Uniform Quantum Fourier Transforms from Fast Integer Multiplication

Zhou Kangyun

November 25, 2026

hexagon:2611.01104v1

Abstract

We construct polynomial-time uniform unitary circuits for the quantum Fourier transform on $\mathbb{Z}/2^q\mathbb{Z}$ using $O_\delta(q(\log\log q)^{1/2+\delta})$ elementary Clifford and T gates in total, for every fixed $\delta > 0$ and fixed or inverse-polynomial error. From OpenAI's November 2026 integer multiplication algorithm [4], we construct a polynomial-time uniform Boolean multiplier of size $O_\delta(n(\log n)^{1/2+\delta})$ by implementing its normalized Walsh transforms with bounded-fan-in gates. We use this multiplier to implement the product phases in the QFT circuit of Kayal, Mittal, Paraashar, and Saurabh [3]. A single phase-gradient register serves every product phase and costs $\mathrm{poly}(\log q)$ gates to prepare and uncompute. We also compute the rotation angle in their exact QFT circuit to d bits of precision in deterministic time polynomial in q and d, giving uniform descriptions of their exact and approximate circuits. With intermediate measurements and exact dyadic phases, one product phase and a semiclassical Fourier measurement implement the exact transform in $O(M(q) + q)$ operations on every branch, where $M(q)$ is the Boolean circuit size of q-bit multiplication.

1 Introduction

The quantum Fourier transform (QFT) is a basic building block of various quantum algorithms. At fixed or inverse-polynomial error, we reduce the total Clifford+T gate count for the q-qubit QFT to $o(q \log\log q)$.

On $\mathbb{Z}/2^q\mathbb{Z}$, the QFT F_q acts on each computational basis state $|x\rangle$, $0 \le x < 2^q$, by the rule

$$F_q|x\rangle = 2^{-q/2} \sum_{y=0}^{2^q-1} e^{2\pi i xy/2^q}|y\rangle.$$

The amplitude of $|y\rangle$ has magnitude $2^{-q/2}$ and phase angle $2\pi xy/2^q$. Cleve and Watrous used this dependence on xy to implement all phases between two qubit blocks with a single multiplication [1]. Ahokas, Cleve, and Hales observed that at finite accuracy it suffices to multiply the leading bits of the two blocks [2]. Kayal, Mittal, Paraashar, and Saurabh combined this truncation with exact Fourier decoding, counting each exact dyadic phase rotation as one gate [3].

At fixed or inverse-polynomial error, this recursion multiplies integers of only $O(\log q)$ bits. Using OpenAI's multiplier for these short products, we obtain the following bound on the total number of elementary Clifford+T gates.

Theorem 1 (Total Clifford+T gate count of the QFT)**.** *For every fixed $\delta > 0$, OpenAI's integer multiplication algorithm [4] yields a polynomial-time uniform family of unitary Clifford+T circuits for the quantum Fourier transform F_q on $\mathbb{Z}/2^q\mathbb{Z}$. For any fixed $0 < \epsilon < 1$, or for $\epsilon = q^{-c}$ with any fixed $c > 0$, the circuit approximates F_q to within error ϵ using*

$$O_\delta(q(\log\log q)^{1/2+\delta}) \tag{1}$$

elementary one- and two-qubit gates in total, including all gates required to prepare and uncompute auxiliary registers.

With an $O(n \log n)$ multiplier [5], the same construction gives an upper bound of $O(q \log\log q)$ total gates. For $0 < \delta < 1/2$, Theorem 1 improves this bound by a factor of $(\log\log q)^{1/2-\delta}$.[1]

Section 2 implements the QFT recursion of Kayal et al. over Clifford+T with the same asymptotic gate count. We compute the leaf rotation angles by direct enumeration and reuse one phase-gradient register across the tree. Each m-bit product phase then costs $O(M(m) + m)$ gates for a Boolean multiplier of size $M(m)$. Section 3 counts the bounded-fan-in Boolean gates for the normalized Walsh transforms and fixed-point arithmetic in OpenAI's multiplier, implementing its fixed permutations as wiring. Substituting the resulting uniform multiplier of size $M(m) = O_\delta(m(\log m)^{1/2+\delta})$ into the bound of Section 2 proves Theorem 1. Section 4 resolves the uniformity question of Kayal et al. [3, Section 5] by computing the rotation angle in their exact circuit to error 2^{-d} in deterministic time polynomial in q and d. With intermediate measurements and unit-cost exact dyadic phases, Section 5 implements the exact QFT using one product phase, a semiclassical Fourier measurement, and a conditional shift, for $O(M(q) + q)$ operations per branch.

Notation and circuit model. All logarithms are base two. Our elementary gates are H, S, T, CNOT, and their inverses, acting on arbitrary wires. Both Toffoli and SWAP admit constant-size decompositions over this gate set. For a circuit C acting on q input qubits and a auxiliary qubits, we measure error by the operator norm of the difference of the corresponding isometries:

$$\sup_{\|\psi\|=1} \left\| C(|\psi\rangle|0^a\rangle) - (F_q|\psi\rangle)|0^a\rangle \right\| \le \epsilon. \tag{2}$$

By tensoring with the identity, this guarantee extends automatically to inputs entangled with an arbitrary reference system. Given a target error ϵ, we set $d = \lceil \log(1/\epsilon) \rceil$. A circuit family is *polynomial-time uniform* if a single deterministic classical algorithm outputs the circuit description on input $(1^q, 1^d)$ in time polynomial in q and d. In the accuracy regimes of Theorem 1 we have $d = O(\log q)$, and the asymptotic bounds hold for any fixed ϵ or fixed exponent c.

2 From multiplication circuits to total gate count

The QFT circuit couples two blocks of qubits by computing the product of their integer labels, applying a phase determined by the product, and reversing the multiplication. Let $M(n)$ count the Boolean gates used by the multiplier to compute the full $2n$-bit product of two unsigned n-bit integers. Each Boolean gate reads at most a fixed number of bits, independent of n (bounded fan-in). The standard reversible simulation [3, Section 2] writes each gate's output to a fresh bit using $O(1)$ NOT, CNOT, and Toffoli gates. The exact Clifford+T decomposition of Toffoli [9, Eq. (1)] thus gives a constant Clifford+T cost per Boolean gate. We can therefore bound the quantum gate count in terms of $M(n)$ before choosing a particular multiplier in Section 3.

Proposition 2. *Suppose there is a polynomial-time uniform Boolean multiplier of size $M(n) = O(n(\log n)^\alpha)$ for some fixed $\alpha > 0$. Then at fixed or inverse-polynomial error, F_q admits a polynomial-time uniform unitary Clifford+T circuit of total size $O(q(\log\log q)^\alpha)$.*

2.1 The QFT construction of Kayal et al.

We review the approximate QFT circuit of Kayal et al. [3, Sections 3 and 4] because our implementation needs two features of their construction. Their decoder specifies the rotation angle that we compute in Section 2.2 and the precision it requires. Their recursion bounds the size and number of the products whose Clifford+T gate cost we analyze in Section 2.3.

[1] Related work gives upper bounds on the T-count of approximate QFT circuits [6, 7] and lower bounds on the number of T states required at finite accuracy [8].

The Cleve–Watrous recursion [1] splits a register into two nearly equal blocks of r and s qubits. Between the two recursive transforms it applies the phase $e^{2\pi i(u/2^r)(v/2^s)}$, where u labels the untouched block and v the transformed block. Following Ahokas, Cleve, and Hales [2], Kayal et al. retain only the ℓ leading bits of each operand whenever $r + s > 2\ell$. Each fraction changes by less than $2^{-\ell}$, so the operator-norm error at that split is at most $4\pi 2^{-\ell}$. In general, a product phase on two operands of at most m bits has the form

$$|u, v\rangle \mapsto e^{2\pi i uv/2^b}|u, v\rangle, \qquad 1 \le b \le 2m. \tag{3}$$

In the model of Kayal et al., each dyadic phase $P(2\pi a/2^b) = \mathrm{diag}(1, e^{2\pi i a/2^b})$ costs one gate. Computing uv, applying phases to its b lowest bits, and uncomputing the product then costs $O(M(m) + m)$ gates [3, Lemma 2.1].

For our target error we take $\ell = \lceil \log q \rceil + d + 9$. The recursion stops at blocks of width $m \le 2\ell$, called leaves. For $m < 64$ the elementary exact QFT in the dyadic gate set suffices. At larger leaves Kayal et al. implement an exact F_m by amplifying an approximate Fourier decoder to certainty and reversing the resulting clean circuit for $F_m^\dagger$ [3, Lemma 3.1]. This costs $O(m(\log m)^\alpha)$ gates when the decoder's extra rotation is counted at unit cost. If $q > 2\ell$, each leaf has width at least ℓ, so there are $O(q/\ell)$ leaves and $O(q/\ell)$ splits. Summing the leaf circuits and the products on ℓ-bit operands gives

$$O\!\left(\frac{q}{\ell}\big(M(\ell) + \ell(\log \ell)^\alpha\big)\right) = O\big(q(\log \ell)^\alpha\big). \tag{4}$$

Tracking wire labels through the recursion leaves one final permutation of cost $O(q)$. If $q \le 2\ell$, the circuit consists of a single leaf.

To see which angle the leaf decoder needs, let $|\phi_x\rangle = F_m|x\rangle$. For its approximate inverse $W^\dagger$, we choose W to be the same balanced recursion on m qubits with $k = \lceil \log m \rceil + 5$ leading bits per operand and exact Cleve–Watrous transforms on blocks of width at most $2k$. The split errors sum to $\|W - F_m\| \le 4\pi m 2^{-k} \le 1/2$.

The decoder keeps $|\phi_x\rangle$ for checking the answer. On a second copy it shifts the Fourier label to $x - r$, applies $W^\dagger$, and adds r to the candidate answer, using a uniform superposition of shifts r modulo 2^m. Averaging over the shifts gives the same success probability p_m for every input label. A flag qubit prepared in $B(\theta_m)|0\rangle = \cos\theta_m|0\rangle + \sin\theta_m|1\rangle$ sets the probability of marking a correct answer with flag 1 to $p_m \sin^2\theta_m = 1/4$. One step of amplitude amplification then recovers x with certainty [3, Lemmas 2.4 and 3.1]. Thus

$$p_m = 2^{-m}\sum_x \big|\langle x|W^\dagger F_m|x\rangle\big|^2, \qquad \theta_m = \arcsin\frac{1}{2\sqrt{p_m}}, \qquad \frac{3}{4} \le p_m \le 1. \tag{5}$$

The lower bound follows because the component of $W^\dagger|\phi_x\rangle$ orthogonal to $|x\rangle$ has norm at most $1/2$. The circuit copies the recovered x to a clean register and reverses the decoder. A product phase controlled by x erases the phases of the retained Fourier state, and Hadamards return it to $|0^m\rangle$. Swapping the registers completes $F_m^\dagger$. If W uses $G(m)$ gates and $a(m)$ auxiliary qubits, the full circuit uses $O(G(m) + M(m) + m)$ gates and $O(a(m) + M(m) + m)$ auxiliary qubits [3, Lemma 3.1].

Kayal et al. show how accurately the angle in (5) must be specified [3, Lemma 4.1 and Section 4.2]. With $J = SH$, the identity $JP(2\theta)J^\dagger = e^{i\theta}B(\theta)$ lets them replace each flag rotation by $JP(\varphi_m)J^\dagger$, where $\varphi_m = 2\pi a_m/2^\ell$ and $|\varphi_m - 2\theta_m| \le 2\pi 2^{-\ell}$. The scalar phases cancel between the decoder and its inverse. There are at most six rotations or adjoints per leaf, so rounding changes the full leaf unitary, including its workspace, by at most $12\pi 2^{-\ell}$ in operator norm. This bound permits workspace reuse across leaves. Summing over at most q leaves and at most q splits gives a total error on clean inputs of at most $16\pi q 2^{-\ell} < 2^{-d}/2$, leaving room for the phase-gradient preparation in Section 2.3.

2.2 Computing the leaf rotation angles

We compute the integer a_m for each leaf width that requires a flag rotation by directly evaluating the average in (5). At fixed or inverse-polynomial error, $m \le 2\ell = O(\log q)$, so we can enumerate all pairs of m-bit labels x, y in time polynomial in q.

For a given pair x, y, let $\varphi_W(x, y)$ be the phase angle of $\langle y|W|x\rangle$. We compute it by adding the retained product phases at the splits to the exact Fourier phases on the terminal blocks of W. Each split uses the input bits of the untouched block and the output bits of the transformed block. Since each matrix entry has magnitude $2^{-m/2}$, the overlap for a fixed x is

$$\langle x|W^\dagger F_m|x\rangle = 2^{-m} \sum_{y=0}^{2^m-1} \exp(i\,(2\pi xy/2^m - \varphi_W(x, y)))\,.$$

We square the magnitude of each overlap and average over x to obtain p_m. Evaluating the unit complex numbers to absolute error $2^{-h-O(1)}$ and accumulating the sums with $O(m + h)$ bits of precision gives p_m to error 2^{-h} in $2^{O(m)}\mathrm{poly}(m, h)$ bit operations.

The function $p \mapsto \arcsin(1/(2\sqrt{p}))$ has bounded derivative on $[3/4, 1]$. With $h = \ell + O(1)$, we can therefore compute θ_m/π to within $2^{-\ell-2}$. We round this approximation to the nearest multiple $a_m/2^\ell$, adding at most $2^{-\ell-1}$ to the error. The resulting integer a_m meets the precision requirement of Section 2.1 and specifies the phase gate $P(2\pi a_m/2^\ell)$ and its inverse in the circuit description.

2.3 Implementing the recursive QFT over Clifford+T

We show that the approximate QFT of Kayal et al. [3] retains its asymptotic gate count when implemented as a unitary Clifford+T circuit. We use one phase-gradient register throughout the recursion, both for the product phases and for the leaf rotations computed in Section 2.2. Nam, Su, and Maslov used such a register repeatedly in their approximate QFT [6, Section II]. At a node with short operands we add only to the corresponding high bits of the register, keeping the phase within the gate budget for that multiplication. For $B \ge 1$, write the B-qubit phase-gradient state as

$$|\gamma_B\rangle = 2^{-B/2} \sum_{y=0}^{2^B-1} e^{-2\pi iy/2^B}|y\rangle. \tag{6}$$

Lemma 2.1 (Phase-gradient implementation)**.** *For $1 \le b \le B$, the state $|\gamma_B\rangle$ implements a product phase (3) on m-bit operands using $O(M(m) + m)$ elementary Clifford+T gates, or a phase $P(2\pi a/2^b)$ using $O(b)$ such gates. Each use preserves the ideal state. If its preparation has vector error at most η up to a global phase, reusing the register throughout the circuit and reversing its preparation at the end adds at most 2η to the error on clean inputs.*

Proof. Adding z modulo 2^b to $|\gamma_b\rangle$ gives the phase $e^{2\pi iz/2^b}$ by phase kickback [10, Section 7]. Writing $y = 2^{B-b}h + r$ in (6) shows that the high b qubits of $|\gamma_B\rangle$ are in state $|\gamma_b\rangle$ and factor from the rest [11, Section IV.C]. Compute uv reversibly, add its b low bits to the register, and uncompute uv. The multiplier and its inverse cost $O(M(m) + m)$ gates. The ripple-carry adder of Cuccaro et al. [12] costs $O(b)$, with $b \le 2m$. A single-qubit phase uses the same adder with a constant a controlled by the data qubit.

Let V be the dyadic circuit and A its implementation with the gradient register. Since each addition restores its scratch qubits, $A(|\psi\rangle|\gamma_B\rangle) = (V|\psi\rangle)|\gamma_B\rangle$ for any joint state of the data and original workspace, also in the presence of a reference system. After aligning global phases, substituting the prepared state on the two sides changes each side by at most η, by unitarity. The triangle inequality gives 2η for the entire circuit, and reversing the preparation preserves this bound. □

We now apply the lemma to the recursion of Section 2.1. At an outer split, the two retained operands have ℓ bits each and the phase denominator is $2^{2\ell}$. Inside an m-qubit leaf, the shifts, checks,

and uncomputation in the decoder use reversible Boolean logic, Hadamards, and product phases on operands of at most m bits [3, Section 3]. Within W and $W^\dagger$, expanding the exact Cleve–Watrous transforms [1] gives product phases at the widths of the corresponding splits. Thus every such denominator has exponent at most $2m \le 4\ell$. In the elementary QFT at $m < 64$, each controlled phase has denominator at most 2^m and uses a constant addition conditioned on both data bits. The remaining phases are the rounded flag rotations and their inverses, with denominator 2^ℓ. A register of width $B = 4\ell + 2$ therefore suffices for the whole tree. We keep this register and the adder scratch separate from the decoder workspace, so each reflection tests exactly the qubits prescribed by the decoder.

For a product on t-bit operands, Lemma 2.1 charges $O(M(t) + t)$ gates at that node, regardless of B. This is the cost already counted in (4). Hadamards and elementary reversible Boolean gates have constant Clifford+T cost, and a zero test uses linearly many such gates in its register width. A leaf also uses at most six rounded flag rotations [3, Section 4.2], each costing $O(\ell)$ gates. For $q > 2\ell$ there are $O(q/\ell)$ leaves, so these rotations add only $O(q)$ gates. For $q \le 2\ell$ the single leaf costs $O(q(\log q)^\alpha + \ell)$. In both cases the ideal gradient state gives a gate count of $O(q(\log \ell)^\alpha)$ in the accuracy regimes of Proposition 2.

We prepare the gradient register once and reverse its preparation after the last use. Its j-th most significant qubit is $P(-2\pi/2^j)H|0\rangle$. Set $\eta = 2^{-d}/4$ and synthesize each of these B single-qubit operations to accuracy η/B up to a global phase. Their matrix entries are computable to s bits in time polynomial in $B + s$. The finite-precision Solovay–Kitaev theorem [13, Theorem 4.3] thus gives a deterministic synthesis over H, S, T and their inverses. The tensor product has vector error at most η, and preparation and uncomputation together cost $O(B \operatorname{polylog}(B/\eta))$ gates.

At fixed or inverse-polynomial error, $d = O(\log q)$ and $B = O(\log q)$, so this one-time cost is $\operatorname{poly}(\log q) = o(q)$. Adding the 2η preparation bound to the truncation and rounding errors of Section 2.1 gives $2\eta + 16\pi q 2^{-\ell} \le 2^{-d} \le \epsilon$. Since $\log \ell = \log\log q + O(1)$, the total Clifford+$T$ gate count is $O(q(\log\log q)^\alpha)$. The uniform multiplier, the angle computation in Section 2.2, and the deterministic synthesis above generate the circuit in polynomial time in these regimes. Theorem 4 supplies a single angle algorithm for unrestricted inputs $(1^q, 1^d)$. This proves Proposition 2.

3 Boolean circuits from fast integer multiplication

We verify that OpenAI's multiplier satisfies the hypothesis of Proposition 2, so that we can use it in our Clifford+T implementation of the QFT construction of Kayal et al. [3]. OpenAI proves a time bound of $T(n) = O_\delta(n(\log n)^{1/2+\delta})$ on a multitape Turing machine [4, Theorem 5.1]. The general simulation of Pippenger and Fischer [14] gives a Boolean circuit of size $O(T(n) \log T(n))$, adding a factor of $\log n$ to this upper bound. To retain the exponent $1/2 + \delta$ in a polynomial-time uniform bounded-fan-in circuit, we use the multiplier's fixed record permutations as wiring and count the Boolean gates for its word operations.

Proposition 3. *For every fixed $\delta > 0$, the integer multiplication algorithm of [4] yields a polynomial-time uniform bounded-fan-in Boolean circuit of size $O_\delta(n(\log n)^{1/2+\delta})$ for the full product of two n-bit integers.*

Proof. We use the array layout of OpenAI's algorithm to count word operations and its precision bounds to count Boolean gates per operation. The algorithm forms the convolution of the input digit arrays using two forward transforms, pointwise multiplication, and one inverse transform [4, Section 5]. It uses $N = 2^k$ records with $k = \Theta(\log n)$ and $Nk = \Theta(n)$. Each transform has $O(\log(2 + k))$ layers. A layer permutes the records, multiplies each record by a fixed coefficient, or applies a normalized Walsh transform on $s = O(k/\log k)$ selected address bits [4, Lemma 5.2].

Boolean gate count. Choose rational parameters $1/2 < \rho < 1$ and $\eta > 0$ with $\rho + \eta < 1/2 + \delta$. At the precision prescribed in [4, Lemma 4.1 and Section 5.4], each fixed-point word has $O_\rho(k)$ bits. For fixed n, the record permutations are just wiring, and the circuit generator can precompute the fixed coefficients. Addition and rounding cost $O_\rho(k)$ gates per word. A fixed-arity Toom–Cook multiplier uses $O_{\rho,\eta}(k^{1+\eta})$ gates per word product [16, Section 1.3.3]. Indeed, for arity r with $\log_r(2r-1) < 1+\eta$, the $2r-1$ recursive products have width $k/r + O_r(1)$, and evaluation, interpolation, and division by fixed constants cost $O_r(k)$ gates. A complex product takes a constant number of these word operations.

It remains to count the word operations in a Walsh layer. OpenAI's normalized Walsh recursion uses fixed integers A, W, σ satisfying $\sigma/W < A^\rho$ [15, Proposition 5.2 and Eq. (5.3)]. On a batch of W rows with $a = 2A^j$ active bits per row ($j \geq 1$), it makes σ calls of width a/A. Each call processes $1/W$ of the parent records, and the remaining work consists of permutations and a fixed number of arithmetic operations per record [4, Lemma 4.1]. Thus the Boolean gate cost $C(a)$ per record satisfies

$$C(2) = O(k^{1+\eta}), \qquad C(a) \leq O(k^{1+\eta}) + (\sigma/W)C(a/A).$$

Since $\sigma/(WA^\rho) < 1$, this gives $C(a) = O(k^{1+\eta}a^\rho)$. For arbitrary s, the same construction handles an odd bit with one butterfly layer and partitions the other bits into widths $2A^j$ using the base-A digits of $\lfloor s/2 \rfloor$. At most $A-1$ blocks have any given width, so their ρ-th powers sum to $O((1+s)^\rho)$. Padding the batches at most doubles the record count [4, Lemma 4.1], so the entire layer costs $O(Nk^{1+\eta}(1+s)^\rho)$ gates.

Exact output and uniformity. Our word circuits implement the same fixed-point operations, with the same coefficients and rounding rules, as the tape algorithm. OpenAI's error bound of less than $1/2$ for each valid convolution coefficient therefore applies to our circuit [4, Section 5.4]. Rounding recovers every integer coefficient exactly. Propagating the $O(k)$-bit carries then gives the full $2n$-bit product in $O(Nk)$ additional gates.

Summing over the three transforms and the pointwise arithmetic gives

$$O(Nk^{1+\rho+\eta}\log(2+k)) = O_\delta(n(\log n)^{1/2+\delta}),$$

where $Nk = \Theta(n)$ and the strict inequality $\rho + \eta < 1/2 + \delta$ absorbs the factor $\log(2+k)$. A direct multiplier covers bounded input lengths. For fixed ρ, the Walsh recursion is effective, and the permutations and coefficients at the required precision are computable from n in polynomial time [4, Lemmas 4.1 and 5.2]. On input 1^n, the generator can therefore enumerate the records and emit their word circuits in polynomial time. □

Substituting $\alpha = 1/2 + \delta$ into Proposition 2 gives the total Clifford+T gate bound of Theorem 1.

4 Uniformity of the exact and dyadic circuits

The QFT construction of Kayal et al. uses a rotation $B(\theta_q)$ whose angle depends on the register width q. Without an efficient classical algorithm for computing this angle, they could not establish that their circuit families are uniform [3]. Assuming a polynomial-time uniform multiplier, they ask whether θ_q can be computed classically to additive error 2^{-d} in time polynomial in q and d [3, Section 5]. In this section we give a deterministic algorithm with this running time. It computes the mean success probability p_q of the approximate Fourier decoder by dynamic programming and then obtains θ_q from (5).

Section 2.2 computes this average for an m-qubit leaf by enumerating all 2^{2m} pairs of input and output labels. At fixed or inverse-polynomial error, each leaf of the approximate q-qubit QFT has $m = O(\log q)$, so the enumeration takes polynomial time in q. For the rotation in the exact q-qubit circuit, the same method would enumerate 2^{2q} pairs even at fixed precision.

Theorem 4. *For the balanced truncated Fourier transform W of Section 2.1 on q bits, the quantities p_q and θ_q in (5) can be computed to additive error 2^{-d} in deterministic time polynomial in q and d.*

For a fixed input, we can sum over the output bits exactly to express the squared overlap in (5) as a product of q cosine-squared factors. Our dynamic program averages that product over the input bits. At each position the phase retained by W depends only on the last $2k = O(\log q)$ input bits. The exact Fourier phase also depends on earlier bits through a single binary fraction whose contribution halves with each new bit. This decay lets us approximate the average over the remaining bits, for each suffix, by a polynomial of degree $O(q + d)$ in that fraction. Working backwards from the last bit, we combine the two possible continuations for each suffix and update only $O(q^2)$ polynomials per position.

Proof. Number the input bits $x_1, \ldots, x_q$ from least to most significant, and assign output bit y_j the weight 2^{q-j}. Setting $z_0 = 0$ and $z_j = (z_{j-1} + x_j)/2$, we have

$$\langle y|F_q|x\rangle = 2^{-q/2} \exp\left(2\pi i \sum_{j=1}^{q} y_j z_j\right).$$

For the truncated transform W, each $z_j = \sum_{i\le j} 2^{-(j-i+1)} x_i$ is replaced by the sum a_j of the retained terms. Indeed, writing $x = u + 2^r v$ and $y = b + 2^s a$ at a split into $r + s$ bits gives $xy/2^{r+s} \equiv ua/2^r + vb/2^s + ub/2^{r+s} \pmod 1$, where the first two terms belong to the children and the last couples the untouched input with the transformed child's output. Retaining the k leading bits of u and b deletes precisely the omitted bilinear terms: at a split after bit h, the surviving cross-terms involve inputs $h - k + 1, \ldots, h$ and outputs $h + 1, \ldots, h + k$. Since exact leaves have width at most $2k$, each a_j depends only on j and the suffix of $w = 2k$ input bits ending at x_j, with its dyadic value determined by the split tree. Summing over the independent output bits gives

$$\left|\langle x|W^\dagger F_q|x\rangle\right|^2 = \prod_{j=1}^{q} \cos^2(\pi(z_j - a_j)). \tag{7}$$

After j input bits, let u denote the suffix of the last w bits (initially padded with zeros). Upon appending a bit $b \in \{0, 1\}$, write u_b for the updated suffix and $t_b = (z + b)/2$. Define $f_q \equiv 1$ and, for $0 \le j < q$,

$$f_j(u, z) = \frac{1}{2} \sum_{b=0}^{1} \cos^2(\pi(t_b - a_{j+1}(u_b)))\, f_{j+1}(u_b, t_b).$$

Then $p_q = f_0(0, 0)$, and there are $2^w = O(q^2)$ suffixes at each stage. To handle the continuous variable z, fix any future bit string: after h additional steps, the argument of its $\cos^2$ factor has coefficient 2^{-h} on z (before multiplying by π), so the factor is a sum of exponentials $e^{2\pi i\nu z}$ with frequencies $\nu \in \{0, \pm 2^{-h}\}$ whose coefficient magnitudes sum to 1. Multiplication adds frequencies, and averaging cannot increase the ℓ_1 norm of the coefficients. Hence every $f_j(u, \cdot)$ admits an exponential expansion with $|\nu| \le \sum_{h\ge1} 2^{-h} = 1$ and total coefficient magnitude at most 1.

Equip power series in z with the ℓ_1 norm of their coefficients. The tail of the exact series for f_j beyond degree D has norm at most $\tau_D = \sum_{n>D} (2\pi)^n/n!$. Whenever $D + 1 \ge 4\pi e$, successive terms decay by at least a factor of $1/2$, and $n! \ge (n/e)^n$ yields

$$\tau_D \le 2\left(\frac{2\pi e}{D+1}\right)^{D+1} \le 2^{-D}.$$

In each dynamic-programming step, we substitute $(z + b)/2$ into the stored degree-D polynomial, multiply by the $\cos^2$ factor, and truncate to degree D. Substitution has operator norm at most 1 in the coefficient ℓ_1 norm. The $\cos^2$ multiplier is a linear combination of $1, e^{\pi iz}, e^{-\pi iz}$ with coefficient magnitudes $1/2, 1/4, 1/4$, so its power series has norm at most $C = (1 + e^\pi)/2 < 16$. Letting E_j be

the maximum over all suffixes u of the ℓ_1 distance between the computed polynomial and the exact series $f_j(u,\cdot)$, and noting that truncation to degree D has norm 1, we obtain

$$E_q = 0, \qquad E_j \le CE_{j+1} + \tau_D + \lambda,$$

where λ bounds the coefficient-norm rounding error per update. Choosing

$$s = d + 4q + \lceil \log q \rceil + 4, \qquad D = \max\{35, s\}, \qquad \lambda = 2^{-s}$$

gives $E_j \le qC^q(\tau_D + \lambda) \le 2^{-d-3}$ for all j.

In binary arithmetic, the substitution matrix has exact entries $2^{-n}\binom{n}{h}b^{n-h}$ representable in $O(D)$ bits, and the degree-D coefficients of the multiplier arise from $(\pm\pi i)^n/n!$ times a unit phase at a known dyadic multiple of π, which can be evaluated to arbitrary precision in polynomial time via Taylor summation. Direct substitution and convolution use $O((D+1)^2)$ multiply-adds per suffix. Because the exact series has norm at most $e^{2\pi}$ by its frequency bound, whenever $E_{j+1} \le 1$ the input polynomial has norm at most $e^{2\pi} + 1$. Rounding each multiply-add and each multiplier coefficient to absolute error u therefore incurs a local error of at most $C_0(D+1)^2u$ for an absolute constant C_0 (since each substitution rounding is amplified by at most C, each convolution rounding enters once, and coefficient perturbations have total norm at most $(D+1)u$). Taking $u \le \lambda/[C_0(D+1)^2]$ closes the induction from $j = q$ down to 0. The algorithm runs over q stages with $O(q^2)$ suffixes per stage and $O(D + s + \log(D+1))$ bits per working number. Evaluating at $z = 0$ yields p_q to within 2^{-d-3}. Projecting onto $[3/4, 1]$ preserves this accuracy, and since

$$\left| \frac{d}{dp} \arcsin \frac{1}{2\sqrt{p}} \right| = \frac{1}{2p\sqrt{4p-1}} \le \frac{\sqrt{2}}{3} \qquad \left(\frac{3}{4} \le p \le 1 \right),$$

evaluating the arcsine to absolute error 2^{-d-2} determines θ_q to within 2^{-d}. □

We remark that Mosca and Zalka evaluated the mean success probability of their Fourier decoder using power series and power sums [17, Section 4.1], and that an alternative polynomial-time algorithm for (5) can be obtained via matrix product operators (MPOs). Specifically, Chen and Lindsey [18, Theorem 4] approximate the unnormalized bit-reversed Fourier matrix with bond dimension $K+1$ and maximum entrywise error $(q-1)\Lambda_K^{q-2}E_K$, where $\Lambda_K = O(\log(K+1))$ and $E_K \le (C_1/K)^K$ for an absolute constant C_1 and sufficiently large K. Conjugating their cores for the positive Fourier sign and setting $K = \lceil C_2(q+d+1)^2 \rceil$ for a sufficiently large constant C_2 bounds the entrywise error by $\eta = 2^{-d-4}$. In the same bit order, the unnormalized W admits an exact MPO of bond dimension 2^w, whose transitions append x_j to the suffix and emit $e^{2\pi i y_j a_j}$. Writing $N = 2^q$, $F^\circ = \sqrt{N}F_q$, and $W^\circ = \sqrt{N}W$, we have

$$p_q = N^{-3} \sum_{x,y,z} F^\circ_{yx}\, \overline{W^\circ_{yx}}\, \overline{F^\circ_{zx}}\, W^\circ_{zx}.$$

Since each normalized column of F_q is perturbed by at most η in Euclidean norm, the squared overlaps change by at most $2\eta+\eta^2$. Contracting the four MPOs site by site keeps the bond dimension polynomial; each transfer matrix has polynomially many entries per row, each of magnitude $\mathrm{poly}(q+d+2)$ (as each Fourier core entry is bounded by $1 + \Lambda_K$). Over q sites, rounding errors amplify by at most $(q+d+2)^{O(q)}$, so $d + O(q\log(q+d+2))$ fractional bits suffice. Finally, the core entries are built from phases and Lagrange cardinal polynomials on a Chebyshev–Lobatto grid whose distinct nodes are separated by $\Omega(K^{-2})$, so each cardinal polynomial can be evaluated to the required precision with $O(K\log(K+1))$ additional guard bits.

Corollary 5. *Under the multiplication hypothesis of Proposition 2, the exact unitary circuit of Kayal et al. [3] has a polynomial-time uniform description and uses $O(q(\log q)^\alpha)$ gates, counting dyadic phases and the exact rotation $B(\theta_q)$ at unit cost. For any $0 < \epsilon < 1$, their approximate circuit admits a uniform implementation over Hadamard, CNOT, and dyadic phases using $O(q(\log L)^\alpha)$*

gates and $O(L(\log L)^\alpha)$ auxiliary qubits, where $L = \log(q/\epsilon)$. By Proposition 3, both bounds hold with $\alpha = 1/2 + \delta$.

Proof. By Theorem 4, the rotation $B(\theta_q)$ is specified by a polynomial-time classical routine evaluating θ_q to arbitrary binary precision, and acts at unit cost in the exact model. The constant-error circuit W uses $O(q(\log\log q)^{\alpha+1})$ gates: its $O(q/k)$ leaves of width $O(k)$ use the exact Cleve–Watrous recursion at cost $O(k(\log k)^{\alpha+1})$ each, while its truncated product phases cost $O(M(k)+k)$ each. Applying the leaf decoder of Section 2.1 at width q gives clean circuits for F_q and $F_q^\dagger$ with $O(M(q)+q)$ gates beyond the constant number of calls to W and $W^\dagger$. Since $M(q) = O(q(\log q)^\alpha)$ and $(\log\log q)^{\alpha+1} = o((\log q)^\alpha)$, the total is $O(q(\log q)^\alpha)$ gates.

For the approximate circuit, we take ℓ as in Section 2 and compute each leaf angle via Theorem 4. The full-workspace error bound in Section 2.1 allows sequential calls to reuse the workspace of a single leaf or multiplication. Together with (4), the bounds in that section establish the claim for $q > 2\ell$. For $q \le 2\ell$, a single rounded leaf costs $O(q(\log q)^\alpha)$ and satisfies both bounds. The classical generator runs in time polynomial in q and d. □

5 Exact Fourier transformation with intermediate measurements

Section 4 computes the angle of the special rotation $B(\theta_q)$ in the exact QFT circuit of Kayal et al. [3]. With intermediate measurements and classical feedforward, we can bypass this rotation and implement F_q exactly using one product phase. We count each elementary gate and each binary measurement at unit cost. The gate set includes exact dyadic phases whose angles can be chosen in polynomial time from earlier measurement outcomes.

Proposition 6. *A uniform Boolean multiplier of size $M(q)$ yields a uniform exact adaptive circuit for F_q using $O(M(q)+q)$ quantum gates and measurements on every branch. In particular, the multiplier of Proposition 3 gives $O_\delta(q(\log q)^{1/2+\delta})$ operations in this model.*

Proof. We use Hall's one-dit teleportation identity [20, Fig. 1], representing each N-dimensional system by q qubits, where $N = 2^q$. Write $\omega = e^{2\pi i/N}$ and let $|\psi\rangle = \sum_x a_x|x\rangle$ be the input. Prepare a second register in the uniform superposition with q Hadamards. A single product phase (3) between the registers then gives

$$\frac{1}{\sqrt{N}}\sum_{x,y} a_x\omega^{xy}|x\rangle|y\rangle = \sum_x a_x|x\rangle|\phi_x\rangle, \qquad |\phi_x\rangle = F_q|x\rangle.$$

For each input label x, this one operation supplies the phases for every output label y. Computing xy, applying dyadic phases to its low q bits, and uncomputing the product costs $O(M(q)+q)$ gates [3, Lemma 2.1].

To erase the input label while preserving the superposition, measure the first register in the Fourier basis using the semiclassical procedure of Griffiths and Niu [19]. For each bit, the earlier outcomes determine a single dyadic phase, followed by a Hadamard and a binary measurement. This uses $O(q)$ single-qubit gates and q measurements on every branch. For outcome j, the overlap $\langle\phi_j|x\rangle = N^{-1/2}\omega^{-jx}$ leaves the second register in the unnormalized state $N^{-1/2}\sum_x a_x\omega^{-jx}|\phi_x\rangle$.

Subtracting j modulo N from the second register multiplies $|\phi_x\rangle$ by ω^{jx} and cancels the measurement phase. The corrected branch operator is therefore $N^{-1/2}F_q$, so every outcome gives the exact transform even for an input entangled with a reference. The subtraction of the known j costs $O(q)$ gates [12]. Preparation, measurement, and correction thus add only $O(q)$ operations to the single product phase on each branch, giving the claimed $O(M(q)+q)$ total. The circuit is uniform because the multiplier has a polynomial-time generator and the measurement angles and correction are computable from the outcome bits in polynomial time. □

Open questions. Can an exact unitary circuit over Hadamard, CNOT, and dyadic phases alone match the $O(M(q) + q)$ size bound of Proposition 6? At fixed accuracy, Theorem 1 leaves open whether the quantum Fourier transform admits polynomial-time uniform unitary Clifford+T circuits of linear total size $O(q)$.

Disclosure of AI usage

GPT-6.2 Astra carried out most of the calculations and developed most of the proofs in this paper. Zhou Kangyun proposed applying OpenAI's new integer multiplication algorithm to the quantum Fourier transform and reviewed the manuscript for clarity and mathematical correctness.

References

[1] R. Cleve and J. Watrous. Fast parallel circuits for the quantum Fourier transform. In *Proc. 41st IEEE Symposium on Foundations of Computer Science*, pages 526–536, 2000. arXiv:quant-ph/0006004.

[2] G. Ahokas, R. Cleve, and L. Hales. The complexity of quantum Fourier transforms and integer multiplication. ERATO conference abstract, 2003. See G. Ahokas, *Improved algorithms for approximate quantum Fourier transforms and sparse Hamiltonian simulations*, M.Sc. thesis, University of Calgary, 2004, Section 3.1.

[3] C. Kayal, R. Mittal, M. Paraashar, and N. Saurabh. Improved exact and approximate quantum Fourier transforms from multiplication, 2026. arXiv:2611.06341v1.

[4] OpenAI. The logarithmic exponent of integer multiplication on tapes. November 24, 2026.

[5] D. Harvey and J. van der Hoeven. Integer multiplication in time $O(n \log n)$. *Annals of Mathematics*, 193:563–617, 2021. doi:10.4007/annals.2021.193.2.4.

[6] Y. Nam, Y. Su, and D. Maslov. Approximate quantum Fourier transform with $O(n \log n)$ T gates. *npj Quantum Information*, 6:26, 2020. arXiv:1803.04933.

[7] B. Park and D. Ahn. Reducing T-count and T-depth in approximate quantum Fourier transform circuits. *Scientific Reports*, 15:37199, 2025. doi:10.1038/s41598-025-21087-2.

[8] C. Zhu, Y.-A. Chen, Z. Liu, Z. Yu, Z. Shen, and X. Wang. Amortized stabilizer Rényi entropy of quantum dynamics, 2026. arXiv:2409.06659v2.

[9] P. Selinger. Quantum circuits of T-depth one. *Physical Review A*, 87:042302, 2013. doi:10.1103/PhysRevA.87.042302.

[10] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London A*, 454:339–354, 1998. arXiv:quant-ph/9708016.

[11] A. Wu, L. Lu, P. Pang, J. Yin, J. Leng, and M. Guo. Shared phase arithmetic for parallel quantum rotations, 2026. arXiv:2609.36574.

[12] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton. A new quantum ripple-carry addition circuit, 2004. arXiv:quant-ph/0410184.

[13] D. van Melkebeek and T. Watson. Time-space efficient simulations of quantum computations. *Theory of Computing*, 8:1–51, 2012.

[14] N. Pippenger and M. J. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, 1979. doi:10.1145/322123.322138.

[15] OpenAI. The logarithmic exponent of finite Fourier transforms. November 24, 2026.

[16] R. P. Brent and P. Zimmermann. *Modern Computer Arithmetic*. Cambridge University Press, 2010, Section 1.3.3.

[17] M. Mosca and C. Zalka. Exact quantum Fourier transforms and discrete logarithm algorithms, 2003. arXiv:quant-ph/0301093.

[18] J. Chen and M. Lindsey. Direct interpolative construction of the discrete Fourier transform as a matrix product operator, 2024. arXiv:2404.03182.

[19] R. B. Griffiths and C.-S. Niu. Semiclassical Fourier transform for quantum computation. *Physical Review Letters*, 76:3228–3231, 1996. arXiv:quant-ph/9511007.

[20] W. Hall. Cluster state quantum computation for many-level systems. *Quantum Information and Computation*, 7:184–208, 2007. arXiv:quant-ph/0512130.