2 Shor’s “mixed-radix” QFT

We conclude with a brief discussion of Shor’s original “mixed radix” method for computing the quantum Fourier transform, as it too can be parallelized (although to our knowledge not as efficiently as the power-of-2 case discussed previously in this paper).

Shor’s original method for computing the QFT is based on the Chinese Remainder Theorem and its consequences regarding \mathbb{Z}_m for given modulus m . Here the modulus is $m = m_1 m_2 \cdots m_k$ for m_1, \dots, m_k pairwise relatively prime and $m_j \in O(\log m)$. Thus $k \in O(\log m / \log \log m)$ is somewhat less than the number of bits of m , and each m_j has length logarithmic in the length of m . Taking m_j to be the j^{th} prime results in a sufficiently dense collection of moduli m for factoring [34] (see Rosser and Schoenfeld [32] for explicit bounds and a detailed analysis of such bounds).

Although stated somewhat differently by Shor, the mixed radix QFT method may be described as follows:

1. For $j = 1, \dots, k$ define $f_j = \frac{m}{m_j}$ and set $g_j \in \{0, \dots, m_j - 1\}$ such that $g_j \equiv f_j^{-1} \pmod{m_j}$.
2. Define C to be the (reversible) operator acting as follows for each $x \in \{0, \dots, m - 1\}$:

$$C : |x\rangle \mapsto |(x \bmod m_1), \dots, (x \bmod m_k)\rangle$$

3. Define A to be a (reversible) operator such that

$$A : |x_1, \dots, x_k\rangle \mapsto |g_1 x_1, \dots, g_k x_k\rangle$$

for each $(x_1, \dots, x_k) \in \{0, \dots, m_1 - 1\} \times \cdots \times \{0, \dots, m_k - 1\}$.

4. Let F_m and F_{m_j} denote the QFT for moduli m and m_j , $j = 1, \dots, k$, respectively. Then the following relation holds:

$$F_m = C^\dagger (F_{m_1} \otimes \cdots \otimes F_{m_k}) A C. \tag{16}$$

Thus, to perform the QFT modulo m on $|x\rangle$, first convert x to its *modular representation* (x_1, \dots, x_k) using the operator C , multiply each x_j by g_j (modulo m_j), perform the QFT modulo m_j independently on coefficient j (for each j), then apply the inverse of C to convert back to the ordinary representation of elements in $\{0, \dots, m - 1\}$.

The numbers computed in step 1 are used in the standard proof of the Chinese Remainder Theorem: given x_1, \dots, x_k , we may compute $x \in \{0, \dots, m - 1\}$ satisfying $x \equiv x_j \pmod{m_j}$ for each j by taking $x = \sum_{j=1}^k f_j g_j x_j \bmod m$. Thus the operator C can be implemented efficiently, since the mappings $x \mapsto ((x \bmod m_1), \dots, (x \bmod m_k))$ and $((x \bmod m_1), \dots, (x \bmod m_k)) \mapsto x$ are efficiently computable (e.g., with size $O(\log^2 m)$ circuits [2]). In the present case C can be parallelized to logarithmic depth, since each of the moduli are small. Similarly, the operator A can be parallelized to logarithmic depth.

To see that the relation (16) holds, we may simply examine the action of the operator on the right hand side on computational basis states:

$$\begin{aligned}
& C^\dagger(F_{m_1} \otimes \cdots \otimes F_{m_k})AC|x\rangle \\
&= C^\dagger(F_{m_1} \otimes \cdots \otimes F_{m_k})|g_1x_1, \dots, g_kx_k\rangle \\
&= \frac{1}{\sqrt{m}}C^\dagger \sum_{y_1, \dots, y_k} \exp(2\pi ig_1x_1y_1/m_1) \cdots \exp(2\pi ig_kx_ky_k/m_k)|y_1, \dots, y_k\rangle \\
&= \frac{1}{\sqrt{m}}C^\dagger \sum_{y_1, \dots, y_k} \exp(2\pi i(f_1g_1x_1y_1 + \cdots + f_kg_kx_ky_k)/m)|y_1, \dots, y_k\rangle \\
&= \frac{1}{\sqrt{m}} \sum_{y_1, \dots, y_k} \exp(2\pi i(f_1g_1x_1y_1 + \cdots + f_kg_kx_ky_k)/m)|f_1g_1y_1 + \cdots + f_kg_ky_k \pmod{m}\rangle \\
&= \frac{1}{\sqrt{m}} \sum_y \exp(2\pi ixy/m)|y\rangle \\
&= F_m|x\rangle
\end{aligned}$$

Finally, each of the independent QFTs modulo m_1, \dots, m_k can of course be done in parallel. Here, however, a problem arises if our goal is to parallelize the entire process. Originally Shor suggests implementing each of these operations by circuits of size m_j (not $\log m_j$), since any quantum operation can be computed by circuits with exponential-size quantum circuits [3]. This results in a linear-depth circuit overall, although the circuit will be exact.

However, we may try to compute each F_{m_j} more efficiently. There are a few possibilities for how to do this, all (apparently) requiring approximations of each F_{m_j} . First, we may apply the method of Kitaev [25] to approximate these QFTs. Alternately, we may use the arbitrary modulus method we have proposed in section 7.1. Finally, we have noted that this method works for any two moduli (not just for the larger modulus a power of 2) so that we may in fact recurse using the mixed-radix method to approximate each F_{m_j} .

In all cases, our analysis has revealed that the mixed radix method results in worse size and/or depth bounds than the power of 2 method presented in Section 3.

8 Conclusion

We have proved several new bounds on the circuit complexity of approximating the quantum Fourier transform, and have applied these bounds to the problem of factoring using quantum circuits. There are several related open questions, a few of which we will now discuss.

First, is it possible to perform the quantum Fourier transform *exactly* using logarithmic- or poly-logarithmic-depth quantum circuits? The best currently known upper bound on the depth of the exact QFT is linear in the number of input qubits.

Next, can the efficiency of our techniques be improved significantly? We have concentrated on asymptotic analyses of our circuits, and we believe it is certain that our circuits can be optimized significantly for “interesting” input sizes (perhaps several hundred to a few thousand qubits).

Finally, the fact that the quantum Fourier transform can be performed in logarithmic depth suggests the following question: are there interesting natural problems in BQNC (bounded-error quantum NC) not known to be in NC or RNC? For instance, computing the gcd of two n -bit

integers and computing $a^b \bmod c$ and $a^{-1} \bmod c$ for n -bit integers a , b , and c is not known to be possible using polynomial-size circuits with depth poly-logarithmic in n in the classical setting. Are there logarithmic- or poly-logarithmic-depth quantum circuits for these problems? Greenlaw, Hoover and Ruzzo [19] list several other problems not known to be classically parallelizable, all of which are interesting problems to consider in the quantum setting.

Acknowledgments

We thank Wayne Eberly for helpful discussions on classical circuit complexity, and Chris Fuchs and Patrick Hayden for an informative discussion regarding quantum state distance measures.

References

- [1] D. Aharonov and M. Ben-Or. Polynomial simulations of decohered quantum computers. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, 1996.
- [2] E. Bach and J. Shallit. *Algorithmic Number Theory, Volume I: Efficient Algorithms*. MIT Press, 1996.
- [3] A. Barenco, C. H. Bennett, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review Letters A*, 52:3457–3467, 1995.
- [4] P. Beame, S. Cook, and H. J. Hoover. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15(4):994–1003, 1986.
- [5] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17:525–532, 1973.
- [6] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.
- [7] D. Boneh and R. Lipton. Quantum cryptanalysis of hidden linear functions. In *Advances in Cryptology – Crypto’95*, volume 963 of *Lecture Notes in Computer Science*, pages 242–437. Springer-Verlag, 1995.
- [8] G. Brassard, P. Høyer, and A. Tapp. Quantum counting. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 820–831, 1998.
- [9] R. Cleve. A note on computing quantum Fourier transforms by quantum programs. Manuscript. Available at <http://www.cpsc.ucalgary.ca/~cleve/papers.html>, 1994.
- [10] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society, London*, A454:339–354, 1998.
- [11] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

- [12] J. W. Cooley. The re-discovery of the fast Fourier transform algorithm. *Mikrochimica Acta*, 3:33–45, 1987.
- [13] J. W. Cooley and J. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.
- [14] D. Coppersmith. An approximate Fourier transform useful in quantum factoring. Technical Report RC19642, IBM, 1994.
- [15] C. Fuchs. *Distinguishability and Accessible Information in Quantum Theory*. PhD thesis, University of New Mexico, 1995. Los Alamos Preprint Archive, quant-ph/9601020.
- [16] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.
- [17] C. F. Gauss. Theoria interpolationis methodo nova tractata. In *Werke III, Nachlass*, pages 265–330. Königliche Gesellschaft der Wissenschaften, Göttingen, 1866. Reprinted by Georg Olms Verlag, Hildesheim, New York, 1973.
- [18] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Springer, 1999.
- [19] R. Greenlaw, H. J. Hoover, and W. Ruzzo. *Limits to Parallel Computation*. Oxford University Press, 1995.
- [20] D. Grigoriev. Testing shift-equivalence of polynomials using quantum machines. In *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation*, pages 49–54, 1996.
- [21] L. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 212–219, 1996.
- [22] L. Hales and S. Hallgren. Quantum Fourier sampling simplified. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 330–338, 1999.
- [23] M. T. Heideman, D. H. Johnson, and S. Burris. Gauss and the history of the Fast Fourier Transform. *IEEE ASSP Magazine*, pages 14–21, 1984.
- [24] P. Høyer. *Quantum Algorithms*. PhD thesis, Odense University, Denmark, 2000.
- [25] A. Yu. Kitaev. Quantum measurements and the abelian stabilizer problem. Manuscript, 1995. Los Alamos Preprint Archive, quant-ph/9511026.
- [26] R.E. Ladner and M.J. Fischer. Parallel prefix computation. *Journal of the ACM*, 27(4):831–838, 1980.
- [27] D.K. Maslen and D.N. Rockmore. Generalized FFTs – a survey of some recent results. In L. Finkelstein and W. Kantor, editors, *Proceedings of the DIMACS Workshop on Groups and Computation*, pages 329–369, 1995.

- [28] C. Moore and M. Nilsson. Parallel quantum computation and quantum codes. Los Alamos Preprint Archive, quant-ph/9808027, 1998.
- [29] M. Mosca. Quantum searching and counting by eigenvector analysis. In *Proceedings of Randomized Algorithms, Workshop of MFCS 98*, 1998.
- [30] Y. Ofman. On the algorithmic complexity of discrete functions. *Cybernetics and Control Theory*, 7(7):589–591, 1963.
- [31] J. Reif. Logarithmic depth circuits for algebraic functions. *SIAM Journal on Computing*, 15(1):231–242, 1986.
- [32] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6:64–94, 1962.
- [33] A. Schönhage and V. Strassen. Schnelle multiplikation großer zahlen. *Computing*, 7:281–292, 1971.
- [34] P. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.
- [35] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

(The Los Alamos Preprint Archive may be found at <http://xxx.lanl.gov/> on the World Wide Web.)