Quantifying the Resource Cost of Shor's factoring algorithm

Guangqi Gao, Jerry Xia, Avery Adam Books, Yiqun Zhao

December 2025

1 Introduction

Shor's factoring algorithm is one of the most celebrated examples of a quantum algorithm that achieves an exponential speedup over the best known classical methods. In the standard complexity-theoretic description, Shor's algorithm factorizes an n-bit integer N in time polynomial in n, in stark contrast to the sub-exponential but super-polynomial classical algorithms such as the number field sieve. This asymptotic separation has often been used to argue that large-scale quantum computers will decisively outperform classical machines on cryptographically relevant instances.

At the same time, such complexity-theoretic statements are typically formulated in an idealized gate model that does not explicitly track the intermediate resource overheads. A similar issue appears already in Grover's search algorithm: the textbook $O(\sqrt{N})$ speedup assumes that each application of the oracle U_f is a unit-cost operation, whereas in any concrete implementation U_f itself must be compiled into a sequence of elementary one- and two-qubit gates. In the case of Shor's algorithm, the corresponding "oracle" is modular exponentiation, and its implementation cost is usually hidden inside the statement that order finding can be done in "polynomial time."

2 Objectives of This Work

In this project, we focus on making these costs explicit at the level of asymptotic circuit complexity. Rather than attempting a full architecture-specific resource estimate with hardware-dependent constants, we analytically decompose the modular exponentiation subroutine into elementary building blocks (Fourier adders, modular adders, modular multipliers, and controlled modular exponentials) and track how the gate counts, circuit depth, and logical qubit requirements scale with the input size $n = \log_2 N$. We work within a realistic but simplified gate model, using a hardware-motivated universal gate set such as $\{R_Z, SX, X, \mathrm{CX}\}$, and we study how standard optimizations—most notably the use of the approximate Quantum Fourier Transform (AQFT)—affect the overall asymptotic resources.

To connect these logical-level complexities to fault-tolerant execution, we also incorporate a coarse-grained surface-code model to estimate how many physical qubits are required as a function of the number of logical qubits and non-Clifford operations. Our treatment is intentionally high-level: we do not tune architecture-specific constants, schedules, or decoder details, but instead aim to capture the dominant scaling behavior and its dependence on algorithmic design choices. In this way, our analysis sits between purely abstract big-O statements and fully engineered resource estimates, and is intended to clarify which parts of Shor's algorithm dominate the space–time complexity once realistic gate decompositions and approximate QFT constructions are taken into account.

3 Universal Gate Sets

Before we analyze the resource requirements of the arithmetic subroutines used in Shor's algorithm, it is essential to clarify the computational model in which these resources are counted. In particular, any

assessment of circuit complexity must specify the elementary operations from which larger unitaries are constructed. Although Shor's algorithm is often described in terms of high-level unitaries such as modular addition, modular multiplication, and the Quantum Fourier Transform, an actual quantum device can only implement a fixed, finite set of primitive one- and two-qubit gates. All higher-level operations must therefore be decomposed into sequences of these elementary gates.

For this reason, prior to analyzing the cost of Shor's algorithm itself, we first review the theoretical foundations of universality: why any unitary transformation on n qubits can be synthesized from a small universal gate set, and how the choice of such a set influences circuit depth and gate count. This discussion provides the framework needed for the resource analysis that follows, and allows us to express the cost of the algorithm in terms of hardware-relevant primitive operations.

3.1 Mathematical Reasoning for Universality

The concept of universality in quantum computing refers to the ability of a finite set of quantum gates to approximate any unitary transformation on an arbitrary number of qubits to arbitrary precision. This section outlines the mathematical basis for universality through three pillars: Lie algebraic controllability, the Solovay–Kitaev theorem, and the construction in Barenco et al. (1995) that shows any unitary operation in $U(2^n)$ can be composed from arbitrary single-qubit gates and a fixed entangling two-qubit gate.

Lie Algebra and Controllability

In continuous quantum control theory, a quantum system is said to be *controllable* if one can generate any unitary operator on the system's Hilbert space through time evolution under a set of available Hamiltonians. Formally, consider a set of Hamiltonians $\{H_k\}$. The unitaries that can be generated by evolving under these Hamiltonians (and their nested commutators) are of the form:

$$U(t) = \mathcal{T} \exp\left(-i \int_0^t H(t')dt'\right)$$

If the Lie algebra generated by $\{H_k\}$ via commutators spans all of $\mathfrak{su}(2^n)$, then the system is controllable in $SU(2^n)$. This implies that the group generated by exponentiating these Hamiltonians is dense in $SU(2^n)$, and hence any unitary in this group can be approximated arbitrarily well by sequences of gates derived from $\{H_k\}$.

For example, the Pauli operators $\{X, Y, Z\}$, multiplied by real scalars, generate the Lie algebra $\mathfrak{su}(2)$. Therefore, combinations of $R_x(\theta), R_y(\theta), R_z(\theta)$ can generate arbitrary $SU(2)$ operations on a single qubit. When extended to two-qubit systems, if the Lie closure of local gates and an entangling gate covers $\mathfrak{su}(4)$, the system is universal.

Solovay–Kitaev Theorem

The Solovay–Kitaev theorem provides a constructive guarantee that finite, discrete gate sets can still achieve universal quantum computation. Let $\mathcal{G} \subset SU(2)$ be a finite gate set that generates a dense subgroup of $SU(2)$, and suppose $\mathcal{G}$ is closed under taking inverses. Then for any unitary $U \in SU(2)$ and any accuracy $\epsilon > 0$, there exists a sequence of gates from $\mathcal{G}$ whose product approximates U within operator norm ϵ, and the sequence length scales as:

$$\mathcal{O}(\log^c(1/\epsilon)), \quad \text{for some } c < 4$$

This result is significant because it shows that even when using a finite universal gate set (e.g., Clifford+T), one can efficiently approximate any quantum circuit. The theorem bridges the gap between the continuous structure of $SU(2^n)$ and practical quantum compilation using discrete gates.

Arbitrary 1-Qubit Gates + One Entangling 2-Qubit Gate = Universality

Barenco et al. (1995) [1]proved a foundational result: any set containing all single-qubit unitaries $U(2)$ and any entangling two-qubit gate (such as CNOT) is universal—that is, it can approximate any unitary operation in $U(2^n)$ to arbitrary precision.

This matters because it provides a general recipe for building universal gate sets:

$$\text{Full 1-qubit control} + \text{Any 2-qubit entangling gate} \Rightarrow \text{Universality}$$

From this principle, many practical universal sets follow by restricting the single-qubit gates to a finite (but dense-generating) subset. For example:

- $\{H, T, \text{CNOT}\}$ — the standard Clifford+T set used in fault-tolerant quantum computing
- $\{R_Z(\theta), R_X(\theta), \text{CNOT}\}$ — common in analog quantum control
- $\{U_3, \text{CX}\}$ — the default universal basis for IBM Quantum systems
- $\{R_Z, SX, X, \text{CX}\}$ — minimal and hardware-native set also known to be universal

The power of this result lies in its generality: once arbitrary single-qubit operations are accessible (even approximately), and one entangling gate is included, the full space of quantum computation becomes reachable. This shifts the question of universality from a long list of requirements to a compact structural condition—enabling both theoretical classification and practical design of universal gate sets.

In contrast to classical reversible logic, where $\{\text{XOR}, \text{NOT}\}$ is not universal, the inclusion of quantum amplitudes and continuous single-qubit rotations enables quantum systems to access the full group structure of $SU(2^n)$, revealing the distinctive power of quantum computation.

3.2 Common Universal Gate Sets and Their Properties

While the mathematical definition of universality allows for many possible quantum gate sets, practical implementations tend to cluster around a few well-studied families. In this section, we introduce two major types of universal gate sets—discrete and continuous—and explain both their theoretical basis and practical implications. We also highlight a fundamental result: that all universal gate sets are equivalent in computational power, meaning that the complexity class BQP is independent of gate set choice.

Clifford + T Gate Set (Discrete, Fault-Tolerant-Oriented)

The Clifford+T gate set, consisting of $\{H, S, \text{CNOT}, T\}$, is among the most studied in quantum computation. Clifford gates alone form a group that stabilizes Pauli operators and admits efficient classical simulation (Gottesman–Knill theorem), but they are not computationally universal. The addition of the T gate—a $\pi/8$ phase rotation—breaks this restriction and enables dense approximation of any single-qubit unitary in $SU(2)$. This makes the set:

$$\{H, T, \text{CNOT}\}$$

a minimal, discrete universal gate set.

This gate set plays a central role in the theory of fault-tolerant quantum computing. Clifford gates can typically be implemented transversally in quantum error-correcting codes, while T gates are introduced through costly, but fault-tolerant, techniques such as magic state distillation. The trade-off is efficiency: small-angle single-qubit rotations must be approximated with long sequences of H and T gates, increasing circuit depth and error accumulation. Nevertheless, its compatibility with quantum error correction makes Clifford+T the de facto standard for universal, fault-tolerant quantum circuits.

Arbitrary Single-Qubit Rotations + CNOT (Continuous, NISQ-Oriented)

A second major category of universal gate sets relies on analog-accessible, continuous single-qubit rotations combined with a fixed two-qubit entangling gate. Examples include:

$$\{R_z(\theta), R_x(\theta), \text{CNOT}\}, \quad \text{or} \quad \{U_3(\theta, \phi, \lambda), \text{CX}\}$$

where U_3 denotes a generic SU(2) single-qubit rotation, as used on IBM Quantum devices.

In these sets, single-qubit gates are not drawn from a discrete library but are parameterized by continuous angles. As such, any desired single-qubit operation can be applied exactly (subject to hardware precision). Together with a universal entangling gate like CNOT, these gates generate a dense subset of $SU(2^n)$, and are thus universal.

This model is common in near-term quantum devices where minimizing circuit depth is crucial. Unlike Clifford+T, these sets do not assume fault tolerance, but instead aim for practical performance on NISQ (Noisy Intermediate-Scale Quantum) processors. Their efficiency and simplicity make them well-suited for algorithm prototyping, benchmarking, and experimental implementation.

Gate-Set Independence of BQP

Although different universal gate sets may look structurally different, they are all equivalent in computational power. As emphasized in Preskill's lecture notes[2], any two universal gate sets can simulate each other with only a polylogarithmic overhead in circuit size. This conclusion is based on three observations:

1. A universal gate set allows arbitrary approximation of unitary operators.
2. The Solovay–Kitaev theorem guarantees that any such approximation can be achieved with circuit depth scaling as $\mathcal{O}(\log^c(1/\epsilon))$, for $c < 4$.
3. Therefore, a circuit built from one universal gate set G' can be simulated by a circuit over another set G with only polylogarithmic expansion in gate count.

This implies that the complexity class BQP (Bounded-Error Quantum Polynomial Time)—which captures the power of quantum computation—is independent of the chosen universal gate set. All such sets define the same class of efficiently solvable quantum problems.

This insight justifies flexibility in gate set selection. Whether one uses a discrete set like Clifford+T for theoretical decompositions or a continuous set like $\{R_Z, SX, X, \text{CX}\}$ for hardware efficiency, the essential computational capabilities remain unchanged.

3.3 Gate Set Selection for This Project

This project aims to analyze and implement quantum algorithms using gate sets that are both theoretically universal and practically realizable. To meet these goals, we use a gate set compatible with physical devices, while maintaining compatibility with fault-tolerant circuit design.

Practical Implementation Gate Set: $\{R_Z, SX, X, \mathbf{CX}\}$

For practical circuit construction and simulation, we adopt the gate set:

$$\{R_Z(\theta), SX, X, \text{CX}\}$$

This set is directly supported by superconducting quantum processors (e.g., IBM Quantum), making it highly suitable for implementation on near-term devices. Each gate is optimized for low noise and efficient calibration:

- $R_Z(\theta)$ is implemented virtually, incurring no gate time or decoherence.

- SX and X are single-qubit rotations about the X axis.
- CX is the standard two-qubit entangling operation.

These gates allow for the synthesis of arbitrary single-qubit unitaries via Z–X–Z Euler decomposition, and together with CX, form a universal gate set.

Toward Fault-Tolerant Universality: Clifford+T Compatibility

Although $\{R_Z, SX, X, \text{CX}\}$ is suitable for NISQ execution, it is not fault-tolerant by default. As quantum systems scale, it becomes necessary to consider gate sets compatible with quantum error correction (QEC). In this context, the Clifford+T set:

$$\{H, S, T, \text{CNOT}\}$$

is widely adopted as the standard for fault-tolerant quantum computation.

Clifford gates (H, S, CNOT) can be implemented transversally in many stabilizer codes, ensuring that errors do not spread across qubits. T gates, while not transversal in most codes, can be introduced fault-tolerantly using magic state distillation, a method for preparing high-fidelity logical T gates.

Importantly, our selected gate set supports conversion into this fault-tolerant model:

$$R_Z(\theta) \xrightarrow{\text{approx.}} \text{Clifford + T}$$

Any $R_Z(\theta)$ rotation can be approximated by a sequence of H and T gates using efficient synthesis algorithms. This ensures that circuits constructed in our gate set can be adapted to fault-tolerant execution.

We prioritize the gate set $\{R_Z, SX, X, \text{CX}\}$ for its experimental feasibility, low overhead, and decomposition efficiency. At the same time, we ensure compatibility with Clifford+T constructions, recognizing their role in error-corrected architectures. This layered approach allows us to bridge practical implementation with long-term scalability in quantum computing.

4 Implementation of Shor's algorithm

4.1 Introduction

Shor's algorithm provides an efficient quantum method for factoring an integer N. The algorithm combines classical number-theoretic preprocessing with a quantum subroutine for finding the order of an integer modulo N. The full factorization procedure can be summarized as follows:

1. If N is even, return 2.
2. Check whether $N = p^k$ is a perfect power for integers $p \geq 2$ and $k \geq 2$. If so, return p. It can be done in polynomial time.
3. Choose a random integer a with $1 < a < N$, and compute $\gcd(a, N)$. If the gcd is greater than 1, this produces a nontrivial factor.
4. Execute the quantum order-finding subroutine to compute the smallest integer $r > 0$ satisfying
$$a^r \equiv 1 \pmod{N}.$$
5. If r is odd or if $a^{r/2} \equiv -1 \pmod{N}$, return to step 3 with a new random a.
6. Otherwise compute
$$\gcd(a^{r/2} - 1, N) \quad \text{and} \quad \gcd(a^{r/2} + 1, N),$$
producing a nontrivial factor of N.

All steps are classical except for the order-finding subroutine. Thus, the quantum speedup arises entirely from efficiently computing the period of the modular exponentiation function.

4.2 Quantum Order Finding

Given a and N, the quantum subroutine computes the order r of a modulo N using quantum parallelism and the Quantum Fourier Transform (QFT). The function

$$f(x) = a^x \bmod N$$

is periodic with period r. The circuit operates on:

- A control register of $m \approx 2n$ qubits initialized by Hadamards into the superposition $\frac{1}{2^{m/2}} \sum_x |x\rangle$.
- A work register of n qubits initialized to $|1\rangle$, which means every qubit is in $|0\rangle$.

After controlled modular exponentiation:

$$|x\rangle |1\rangle \longmapsto |x\rangle |a^x \bmod N\rangle ,$$

phase information encoding the order r is extracted via the inverse QFT.

4.3 Structure of the Order-Finding Circuit

The circuit consists of the following components:

1. Superposition Initialization: Applying Hadamard gates to the control register produces a uniform superposition over all possible exponents.
2. Controlled Modular Exponentiation: For each control qubit j, apply the unitary

   $$U_{a^{2^j}} : |y\rangle \longmapsto |a^{2^j} y \bmod N\rangle ,$$

   controlled on that qubit. These gates are built using modular multipliers.
3. Inverse Quantum Fourier Transform: The inverse QFT converts the periodic modulation of the amplitudes into peaks at integer multiples of $2^m/r$. A semiclassical version using sequential measurements minimizes qubit requirements.

4.4 Implementation of the Quantum Fourier Transform

We begin with the factorized expression of the quantum Fourier transform acting on a computational basis state $|x\rangle$,

$$\mathrm{QFT}(|x\rangle) = \frac{1}{\sqrt{N}} \bigotimes_{j=1}^{n} \left(|0\rangle + \omega_N^{x 2^{n-j}} |1\rangle \right), \qquad N = 2^n.$$

This formula immediately reveals that the output of the QFT is a tensor product of n single-qubit states. Each output qubit therefore depends only on a specific subset of the binary digits of x, which allows us to construct the full QFT by treating the n output qubits one at a time. Moreover, each factor has the form $\big(|0\rangle + e^{i\phi}|1\rangle\big)$: the first term carries no phase, while the second carries a phase that encodes information about the input x. Crucially, the phase appearing on the j-th output qubit, $\omega_N^{x 2^{n-j}}$, can be expanded into a product of dyadic phase rotations determined by the individual bits of x. Because these phases depend on the higher-order bits of x, they can be implemented using controlled phase gates acting between qubits.

Thus, starting from the tensor-product structure above, the QFT naturally decomposes into a Hadamard gate generating the superposition on each qubit, followed by a sequence of controlled-R_k rotations that introduce the appropriate conditional phases. This observation provides the foundation for the standard gate-level QFT circuit which is shown in Figure 1, where

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{i2\pi/2^k} \end{pmatrix},$$

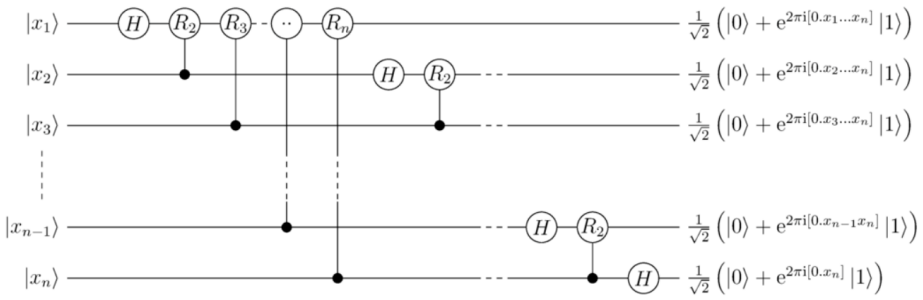

Figure 1: Quantum Fourier Transform circuit (source: adapted from Wikipedia, [3]).

To more accurately reflect the constraints of real quantum hardware, we further decomposed the controlled phase-rotation gates appearing in the QFT into elementary gates from a universal gate set. In practice, controlled rotations cannot be implemented directly and must be expressed in terms of hardware-native operations. In our study, we adopted the universal gate set {Rz, X,SX, CNOT} and used Qiskit's transpiler to decompose the QFT circuit accordingly. The resulting decomposition for the two-qubit QFT is shown in Fig. 2, which makes explicit how the ideal controlled-R_k operations are synthesized from the selected primitive gates. Moreover, by varying the number of qubits, we simulated QFT circuits of increasing size and recorded the number of Rz, SX, X and CNOT gates required after decomposition (Fig. 3). Our results indicate that both the CNOT count and the Rz gate count grow quadratically with the number of qubits, consistent with the theoretical $O(n^2)$ scaling of the QFT.

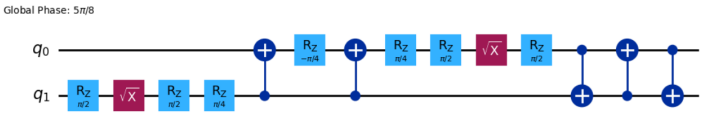

Figure 2: Quantum Fourier Transform circuit on 2 qubits

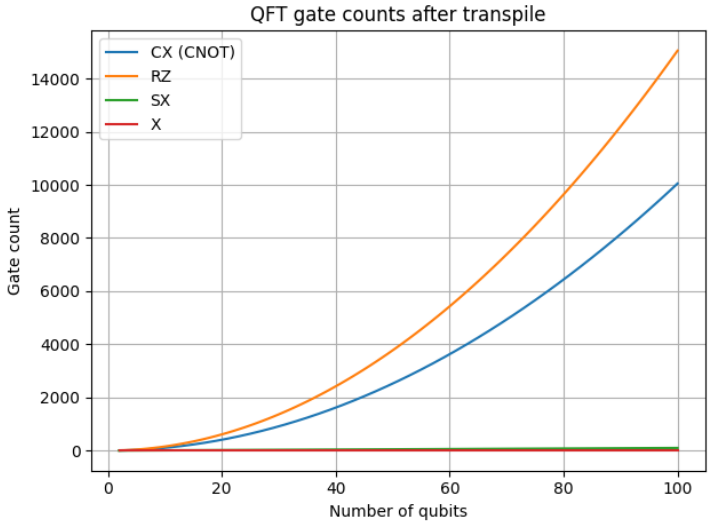

Figure 3: Gate count v.s. number of qubits for QFT circuit

The standard QFT on n qubits requires:

- n Hadamard gates,
- $\frac{n(n-1)}{2}$ controlled phase rotations $R_k = \mathrm{diag}(1, e^{2\pi i/2^k})$.

In total, the QFT uses $O(n^2)$ gates. However, its circuit depth is strictly smaller: many of the controlled-phase gates act on disjoint qubit pairs and can therefore be executed in parallel. As a result, the overall time (or depth) complexity of the QFT is $O(n)$, even though the total gate count is quadratic.

4.4.1 Approximate QFT

The exact Quantum Fourier Transform (QFT) on n qubits applies a Hadamard gate to each qubit together with controlled phase rotations

$$R_k = \mathrm{diag}(1, e^{2\pi i/2^k}),$$

for all $k = 2, 3, \ldots, n$. This requires

$$1 + 2 + \cdots + (n-1) = O(n^2)$$

controlled rotations, giving the standard quadratic gate complexity.

However, many of these rotations—particularly those with very small angles—contribute negligibly to the accuracy required for order finding. Omitting such rotations yields the *approximate* QFT (AQFT).

Why the AQFT Uses $O(n \log n)$ Gates. In the exact QFT, qubit j participates in controlled rotations with angles $2\pi/2^k$ for all

$$k = 1, 2, \ldots, n - j.$$

In the AQFT, we retain only those rotations satisfying

$$k \leq k_{\max},$$

with

$$k_{\max} = \Theta(\log(n/\varepsilon)).$$

This cutoff ensures that all omitted rotations have magnitude at most $2\pi/2^{k_{\max}}$, and Coppersmith's analysis shows that this yields total error $O(\varepsilon)$, well within the tolerance required in Shor's algorithm. Since each qubit now participates in only $O(\log n)$ controlled rotations, the total number of phase gates becomes

$$n \cdot O(\log n) = O(n \log n).$$

4.5 Implementation of Modular Exponentiation

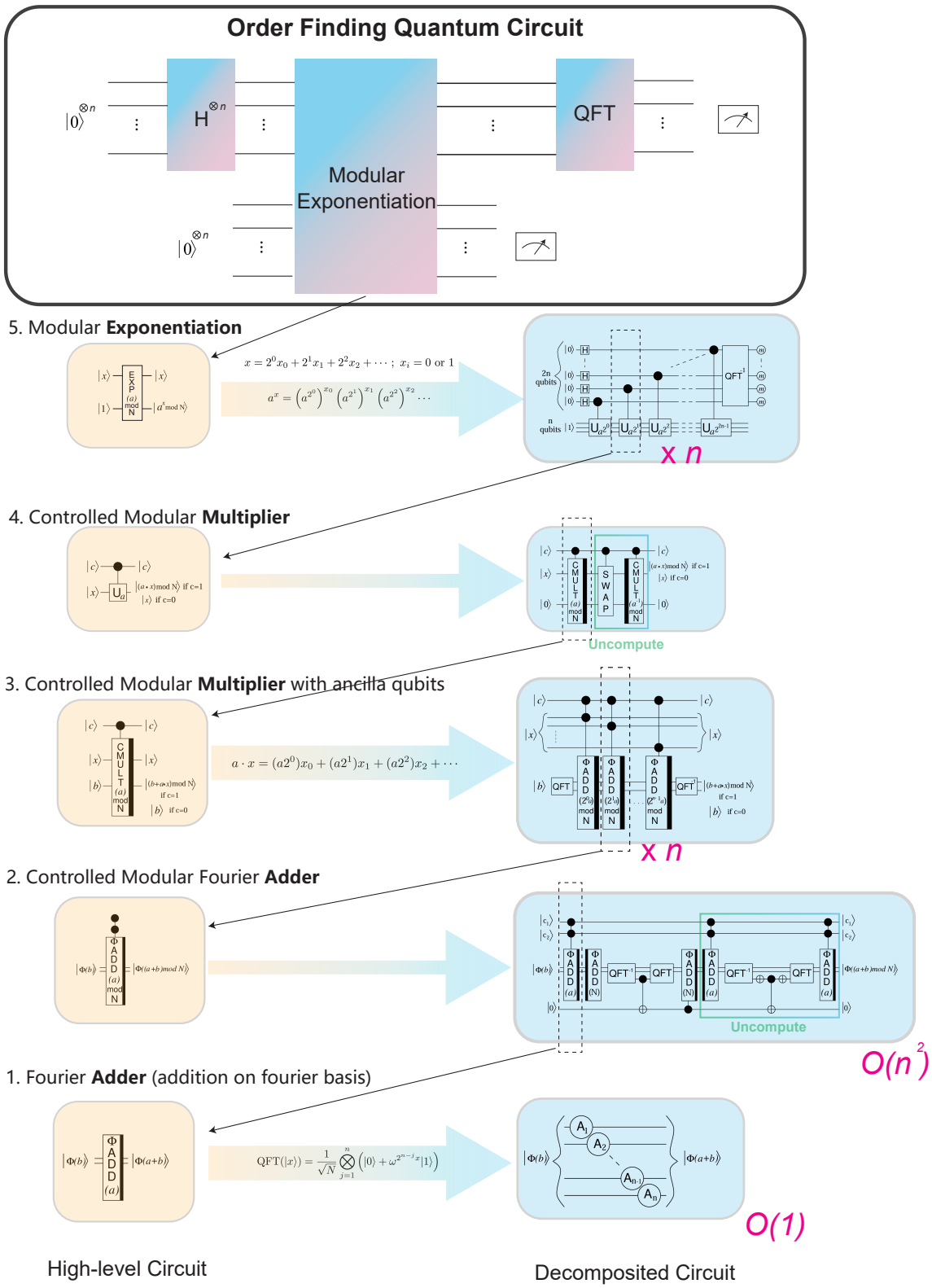

Figure 4: Road map of how to build modular exponentiantion gate which is the main part of the Order finding circuit. Insets are adapted from the 2003 paper [4].

In this section, we introduce the most resource-intensive and structurally complex component of the order-finding circuit: the modular exponentiation subroutine. This block dominates both the gate count and circuit depth, and therefore constitutes the central challenge in realizing Shor's algorithm on quantum hardware. Figure 4 presents a roadmap illustrating how the modular exponentiation unit can be systematically decomposed into simpler building blocks. In the remainder of this section, we follow this hierarchy and construct the circuit in a bottom-up way, beginning with the implementation of modular addition, then extending the design to modular multiplication, and finally assembling these components to obtain the full modular exponentiation module.

1. Fourier Adder $\phi\mathrm{ADD}(a)$

The Fourier adder $\phi\mathrm{ADD}(a)$ is an adder in Fourier space, i.e. it implements the mapping

$$\phi(b) \longmapsto \phi(b+a),$$

where ϕ means a Quantum Fourier Transform. To build the Fourier Adder, we will exploits the explicit structure of the Quantum Fourier Transform (QFT) on computational basis states again. For an n-qubit register encoding an integer b, the QFT has the product form

$$\mathrm{QFT}\,|b\rangle = \frac{1}{2^{n/2}} \bigotimes_{k=0}^{n-1} \left(|0\rangle + e^{2\pi i\, b/2^{k+1}}|1\rangle\right),$$

so that each qubit k carries a phase factor $e^{2\pi i b/2^{k+1}}$ that encodes b with progressively decreasing precision. where a is a known classical integer. In the QFT basis, replacing b by $b+a$ multiplies the phase of qubit k by

$$e^{2\pi i(b+a)/2^{k+1}} = e^{2\pi i b/2^{k+1}} \cdot e^{2\pi i a/2^{k+1}}.$$

Since the first factor is already present in the QFT state, the entire action of the adder reduces to attaching the additional phase $e^{2\pi i a/2^{k+1}}$ to the $|1\rangle$ component of each qubit. Writing the classical number a in binary,

$$a = \sum_{j=0}^{n-1} a_j 2^j,$$

we obtain

$$e^{2\pi i a/2^{k+1}} = \prod_{j=0}^{k} e^{2\pi i a_j/2^{\,k+1-j}},$$

which shows that qubit k requires only a known phase rotation conditioned on those bits $a_j = 1$. Because these rotations are single-qubit $R_z(\theta)$ gates with predetermined angles, the entire addition requires only $O(n)$ elementary gates and no multi-qubit controlled operations on quantum data. Thus, in the Fourier basis, addition by a classical constant can be implemented simply by adjusting the fractional phases already present in the QFT representation, avoiding the need for any carry propagation.

2. Modular Fourier Adder $\phi\mathrm{ADD}(a) \bmod N$

To compute

$$b \longmapsto (b+a) \bmod N,$$

the reversible modular adder performs:

1. Apply $\phi\mathrm{ADD}(a)$.
2. Apply $\phi\mathrm{ADD}(-N)$ and use an inverse QFT to detect overflow (when a+b is larger than N).
3. If there is no overflow, restore the subtracted N.
4. Return to the Fourier basis and uncompute the ancilla.

This requires $n+4$ qubits and $O(nk_{\max})$ gates.

3. Controlled Modular Multiplier with ancilla qubits CMULT(a) mod N

The controlled modular multiplier with ancilla qubits implements, on three registers (c, x, b), the map

$$|c\rangle\,|x\rangle\,|b\rangle \;\longmapsto\; \begin{cases} |0\rangle\,|x\rangle\,|b\rangle, & c = 0, \\ |1\rangle\,|x\rangle\,|b + ax \bmod N\rangle, & c = 1, \end{cases}$$

that is, when the control qubit c is 1 we add ax mod N into the "accumulator" register b, and when $c = 0$ the state is left unchanged. Writing the n-bit integer x in binary as

$$x = \sum_{j=0}^{n-1} x_j 2^j,$$

we obtain

$$ax \equiv \sum_{j=0}^{n-1} x_j \big(2^j a\big) \pmod{N}.$$

Hence the unitary can be decomposed into a sequence of doubly controlled modular additions

$$|c\rangle\,|x_j\rangle\,|b\rangle \;\xrightarrow{c=1,\,x_j=1}\; |c\rangle\,|x_j\rangle\,\big|\,b + 2^j a \bmod N\big\rangle,$$

namely, we apply $\phi_{\mathrm{ADD}}(2^j a \bmod N)$ controlled on both the global control qubit c and the j-th data qubit x_j. Using the modular adder from the previous subsection as a building block, each controlled modular addition requires $O(nk_{\max})$ elementary gates for n-bit arithmetic, where $k_{\max}$ denotes the maximum number of single-qubit phase rotations used in the Fourier adder. Summing over all $j = 0, \ldots, n-1$, the entire controlled modular multiplier CMULT(a) mod N thus uses $O(n^2 k_{\max})$ gates and a total of $2n+O(1)$ qubits: n qubits for the input x, n qubits for the accumulator b, and a constant number of ancilla qubits needed for the modular reduction.

4. Controlled Modular Multiplier U_a

The gate

$$U_a : |y\rangle \longmapsto |ay \bmod N\rangle$$

is implemented by:

1. Controlled-CMULT(a) mod N,
2. A register swap,
3. Controlled-CMULT(a^{-1}) mod N,

ensuring reversibility and clean ancilla qubits.

5. Modular Exponentiation

Controlled-$U_{a^{2^j}}$ gates are applied for each control qubit j. The complete exponentiation requires $O(n^3 k_{\max})$ gates and dominates the resource cost of Shor's algorithm.

4.6 Summary of Resource Requirements

Using an approximate QFT with $k_{\max} = O(\log n)$, the full implementation uses:

- $O(n)$ qubits,
- $O(n \log n)$ gates for a QFT,

- $O(n^2 \log n)$ gates for a modular multiplication,
- $O(n^3 \log n)$ gates for a modular exponentiation,
- Circuit depth $O(n^3)$.

These resource bounds match the most qubit-efficient implementation of Shor's algorithm using elementary gates.

5 Qubit Resource Complexities and QEC

In order to understand how error correction can occur on our system, we first need to understand the existing QEC methods and determine which will be applied to our project based on what is the most physically feasible. This will ensure our project remains experimentally and physically viable. This section will also detail some potential optimizations that can be made and the current state of QEC research.

5.1 Architecture of QEC

In standard 2D fixed-layout architectures the industry standard is surface codes or variants thereof (i.e. rotated surface code). These are used in systems like Google's and IBM's superconducting quantum chips. Surface codes are primarily designed for nearest neighbor connectivity systems. In order for a surface code to implement logical operations, like the CNOT (CX) existing in our gate set chips must perform lattice surgery. This is the temporary "stitching" of two distinct surface code patches into one larger patch. This allows parity measurement without requiring individual quantum information itself. An excellent authorative source on lattice surgery in surface codes is by Dominic Horsman and co-writers[5]

There exists some platforms such as trapped ion and neutral atom that have "non-local" connectivity and are not restricted to just nearest neighbor connectivity. This allows them to implement mathematically superior codes to surface codes but are not possible to build on a standard quantum chip. For reference, trapped ions can implement color codes that allow transversal gates. This allows CX and similar gates to be performed by applying physical operations to all qubits pairwise and avoids lattice surgery [6]. Neutral atoms can use Quantum Low-Density Parity Checks which have a constant encoding rate. This means the physical qubit requirement is significantly lower than a surface code. An early verson of this code on a 2D reconfigurable array was demosntrated by a Harvard and QuEra team, however it is not at a stage where it can be applied to real world chips [7].

For the purposes of our investigation into complexities and overheads of QEC for Shor's algorithm we will use a surface code architecture that implements lattice surgery to perform the CX gate. This is because of the surface codes demonstrated effectiveness and feasibility in real systems.

5.2 QEC Cycle with Surface Code

The QEC cycle in a surface code at a high level and control requirements is as follows:

1. At the beginning of each cycle, ancillary qubits are reset and put into the ground state. Either the 0 state or the + state depending on Z and X stabilizers respectively.
2. 4 CX are performed between measurement qubit and four nearest neighbor qubits. Typically the North, West, East, and South nearest qubits.
3. Measurement for X errors is done via 4 Z-stabilizers depending on the syndrome produced by the CX gates.

4. Errors are tracked in a classical control software and corrections are applied virtually by re-interpreting the future measurement based on tracked errors. Note that physical corrections are only applied when necessary.

This means control requires classical monitoring, which means the classical computer must operate faster than qubits. It must match the minimum weight perfect matching in real-time to prevent an exponential backlog of error data. So, the speed of the logical clock in the circuit is limited by the speed of the classical decoding loop in this architecture. Superconducting circuits are the best candidates because of their fast measurement cycles and their ability to support the required nearest-neighbor architecture

6 Derivation of Total Qubit Complexity

In a Shor's algorithm circuit the number of physical qubits required depends on a tradeoff of physical size of the circuit and computation time. More qubits means larger space complexity but lower time complexity. We can determine the number of physical data qubits required by the standard equation for surface codes $Q_d = N_L(2d+1)^2$ where N_L is the number of logical qubits and d is the code distance[8].

Clifford gates can be implemented without requireing new T-states or additional qubits. As referenced in section 4.3, R_Z gates are the only non-Clifford gate in our deconstruction and therefore are the primary cost driver outside of just the necessary qubits to fulfill the data requirements. R_Z gates can be approximated using Clifford and T gates. So we must derive the total amount of T-factories based on the number of R_Z gates. We will see that the number of T factories is variable such that, more factories has higher qubit requirements but less time complexity, and visa versa.

Note that CX does not require additional qubits due to lattice surgery techique which trades additional qubits for space to move via stitching.

6.1 Derivation of Total Physical Qubit Number

Now that we have defined the number of required physical data qubits. We now need to derive an equation to determine the number of physical qubits for the R_Z gates.

Let:

- Q_t =total number of physical qubits
- N_L =number logical qubits
- N_T =number of T gates
- N_f =number of T factories
- Q_f =number of qubits from T factories
- Q_d =number of data qubits

We can define total number of qubits to be

$$Q_t = Q_d + Q_f$$

Where we know from the previous subsection:

$$Q_d = N_L(2d+1)^2$$

The number of qubits from T factories can be defined as follows. C can be thought of as a "speed factor" where higher C gives higher numbers of qubits and faster runtimes as mentioned in the previous section.

$$Q_f = N_f C d^2$$

For this derivation we will choose $N_f = 2N_L/C$. This is based on the Heuristic argument that T-factories are reasonably optimized for time and space complexity when they take up 50 percent of the chip space. Once again, note this is changeable based on chosen optimization parameters.

This simplifies our total equation to:

$$Q_t = 2(d+1)^2 N_L + N_f C d^2 = 2(d+1)^2 N_L + 2N_L d^2 = 2N_L(2d^2 + 2d + 1)$$

We must now determine the code distance such that a logical error rate is suppressed for the duration of the algorithm. The probability of a logical error per cycle is about

$$P_L 0.1(p_{phys}/p_{th})^{(d+1)/2}$$

where p_{th} is threshold error rate for surface code and p_{phys} is the physical error rate of qubits. We need to survive N_T operations such that $N_T * P_{logical} < P_{fail} = N_T P_L$ When we substitute our equation for P_L and rearrange our equation to solve for d we get

$$d = \frac{2ln(N_T/P_{fail})}{ln(p_{th}/p_{phys})} - 1$$

Then we must determine N_T in terms of N_L. By the Ross-Selinger algorithm (an iteration on Solovay-Kitaev), we know that the number of T gates needed to approximate one R_Z to leading order goes as follows.

$$T_{R_z} \approx log_2(1/\epsilon)$$

Where ϵ is the maximum allowed error per rotation $\epsilon \approx P_{fail}/N_R$ where N_R is the number of rotational gates. We can approximate $N_R/P_{fail} \approx N_L^3 log(N_L)$ for a complete circuit (this omitts the denominator for simplicity of the approximation since P_{fail} is significantly small), so substituting in for N_R/P_{fail} then we see that

$$d \approx \frac{ln(N_L^3 log(N_L)}{ln(p_{th}/p_{phys})}$$

Now we can substitute in d to our total qubit equation to get

$$Q_t = 2N_L(2(\frac{ln(N_L^3 log(N_L))}{ln(p_{th}/p_{phys})})^2 + 2(\frac{ln(N_L^3 log(N_L))}{ln(p_{th}/p_{phys})}) + 1)$$

Which gives our qubit amount complexity considering QEC of order

$$O(N_L * log(N_L)^2)$$

7 Time Complexity

We assumed a reasonably fast circuit with the optimization of 50 percent of space allocated for T factories. However, it is important to note that time complexity, while not derived in this paper, is variable depending on the optimization chosen. As mentioned, more T factories will be faster but will increase spatial complexity.

In addition, there are different ways to optimize the time complexity of a modular exponentiation oracle. In standard, the modular exponentiation is broken down into fundamental components and done in sequence. However, we used a process called "oblivious runway adders," which enables parallel processing. This reduces the complexity of the time by a factor of N_L [8].

8 Summary

In this work we have carried out a detailed, asymptotic analysis of the quantum resources required for implementing Shor's factoring algorithm. Rather than treating the algorithm as a monolithic polynomial-time procedure, we decomposed its main quantum component—modular exponentiation—into its basic arithmetic primitives: Fourier addition, modular addition, modular multiplication, and the controlled modular exponentiation gates used in the order-finding subroutine. For each component, we derived the scaling of gate count, circuit depth, and qubit requirements under compilation to a realistic universal gate set.

A key focus of our analysis was the use of the approximate Quantum Fourier Transform (AQFT), whose truncated structure reduces the cost of arithmetic subroutines from $O(n^2)$ to $O(n \log n)$ without compromising the correctness of Shor's algorithm. Building on this, we found that modular multiplication requires $O(n^2 \log n)$ gates, and modular exponentiation—the dominant component of the algorithm—requires $O(n^3 \log n)$ gates while using only $O(n)$ logical qubits.

To relate these logical-level complexities to physically realizable implementations, we incorporated a coarse surface-code model of fault-tolerant quantum computation. This allowed us to estimate how the number of physical qubits grows with problem size once magic-state distillation for non-Clifford operations is included. While our analysis abstracts away device-specific constants, it captures the dominant scaling behavior and clarifies the structural reasons why modular exponentiation drives the overall resource cost of Shor's algorithm.

Overall, our results place the polynomial-time characterization of Shor's algorithm into a more concrete circuit context, revealing the primary contributors to its space–time complexity and providing a framework for comparing algorithmic variants and hardware architectures at the asymptotic level.

References

[1] A. Barenco et al., "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, pp. 3457–3467, 5 Nov. 1995. DOI: 10.1103/PhysRevA.52.3457. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.52.3457.

[2] J. Preskill, *Lecture notes for ph219/cs219: Quantum information and computation, chapter 5*, Updated July 2015, 2015. [Online]. Available: https://www.preskill.caltech.edu/ph219/chap5_13.pdf.

[3] *Quantum fourier transform — wikipedia, the free encyclopedia*, https://en.wikipedia.org/wiki/Quantum_Fourier_transform, Accessed: 2025-12-5.

[4] S. Beauregard, *Circuit for shor's algorithm using 2n+3 qubits*, 2003. arXiv: quant-ph/0205095 [quant-ph]. [Online]. Available: https://arxiv.org/abs/quant-ph/0205095.

[5] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, "Surface code quantum computing by lattice surgery," *New Journal of Physics*, vol. 14, no. 12, p. 123 011, 2012. DOI: 10.1088/1367-2630/14/12/123011.

[6] Y. Wang et al., "Fault-tolerant one-bit addition with the smallest interesting color code," *Science Advances*, vol. 10, no. 29, eado9024, 2024. DOI: 10.1126/sciadv.ado9024.

[7] Q. Xu et al., "Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays," *Nature Physics*, vol. 20, no. 7, pp. 1084–1090, 2024. DOI: 10.1038/s41567-024-02479-z.

[8] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, p. 433, 2021. DOI: 10.22331/q-2021-04-15-433.