Characterizing the Performance Effect of Trials and Rotations in Applications that use Quantum Phase Estimation

Shruti Patil*, Ali JavadiAbhari*, Chen-Fu Chiang†, Jeff Heckey‡, Margaret Martonosi* and Frederic T. Chong‡
*Department of Computer Science, Princeton University
†Department of Mathematics and Computer Science, University of Central Missouri
‡Department of Computer Science, University of California at Santa Barbara
Email: *{spatil,ajavadia,mrm}@princeton.edu, †cchiang@ucmo.edu, ‡jheckey@ece.ucsb.edu, ‡chong@cs.ucsb.edu

***Abstract*—Quantum Phase Estimation (QPE) is one of the key techniques used in quantum computation to design quantum algorithms which can be exponentially faster than classical algorithms. Intuitively, QPE allows quantum algorithms to find the hidden structure in certain kinds of problems. In particular, Shor's well-known algorithm for factoring the product of two primes uses QPE. Simulation algorithms, such as Ground State Estimation (GSE) for quantum chemistry, also use QPE.**

Unfortunately, QPE can be computationally expensive, either requiring many trials of the computation (repetitions) or many small rotation operations on quantum bits. Selecting an efficient QPE approach requires detailed characterizations of the tradeoffs and overheads of these options. In this paper, we explore three different algorithms that trade off trials versus rotations. We perform a detailed characterization of their behavior on two important quantum algorithms (Shor's and GSE). We also develop an analytical model that characterizes the behavior of a range of algorithms in this tradeoff space.

I. Introduction

A class of computational problems exist for which no polynomial-time algorithm running on a classical Turing Machine has yet been proposed. However, a number of these algorithms—with important practical implications—have been found to be efficiently solvable using a quantum computer. This is known as the BQP (Bounded-error Quantum Polynomial-time) complexity class, and is conjectured to be different from the P (Polynomial-time) or even the BPP (Bounded-error Probabilistic Polynomial-time) class that can be solved efficiently on classical computers. The hope of accelerating important BQP class problems (e.g. factoring) has spurred interest in building quantum computers and in devising quantum algorithms to map computations onto them.

This paper studies an important technique in quantum computation: *Quantum Phase Estimation (QPE)*. QPE is a core building block computation for various quantum applications such as solving systems of linear equations [12], prime factorization [27], and finding answers to quantum many-body problems [31, 32]. In fact, there are really only three core building blocks for useful quantum algorithms: QPE, quantum random walks [9] (used for some graph search algorithms), and Grover's function [11] (used for Grover's search algorithm).

Because of QPE's central importance in many QC applications, characterizing its performance and identifying efficient implementation approaches is important. As one application example, consider Ground State Estimation (GSE). Finding the exact ground-state energy of a system of many particles is an intractable task for classical computers due to the exponential growth in computational cost as a function of the particles involved. However, using quantum systems themselves to store and process data, as is the case in quantum computers, this problem can be efficiently solved. Molecular energies are represented as eigenvalues of an associated Hamiltonian and can be obtained by encoding a molecular wave function into a set of quantum bits (qubits), simulating their evolution using quantum operations on those qubits, and then extracting the energy using quantum phase estimation. Experimental realizations of QPE for calculating chemical system energies have already been reported using linear optics [20].

The overall goal of QPE is to compute the eigenvalue of a quantum unitary circuit. This circuit is also referred to as the *Oracle*, since it can be regarded as a black box of computation, whose eigenvalue will reveal important information for solving the overall problem. Different applications have different oracle circuits of varying complexity, and this is a key factor in the tradeoff space of QPE techniques discussed in this paper.

QPE approaches for obtaining phase estimations have commonly rested on two underlying techniques:

- *Quantum Fourier Transform* (*QFT*): In this approach, qubits are first computed on via the oracle function, and then a QFT is performed to extract an eigenvalue. A full implementation of QFT requires potentially many, exponentially small quantum *rotation* operations, which may be costly to realize with good precision.
- *Repetitive Trials*: Using repetitive trials involves applying the oracle several times, measuring samples from the probability distribution of the eigenvectors, and finding the final answer via classical post-processing of the answers; this reconstructs an estimate based on the collected measurements. This technique reduces the number of quantum rotation operations required (compared to QFT-based approaches) but may involve many trials in order to obtain an accurate estimate.

TABLE I: Comparison of three QPE Methods, with the idealized assumption of perfect rotation precision. (With more realistic imperfect rotation precisions, AQFT will have a non-zero number of trials since it uses repeated trials to regain precision.) ACPA is a middle ground between KHT and AQFT in terms of the number of rotations and trials involved.

	Number of Rotations	Number of Trials
KHT	None	Many
AQFT	Many	None
ACPA	Some	Some

Given these two general implementation styles, specific QPE algorithms vary in their use of them. Table I shows a coarse-level comparison of three distinct approaches for

978-1-4799-6454-3/14/$31.00 ©2014 IEEE

Quantum Phase Estimation which cover a spectrum of possible methods:

- *Kitaev Hadamard Tests (KHT)*: The approach originally proposed by Kitaev [17] relies on a predetermined number of trials to achieve a desired target for the error-rate and precision of estimation.
- *Approximate Quantum Fourier Transform (AQFT)*: This approach is based on QFT, but uses Approximate QFT [4] instead of the full QFT in order to curb the number of rotations.
- *Arbitrary Constant Precision Algorithm (ACPA)*: ACPA is an approach proposed by Ahmadi and Chiang [3] which offers a method between KTH and AQFT in terms of the number of rotations and trials involved.

This paper provides the first detailed performance comparison of these QPE techniques within real, full-scale QC applications. Overall this paper makes the following contributions:

- Through implementing different QPE methods, we perform an empirical workload characterization of the tradeoff between performing higher-precision rotation operations versus performing more trials for obtaining accurate phase estimates.
- We characterize two important quantum applications that use quantum phase estimation as a fundamental building block: Shor's algorithm for prime factorization, and the Ground State Estimation algorithm for calculating molecule energy levels. We analyze the effect of different methods of QPE in each of these algorithms.
- We find that practical resource constraints limit parallel execution within our QPE methods, heavily influencing which methods have the lowest runtime.
- We also find that the application also heavily influences the appropriate QPE method, since some methods require more redundant execution of the application than others.

The rest of this paper is organized as follows: Section II gives background on basic QC concepts and our QC algorithm toolflow. Section III describes QPE in detail, and Section IV describes the different implementation approaches we consider. Section V analyzes the resulting design space tradeoff. Section VI discusses related work, and Section VII concludes. We also include an appendix that derives key application characteristics from Shor's algorithm and Ground State Estimation.

II. Background

This section offers a brief background on basic concepts in quantum computation.

A. Quantum Bits and Operations

A quantum bit (or qubit) is not confined to a binary state. At any time, the state of a qubit can be a linear *superposition* of the 0 and 1 states denoted as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers subject to $|\alpha|^2 + |\beta|^2 = 1$. Therefore, a quantum state can be thought of as a vector of norm 1, with an arbitrary phase—a concept which can be extended to n qubits by thinking of the state of an n-qubit system as a unit vector in the 2^n-dimensional space. This can be represented as a point on the surface of a 2^n-dimensional sphere, commonly referred to as the *Bloch Sphere* (Fig. 1).

Quantum operations are merely norm-preserving rotations, which move the state of an n-qubit system along the surface of the Bloch sphere. Consequently, every quantum operation is a "rotation" operation. However, in the QPE subroutine which is the focus of this paper, the term "rotations" specifically refers to those that are performed about the Z axis only, to distinguish a specific set of operations. For arbitrary rotation angles, no straightforward, error-tolerant implementation on a physical device is known. This forces us to approximate them using a set of "standard" gates—akin to instruction selection in classical compilers. Section III-A discusses this "rotation decomposition" in more detail.

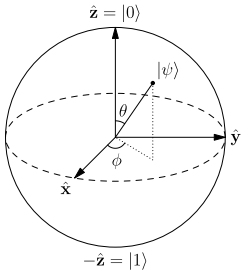

Fig. 1: The Bloch sphere is a good visualization of the quantum state space. Points on the sphere correspond to unique states, and quantum operations are equivalent to moves on the sphere.

B. Quantum Circuits

Ultimately, quantum computation must provide a classical answer to a classical query. Therefore, it must be possible to observe the state of the quantum system, via a *measurement* operation. Measuring a quantum state collapses it to the classical values of 0 or 1, with probabilities proportional to the amplitude of the superposition coefficients.

Following the analogy of a classical logic circuit, a *quantum circuit* can be used to represent a collection of qubits and gates, illustrating the operations performed in a quantum algorithm step-by-step. In these circuits, quantum single- or multi-bit operations are shown with boxes labeled with the operation. Measurement is shown with a "meter" symbol. Single wires denote a qubit, and double wires denote classical bits which are the outputs of measurement. Finally, "controlled" operations are shown with a wire connecting the controlling qubit to the target operation. The meaning is that the operation will be performed if and only if the controlling qubit is in the $|1\rangle$ state. The reader can refer to Figures 4-6 for examples of quantum circuit diagrams.

This paper uses diagrams of quantum circuits to compactly express computations, but our toolflow (described in Section II-D) starts from quantum algorithms written in a high-level programming language and compiles down to an assembly-language level. Resource estimates and algorithm runtimes are generated by our compiler infrastructure. The use of circuit diagrams to show the computation flow is analogous to illustrating a classical computation using compiler-level control/dataflow graphs.

C. Quantum Error Correction

Since quantum operations are inherently probabilistic, and since qubits are extremely sensitive to noise, Quantum Error Correction plays a central role in building quantum computers.

An important result known as the *threshold theorem* states that quantum computation can be made robust against errors when the error rate is below a certain threshold [2]. Intuitively, a quantum computer can sustain its computation if a single error can be corrected before two errors occur. Given an error correction scheme, a reliability threshold can be calculated. This implies that scalable quantum computing is in theory feasible, even though significant engineering challenges remain.

In order to take advantage of the threshold theorem, *Quantum Error Correction* must occur periodically during computation to ensure errors are corrected faster than they are accumulated. A typical error correction scheme, for example, uses a two-level, recursive Steane error correction code [29], where each qubit is augmented with 49 extra qubits and each timestep of computation is backed up by 23,409 additional timesteps [22]. Reducing this cost is extremely important and the amount of achieved reduction can easily determine whether the computation executes in practical timescales (eg. not 100 years) [22].

Although the bulk of resources in any quantum circuit implementation is due to added error correction redundancies (e.g. if every timestep of computation requires 23,409 timesteps of error correction), the algorithmic level (also referred to as the logical level) has very high leverage in controlling this. Algorithms with fast runtimes are less prone to the accumulation of errors and can tolerate fewer error correction steps. Thus, finding better high-level QC algorithms and optimizing their implementation efficiency pays off multiplicatively because of the further benefits that accrue in reducing quantum error correction required.

In our comparison of QPE schemes, we focus on logical qubits and logical timesteps. That is, we estimate costs before error correction is added. We do this because error correction overheads are heavily dependent upon technology and coding choices. We attempt to remain independent of those choices. Our logical comparison is quantitatively valid as long as all the alternatives used end up using the same amount of error correction per logical bit and logical operation. Our comparisons, however, are still qualitatively valid when trying to determine the QPE scheme with the lowest runtime. Adding error correction will maintain runtime relationships, since longer logical runtimes will experience equal or greater error correction overhead than smaller runtimes.

D. Scaffold Language and the ScaffCC Compiler

This work characterizes the performance impact of QPE implementation choices on full-scale quantum applications. To accomplish this, we work with full QC apps and QPE methods written in Scaffold, a C-like quantum programming language [13]. Our evaluations of resource requirements and program runtimes are based on analysis performed as part of the ScaffCC compiler framework [14]. ScaffCC synthesizes logical quantum circuits from a high-level language description. Scaffold includes data types to distinguish quantum and classical bits and offers built-in functions for quantum gates.

Our work here primarily manipulates *logical* qubits without considering their error correction. Subsequent passes could implement additional functionality to co-schedule the logical qubits and their gate operations along with QECC ancilla bits and their operations.

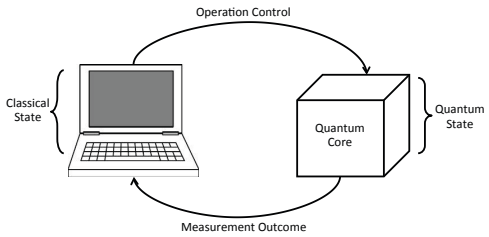

Fig. 2: Quantum computation is performed on a distinct "co-processor" controlled by a classical computer.

E. Execution Model

Ultimately the QC programs we consider here would be executed on QC hardware. While specific options for QC implementation technology vary widely, Fig. 2 shows a general approach in which quantum computations execute on a co-processor unit controlled by a classical computer. Within the quantum processor are several *operating zones* or gates, which can each perform distinct operations in parallel on distinct qubits. Each of the QC operating zones can be controlled to execute any of the gate types this machine has chosen to implement. The gate they implement can be changed from cycle to cycle via control bits sent over from the classical computer. (One can think of the QC operating zones being analogous to an ALU in a classical computer; with a small number of bits, ALUs can be controlled to perform an addition one cycle, subtraction the next.) The *runtime* of a QC is, to first order, the number of these gate operations (i.e., cycles or steps) that need to occur in sequence in order to complete the computation. (Several may happen in parallel.) When QC algorithms are drawn as circuits, the timesteps needed to complete the calculation is referred to as the *circuit depth*.

As with any real-world computer, computation on a QC is fundamentally constrained by the availability of hardware resources. For example, the number of operating zones available poses a fundamental limit on the number of simultaneous parallel computations that are possible. Likewise, the number of logical qubits is another fundamental constraint: in some QCs, qubits are physically large (relative to state bits in classical computers) and in many QCs, sufficient inter-qubit spacing must be provided in order to minimize unintended state interactions or interference that leads to qubit decoherence. In addition, since each logical qubit must be underpinned by dozens or hundreds of physical qubits implementing QECC, the number of logical qubits required by a computation becomes a fundamental constraint. In this work, we use *qubit count* as an indicator of parallelism: more qubits being computed upon means more parallelism. In addition, qubit count can denote a resource constraint: some experiments limit the maximum number of qubits to evaluate application runtime under less idealized, finite-hardware assumptions.

III. Quantum Phase Estimation

If a quantum unitary operator U acting on an *m*-qubit input vector yields:

$$U|u\rangle = e^{i2\pi\varphi}|u\rangle, \tag{1}$$

then $|u\rangle$ is said to be an *eigenstate* (or *eigenvector*) of the corresponding unitary matrix of U. In this case, the *eigenvalue* is a phase shift $e^{i2\pi\varphi}$ that is introduced in the state vector. The goal of QPE is to estimate the value of phase φ in the

eigenvalue for a given unitary quantum circuit U. This phase value reveals important information about the hidden structure of the quantum oracle function in algorithms such as *order finding*, the core computation of Shor's algorithm.

Generally, estimating φ is a challenging task. Two factors affect the adequacy of estimation: The *precision* δ determines how close to the actual eigenvalue the estimation outcome is. The *error rate* ϵ determines how likely it is, given the probabilistic nature of quantum states, for the final estimate to have the desired precision (equivalent to $1 - Prob(success)$).

We use $\widetilde{\varphi}$ to denote an estimate of φ. Since the overall phase shift lies in the $[0, 2\pi)$ interval, an estimate of $0 \leq \varphi < 1$ with n bits of precision is written as:

$$\widetilde{\varphi} = \overline{0.x_1x_2x_3...x_n} \tag{2}$$

Where $\overline{0.x_1x_2x_3...x_n}$ is a notation for $\sum_{i=1}^{n} x_i * \frac{1}{2^i}$ and $x_i \in \{0, 1\}$.

A. Quantum Rotation Decomposition

As mentioned in Section II-A, there are an infinite number of single-qubit rotations that can be done on a Bloch sphere, but we can implement a finite subset on a physical computer. A major cost in some of the QPE approaches arises when algorithms must decompose arbitary rotations—especially precise small rotations—into a *sequence* of physical operations.

Decomposition is possible due to the universality of a small set of quantum gates for representing any single-qubit operation. For example, Hadamard (H), Phase (S), Controlled-NOT (CNOT) and the $\frac{\pi}{8}$-gates (T) form a universal set [6]. If we compare applying the decomposed set of gates instead of the actual (perfectly precise) rotation to a particular qubit state, the decomposition *precision* indicates how closely the approximate outcome state would resemble the exact state (i.e., the likelihood that measuring it would yield the same answer.)

For an arbitrary angle, we need an approach for determining the correct decomposed sequence. In this paper we employ the Single Qubit Circuit Toolkit (SQCT) proposed by Kliuchnikov et al. [18], which offers practical decompositions of up to accuracy 10^{-15}, based on the results of [19].

For the purpose of quantum phase estimation, we are interested in rotation angles used by the quantum Fourier transform, which are of the form $R(\frac{2\pi}{2^k})$, and the tradeoffs that are present in limiting their precision. Fig. 3 shows the length of decomposition obtained from SQCT [28] for these rotation operations. Higher precision decompositions can be directly linked to a longer gate sequence, although at a particular precision level, there is no significant difference between different angles.

Using smaller angles in an algorithm means that the precision of decomposition must be increased, in order to have a faithful approximation of those rotations. Our model (Section V) assumes that the presence of angle $\theta = 2\pi/2^k$ means that decomposition should take place at least with precision:

$$|\theta - \widetilde{\theta}| < \frac{1}{2^k}. \tag{3}$$

The lower dotted line shows this trend, and guides the actual precision that we should pick in different scenarios, depending on the degree of the smallest angle present. This increase in precision would affect the approximation of all gates, thus being costly for circuit size.

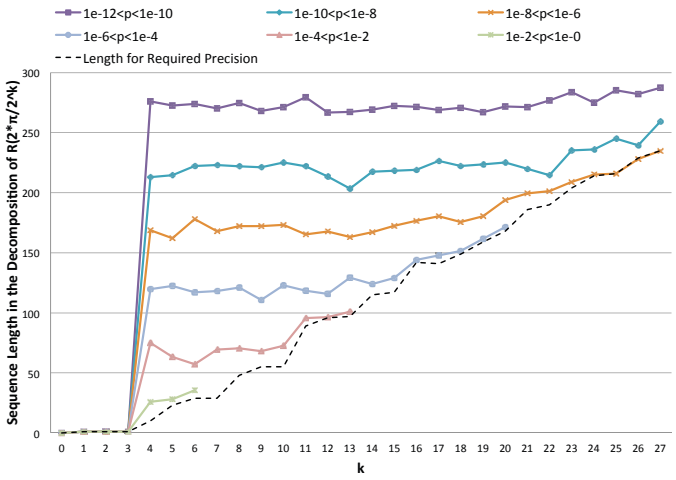

Fig. 3: The length of the sequence of gates generated by SQCT to approximate a rotation operation with angle $\frac{2\pi}{2^k}$, as a function of k. Increased precision (p) yields progressively longer sequences. For $k = 0$ there is no rotation, and for $k = 1, 2, 3$ the standard gates of Z, S, T are obtained respectively. For some loose precisions, small-enough angles approximate to 0.

IV. Implementations of Quantum Phase Estimation

This section describese the three QPE methods and their relative strengths and weaknesses. Overall, they span a design space that presents tradeoffs of runtime (circuit depth), operation-level parallelism (circuit width) and resource usage (number of qubits and gates). They also implicitly differ in the accuracy with which they estimate the phase of the input eigenvector. In order to make the approaches comparable for our characterization, we set each of their success probabilities $(1 - error\ rate)$ to 0.8 in our experiments, i.e. an 80% total probability of obtaining the correct outcome.

A. Kitaev's Hadamard Test with S gate (KHT)

Kitaev [17] proposed an algorithm that avoids arbitary quantum rotations, which expand into long sequences of physical operations as shown in Fig. 3. Unfortunately, the algorithm uses a large number of independent trials to perform QPE to required precision. These trials can be parallelized, but can require an impractical amount of quantum hardware to do so. Our results in Section V-C will elaborate upon this.

Kitaev uses a series of Hadamard tests to perform the phase estimation, each of which requires a single phase-shift gate S:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{4}$$

This matrix is a transformation on a qubit vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ as previously described in Section II. The circuit that accomplishes Kitaev's Hadamard Test is shown in Fig. 4. When applied to the $|0\rangle$ qubit (no probability of measuring "1"), the Hadamard gate (H) results in a qubit that has equal probability of $|0\rangle$ and $|1\rangle$. The Hadamard gate is the inverse of itself and is often applied again before measurement, as in the Hadamard test. Each Hadamard test estimates one bit of the phase φ with a precision of $\frac{1}{16}$ [15]. The accuracy at which

we are computing the estimated phase $\tilde{\varphi}$ can be expressed as:

$$Pr(|\varphi - \tilde{\varphi}| < \frac{1}{2^n}) > 1 - \epsilon \tag{5}$$

To achieve a desired success rate, that is to succeed with $1-\epsilon$ probability, we need m_{KHT} trials of Kitaev's Hadamard test for each bit [3], where

$$m_{KHT} = 55 \ln \frac{4n}{\epsilon} \sim 55 \ln(20n) \tag{6}$$

In the above equation, we set $1/\epsilon$ such that the success probability $1 - \epsilon = 0.8$. This gives us the value of m_{KHT} that achieves an overall success probability equivalent to that of QFT. The estimation of each bit in Equation 2 occurs independently of the other bits. Therefore the circuit requires no controlled phase operations. This also allows many (or all) bits to be estimated simultaneously, subject to the resource constraint that enough qubits are available for the individual Hadamard tests. Thus, KHT has two levels of parallelism, fine-grained parallelism (analogous to instruction-level parallelism) within the computation in Fig. 4, and coarse-grained parallelism (analogous to task-level parallelism) between trials. Indeed, all the approaches have coarse-grained parallelism but only KHT has fine-grained parallelism across bits.

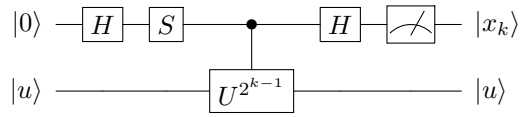

Fig. 4: Kitaev's Hadamard Test with S Gate (time flows from left to right in this circuit diagram)

B. QFT and Approximate QFT (AQFT)

Because KHT requires a large number of trials, a more common implementation of the QPE algorithm is based on Quantum Fourier Transform (QFT), shown in Fig. 5. QFT (and Approximate QFT) trade the expense of arbitrary quantum rotations for fewer trials in achieving desired QPE accuracy.

In the QFT approach, n ancilla qubits are used to form the upper register in the circuit, while n_{LR} qubits form the eigenstate in the lower register, whose periodicity (phase) is to be estimated. The last step of QPE is implemented using the inverse quantum Fourier transform ($QFT^\dagger$) on the ancilla states. The inverse QFT module will cause the probability distribution of the state of the upper register to be clustered around the correct phase value. With a Hadamard gate and a measurement, an estimated bit is revealed. A typical circuit for an n-qubit $QFT^\dagger$ is shown in Fig. 6. The estimated phase is computed from its least to its most significant place. Proceeding one qubit at a time, the estimation accuracy of higher significance qubits increases by taking into account the estimates of all previously determined qubits. This approach provides a success probability of at least $(4/\pi^2)$ [4]. When φ is an exact multiple of $1/2^n$, the success probability is 1.

The QFT circuit requires $O(n^2)$ rotations with degree up to $e^{i2\pi/2^n}$. In reducing this complexity, Barenco [4] showed that the lower bound of success probability of QFT can be achieved with fewer rotations per qubit, by considering the estimates of only the last k (also referred to as k_{AQFT}) qubits instead of *all* previous qubits. This approach achieves an asymptotic probability of $(4/\pi^2)$ when $k > \log_2 n + 2$ is selected [7], thereby reaching the lower bounds of QFT success guarantees. This bounds the number of rotations to $O(n \log_2 n)$ limiting their degrees to $e^{2\pi i/2^k}$. In practice, due to the logarithmic reduction in circuit length, AQFT provides a viable alternative that performs just as well (and sometimes better) than QFT in the presence of decoherence [4]. Therefore, we choose to use the AQFT circuit for our comparison.

A better lower bound on the success probability of AQFT has also been derived as $(4/\pi^2 - 1/16)$ [7]. When estimating an exact n-bit phase, the success probability of AQFT also approaches 1. However, these analyses assume that the rotations are precisely applied. In reality, rotations can only be achieved up to a precision factor (Section III-A discusses this approximation), which reduces the overall success probability. We will discuss this impact in the following section, in the context of both the AQFT method and the ACPA method which is described next.

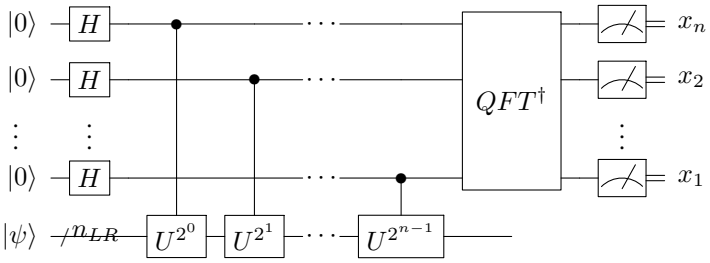

Fig. 5: n-qubit QPE circuit based on QFT to estimate the phase $\varphi = \overline{0.x_1 x_2 x_n}$.

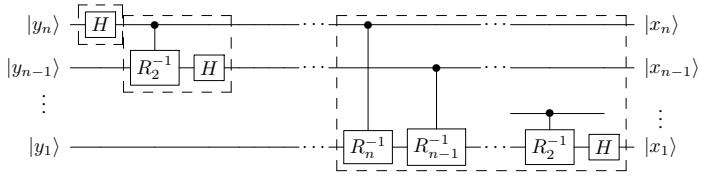

Fig. 6: n-qubit $QFT^\dagger$ circuit. This circuit requires rotations of exponentially increasing precision with the number of qubits.

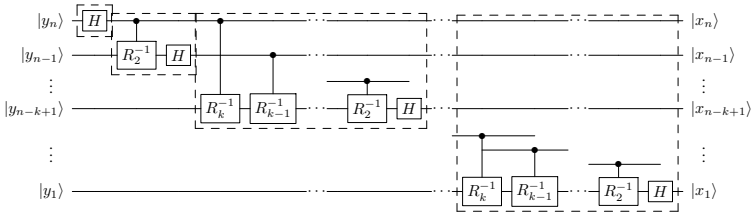

Fig. 7: Limiting the precision of rotations to achieve the n-qubit $AQFT^\dagger$ circuit to estimate the phase $\varphi = \overline{0.x_1 x_2 ... x_n}$, where $k >= 2 + log_2(n)$ is the highest degree of rotation applied. This is the same circuit used in ACPA, but with different requirements for the value of k.

C. Arbitrary Constant Precision Algorithm (ACPA)

The Arbitrary Constant Precision Algorithm (ACPA) [8] bridges the KHT and AQFT approaches, providing a design space where the number of arbitrary rotations can be reduced at the expense of more trials. Intuitively, AQFT performs a series of smaller and smaller (higher-precision) rotations to reach a desired accuracy. ACPA omits some of the smallest rotations at the expense of accuracy. This loss in accuracy is then recovered by performing more trials and then performing a majority vote on the classical bits resulting from measurements (the result is binary).

If the degree of rotation is limited to $e^{2\pi i/2^k}$ where $k \geq 3$ (k also referred to as k_{ACPA}), the number of trials required is:

$$m_{ACPA} = \frac{2ln(1/\epsilon')}{(1 - \frac{\pi^2}{2^{2k-1}})^2} \tag{7}$$

Here, $1 - \epsilon'$ is the success probability of estimating each bit. The overall success probability obtained is $1 - n\epsilon'$. However, the above expression also assumes that the rotations are perfectly implemented. Since practical rotations are imperfect, they increase the number of required trials to reach the desired accuracy. [8] derives the effect of imperfect rotations on the number of trials in ACPA. In particular, if the rotations can be achieved with an accuracy of at least $\eta = \frac{1}{(k-1)2^k}$, the number of trials is given by:

$$m_{ACPA} = \frac{2ln(1/\epsilon')}{(1 - \frac{\pi^2}{2^{2k-3}})^2} \sim \frac{2ln(5n)}{(1 - \frac{\pi^2}{2^{2k-3}})^2} \tag{8}$$

We use this equation to derive the number of required trials. Here, we set $1 - n\epsilon' \geq 0.8$, giving us $\epsilon' \sim 1/5n$ to accomplish an 80% success probability.

One can consider the AQFT approach as the limiting case of ACPA, i.e. when k_{ACPA} is chosen to be $> 2 + \log_2 n$, the two approaches implement the same circuit. Therefore, we can compute the number of trials required for the AQFT circuit as:

$$m_{AQFT} = \frac{2ln(5n)}{(1 - \frac{\pi^2}{2^{2k-3}})^2} \text{ with } k > 2 + \log_2 n \tag{9}$$

D. Phase Kickback (U_f)

Finally, looking at the circuits for all the QPE methods (Figures 4-7), we see that for any type of quantum phase estimation with accuracy of n bits, the range of quantum operations U^{2^k} with $0 \leq k < n-1$ must be applied. The modules of U, U^2, U^4, ..., $U^{2^{n-1}}$ are collectively called the *phase kickback* function and we refer to it as U_f. Fig. 8 depicts the phase kickback for the case of the AQFT circuit. (The first stage of AQFT and ACPA are indeed equivalent to one application of U_f; KHT splits U_f across its different trials.) U_f has in many cases its own interpretation— for example, in Shor's algorithm it is equivalent to a modular exponentiation module, raising its input to the power of a fixed number, modulo the number to be factored.

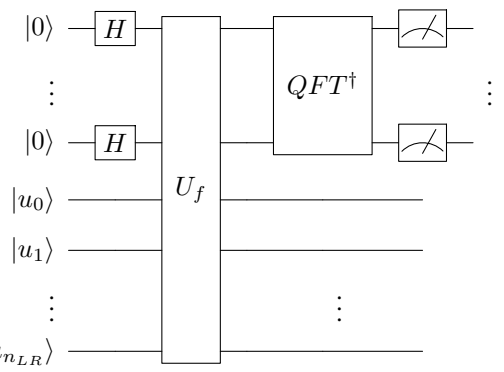

Fig. 8: The QFT-based quantum phase estimation circuit of Fig. 5 redrawn to reflect the U_f module. The cost of U_f is equivalent to 2^{n-1} times the cost of U.

V. Analysis of the Design Space

This section lays out the design space options in detail, and evaluates the tradeoff between QPE techniques, providing runtime results for different precision requirements and resource constraints.

Our analysis is divided into two categories. First, we have factors that constrain the number of trials that can be executed in parallel (task-level parallelism). These include the number of qubits as well as the number of available eigenstates in each application with QPE approach. These resource requirements affect how many parallel trials can be supported by the hardware. Second, we have a factor that affects the execution time of each trial. This is the size of the phase kickback function (U_f) that must be called across the many trials, thus contributing to the circuit size generated by the implementation.

A. Factors Affecting the Parallelization of Trials

These factors actually can also affect parallelization within trials, but we can assume that resources are sufficient for supporting parallel execution within at least one trial.

1) Number of logical qubits: Parallel execution of multiple trials will require multiple copies of quantum state, which we measure in logical qubit count. (The number of physical qubits is larger due to QECC.) Since arbitrary quantum data cannot be copied (the quantum *no-cloning theorem*), independent trials can only occur if they start from the beginning of the entire application or from an intermediate computation that starts with only freshly initialized qubits.

The maximally-parallel implementations require the most qubits since they require a copy of the entire computation for each trial. In general, if p is the parallelization factor achieved during an execution, the number of qubits required by the three methods is given by: $NQ = p * (n_{LR} + 1)$. Since the maximum parallelism available in the algorithms is across the trials and the bits, the number of qubits required can be as high as $mn(n_{LR} + 1)$ in maximally parallel implementations. In practical implementations, p ranges from 1 to mn. For example, KHT may be applied with $p = m$ such that the bits are processed serially while the m trials are performed in parallel. This reduces the requirement to $m_{KHT}(n_{LR} + 1)$ by reusing expensive qubit resources between successive bit estimations.

2) Number of eigenstates available: To estimate the eigenvalues through multiple trials, the phase estimation algorithms must be applied to the same eigenstate a number of times. For the trials to be parallelized, multiple copies of this eigenstate must be available. In general, the parallelization of trials is constrained by the number of such copies available. While this can be bounded by the number of available qubits in the system, in some cases it may simply be too expensive to prepare multiple copies of the states. However, for many applications including Shor's and GSE, these eigenstates can be generated by applying U_f to suitable initial states which are straightforward to prepare. Therefore, our study assumes that with sufficient resouce availability, the eigenstates can be prepared in parallel.

B. Factors Affecting Circuit Size

In addition to the specific computations described for each QPE approach in Section IV, the size and implementation cost of the U_f function is an important factor. Appendix A considers the cost of varying the size of U_f. Here we examine the implementation cost by studying the number of invocations of the U modules. The higher the invocation count of U, the greater the execution time of each trial and the greater the resources needed to support parallelism across trials.

The U_f function performs a central transformation in the QPE algorithm in creating the eigenstate whose periodicity is to be estimated. For many algorithms, the U_f function is also the most expensive part of the circuit. Therefore the cost of its implementation is an important consideration and provides an insight into its complexity. In general, the function U^k can be implemented using k copies of U. Hence, the number of invocations of U is given by:

$$N_U = m * (1 + 2 + 4 + ... + 2^{n-1}) = m * (2^n - 1) \quad (10)$$

In our characterizations, the GSE algorithm uses this scheme to implement U_f. However, for large n, this can result in an undesirably high number of invocations of U. For example, in Shor's algorithm, n is expected to be about 1000. Therefore, a more efficient implementation of U^k where the cost stays constant regardless of the value of k is desirable, and has been proposed for Shor's algorithm [25]. If such implementations are available for U, the number of invocations of U are $N_U = m*n$. For both types of implementations, the smaller the number of trials, the fewer the invocations of U, consequently, one can expect better overall runtime.

Table II shows a simplified comparison of the resources and runtime for the three methods, assuming that only the trials are parallelized. Choosing between Kitaev's algorithm and others depends on the relative cost of U_f and rotations, while choosing a suitable technique between that of ACPA and AQFT depends on the number of trials, number of rotations and the cost of the rotations. In the appendix, we determine these costs empirically.

C. Empirical Evaluation

To evaluate the different design metrics for the QPE implementations, we first limit the circuit to a micro-benchmark whose oracle is substituted by a proxy circuit consisting of simple CNOT gates. The intention is to explore the tradeoff space of QPE methods independent of oracle costs, which can vary significantly across applications. In particular, we show that resource constraints can limit parallelism and heavily influence the choice of QPE implementation. We later generalize our evaluation by plugging in realistic oracle costs and QPE precision targets based on parameters derived in the appendix.

Results were obtained using the ScaffCC compiler framework, which can measure resource usage in terms of logical qubits and logical gates, as well as compute runtime in terms of logical timesteps. The compiler can generate these measurements through static analysis because quantum applications are generally compiled with all inputs known.

1) The Effect of Desired Precision on Trials and Rotations: As described previously, the number of trials varies for the different methods for a given n, the desired QPE precision in bits. Fig. 9 shows how estimating more bits of QPE precision causes a sharper increase in the required number of trials in KHT as opposed to AQFT. The graph also depicts the impact of k_{ACPA} (the highest-precision rotation) on the ACPA method. The value of $k_{ACPA} = 3$ is insufficient to generate a high enough success probability in each trial demanding considerably large number of trials. For k_{ACPA} of 4 or higher, its behavior more closely tracks AQFT. On the other hand, using even a few controlled rotations helps the estimation process more than KHT, which does not take into account the values of the previously estimated bits.

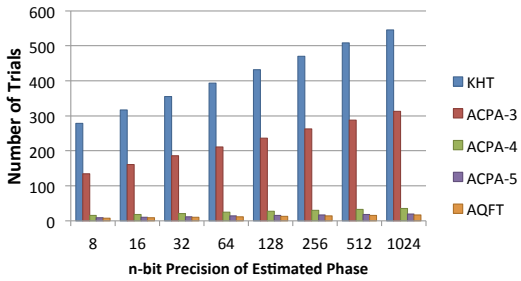

Fig. 9: Number of trials for KHT, ACPA-k and AQFT. Trials for ACPA-k are computed from Equation (8) assuming imperfect rotation gates.

2) Tradeoff between Resources and Runtime: A key tradeoff in our QPE schemes arises when parallelism is constrained by practical constraints in the number of logical qubits that can be implemented in a quantum machine. Specifically, the runtime of each QPE scheme is heavily dependent upon how much concurrency can be physically supported in executing independent trials.

Fig. 10 compares the runtime of algorithms as we bound the number of logical qubits for the QPE implementations requiring 64-bit precision of phase estimates. Low availability of qubits forces a serial execution of the algorithms, driving up their runtimes. With more resources, the runtime of each algorithm improves as their inherent parallelism in the algorithms gets exploited. For KHT, the trials as well as the bits can be processed in parallel; however exploiting this parallelism requires $\sim 10^6$ logical qubits. (With error correction, this could easily be $\sim 10^8$ physical qubits, an impractical number for the foreseeable future.) On the other hand, for ACPA and AQFT, the truly serial parts are the rotations that require bit-wise processing, thus their parallelism can be exploited with fewer available qubits. When a low number of qubits is available, ACPA executes the fastest among the three. When available

TABLE II: Analytical comparison of the three techniques with respect to number of gates, circuit depth and number of qubits, assuming that the trials are performed in parallel while bits are processed serially.

	KHT	ACPA	AQFT
Gates	$m_{KHT}(U_{f_{gates}} + 3n)$	$m_{ACPA}(U_{f_{gates}} + n * (k_{ACPA} - 1) * Avg\_R_{gates})$	$m_{AQFT}(U_{f_{gates}} + n * (k_{AQFT} - 1) * Avg\_R_{gates})$
Runtime	$U_{f_{timesteps}} + 3n$	$U_{f_{timesteps}} + n * (k_{ACPA} - 1) * Avg\_R_{timesteps}$	$U_{f_{timesteps}} + n * (k_{AQFT} - 1) * Avg\_R_{timesteps}$
Num of qubits	$m_{KHT}(1 + n_{LR})$	$m_{ACPA}(1 + n_{LR})$	$m_{AQFT}(1 + n_{LR})$

logical qubits becomes greater than 10^6, all algorithms can be fully parallelized, with KHT offering the best runtime.

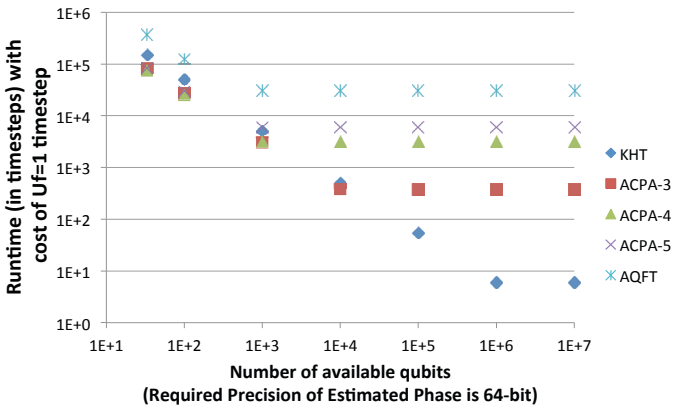

Fig. 10: Space-time tradeoffs presented by the three methods with bounded resource availability. With fewer available resources, ACPA executes fastest, while KHT executes fastest when available resources are as high as $\sim 10^6$.

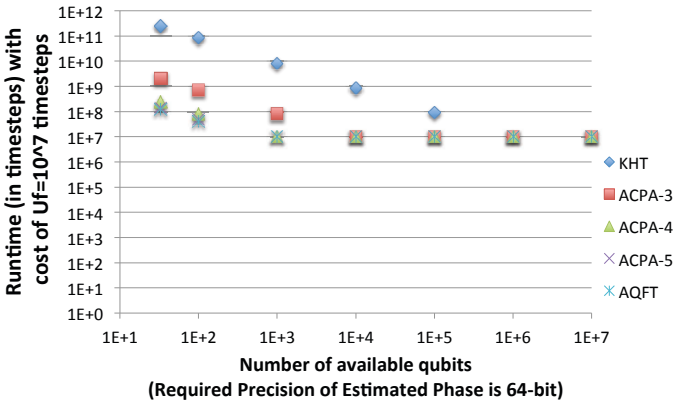

Fig. 11: Space-time tradeoffs with U_f and desired precision set for Shor's algorithm. With fewer available resources, AQFT executes fastest due to fewer invocations of expensive serialized U_f.

3) Resource Constraints with a Larger Oracle Function: We can extend our microbenchmark by increasing the cost of the oracle function from a simple CNOT gate to the oracle for Shor's algorithm and GSE. Figures 11 and 12 show the runtimes for the three methods when the runtime of U_f function is $\sim 10^7$ and $\sim 10^9$ timesteps (order of the size of U_f for Shor's algorithm and GSE, respectively). Desired precision is set for 64 and 8 bits, also according to application. The smaller number of invocations of U_f in the AQFT method allows it to complete execution faster than the other methods. When sufficient number of qubits become available, however, the U_f application can be parallelized, and the truly serial

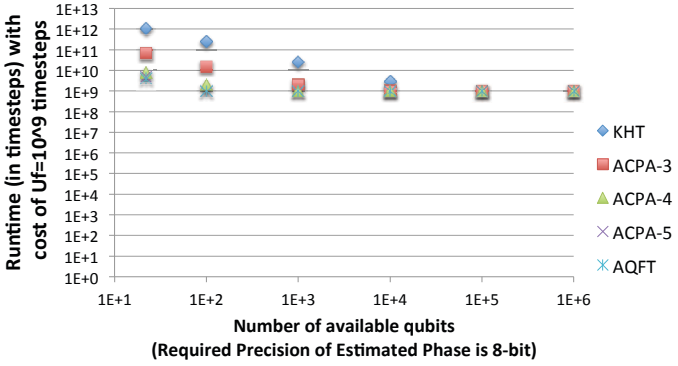

Fig. 12: Space-time tradeoffs with U_f and desired precision set for GSE.

parts of the algorithms (which are the inverse-QFT and the Hadamard Test) give rise to the differences in their runtimes, similar to the trends in Fig. 10.

D. Summary

This section characterized the tradeoffs of different QPE techniques and the design space that they create. While the KHT method is beneficial in terms of parallelism and hence runtime performance with ample resources, it incurs a large cost in terms of qubits and gates. In all metrics, ACPA is a middle solution to the extreme requirements of KHT and AQFT. Ultimately, the amount of resources at hand (especially logical qubits), the level of parallelism supported by the architecture, and the target runtime will determine the method of choice. QPE is a fundamental component of two very important QC applications: Shor's factoring algorithm and GSE. Using the specific characteristics and parameters from these algorithms (see Appendix) our results here have also detailed the tradeoffs regarding how best to implement QPE for these full applications.

VI. Related Work

Efficient decomposition of single-qubit unitaries has been widely studied. The Solovay-Kitaev algorithm [16] is traditionally well-known for this purpose, and an early implementation by Dawson and Nielsen [10] outputs a sequence of order $log^{3.97}(\frac{1}{\epsilon})$ gates for every rotation approximated with ϵ precision. Selinger [26] proposes an improvement which decomposes Z-axis rotations into a sequence of length $4log_2(\frac{1}{\epsilon})$ in the worst case. Paetznick and Svore [23] include non-determinism in their algorithm to achieve gains in circuit cost, while Bocharov et al. [5] improve this by moving away from the traditional decomposition bases of H, S, CNOT and T gates.

Abrams and Lloyd [1] were the first to notice quantum phase estimation can be used in estimating the ground state energy of a molecule. Improvements on making this method faster have since been proposed [24].

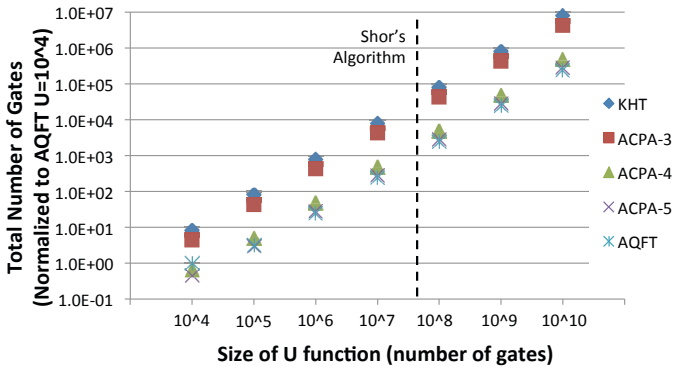

Fig. 13: Comparing the impact of the size of the U_f function on number of total gates using KHT, AQFT and ACPA methods. The size of Shor's U_f function lies around the $5 * 10^7$ line.

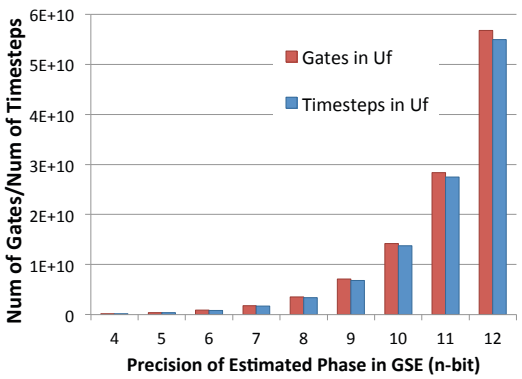

Fig. 14: Obtaining the number of gates and timesteps in the U_f module for the GSE algorithm. The number of gates and timesteps of constituent U's increase exponentially with higher required precision. It can also be seen that little overall parallelism exists in this algorithm.

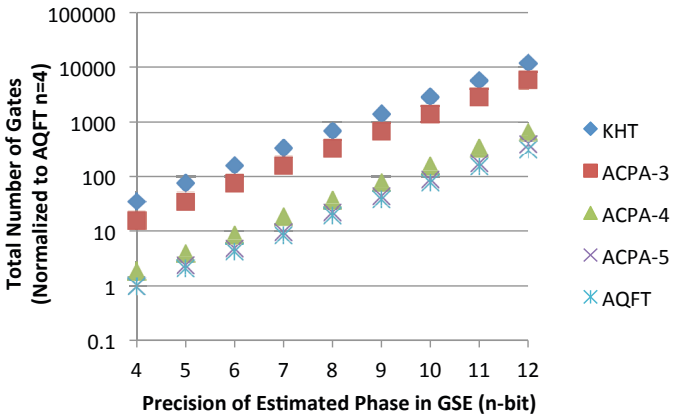

Fig. 15: Comparing the total number of gates of GSE using KHT, AQFT and ACPA methods when the required precision of estimated phase is varied.

Finally, Svore et al. [30] propose an improvement over the KHT method which makes asymptotic improvements on its circuit width and depth, by modifying the Hadamard tests to infer multiple bits of the phase simultaneously instead of one bit at a time. However, this requires extra invocations of the U function, which is a significant cost factor in QPE implementations as per our findings.

VII. Conclusion

This paper has examined the tradeoff of three different methods for quantum phase estimation. These methods on the surface differ in the extent to which they use rotation gates and trials to accomplish the goal of phase estimation, but resource-wise they create a design space which requires different amounts of algorithm runtime and quantum space (or qubit usage). We performed an analytical as well as empirical study of the tradeoffs present in this design space, and discussed the effect of both hardware constraints and the type of algorithm that uses the phase estimation in guiding the choice of a suitable QPE method for different quantum applications. In particular, we find that ACPA provides a good tradeoff between the number of trials and the number of rotations, often achieving a sweet spot in execution time when parallelism is limited by available physical resources.

Appendix

In this appendix, we derive the QPE target precision and oracle size for two algorithms that have quantum phase estimation as a fundamental component: Shor's algorithm for prime factorization, and the Ground State Estimation (GSE) algorithm for finding the ground energy level of a molecule. The fundamental difference between these algorithms (and generally all algorithms involving QPE) is the oracle U function that is used for their implementation.

A. Case Study 1: Shor's Algorithm

Factorizing a large number into its prime constituents is a hard problem for classical computers, and forms the basis of many modern cryptography techniques. A seminal quantum algorithm by Shor [27] offers exponential speedup over the current best known classical algorithm [21]. Quantum phase estimation lies at the heart of Shor's algorithm. We have implemented an optimized phase kickback function U_f for this algorithm, as proposed by Pavlidis and Gizopoulos [25].

In this algorithm, the overall phase kickback module U_f implements a *modular exponentiation* function:

$$|x\rangle|1\rangle \xrightarrow{U_f} |a^x mod N\rangle \tag{11}$$

Referring to our discussion of U and U_f modules in Section IV-D, here the U modules are equivalent to modular multiplication units, while the overall phase kickback module U_f implements a modular exponentiation function. From modular arithmetic it is known that a repeated application of modular multiplication yields modular exponentiation. That is:

$$a^x mod N = (a^{2^0} mod N)^{x_0} + \ldots + (a^{2^{n-1}} mod N)^{x_{n-1}}, \tag{12}$$

where $x_0, \ldots, x_{n-1}$ are the bits of x, the upper register of phase estimation.

Our implementation of this algorithm is efficient in the sense that there is a polynomial cost associated with the phase kickback module U_f as a function of n, because each of the U^{2^k} modules is customized separately and does not result from exponentially many calls to the U module. Analytical analysis yields $n * (796 n_{LR}^2 + 692 n_{LR})$ number of gates and $n*(1045 n_{LR} - 38)$ number of timesteps for the phase kickback circuit [25], where $n_{LR} = n/2$ is the size of the lower register

of phase estimation. These values can be used in the analytical model of Table II to yield the overall cost of implementing Shor's algorithm with the three QPE methods.

Fig. 13 shows this cost as a function of the size of the phase kickback module U_f. A particular problem size (factoring a 32-bit number with 64 bits of precision) has been marked on the graph.

B. Case Study 2: The Ground State Estimation Algorithm

The ground state estimation algorithm is used for estimating the ground state energy E_0 of a molecule [32]. Knowing the energy to lie in an interval $[E_{min}, E_{max}]$, this can be reduced to estimating the phase φ in the equation

$$U|\psi_0\rangle = e^{i2\pi\varphi}|\psi_0\rangle, \tag{13}$$

with U being dependent on the system Hamiltonian H, among other things:

$$U = e^{iE_{max}\tau}e^{iH\tau}. \tag{14}$$

Due to the complexity of implementing the U^{2^k} subcircuits, the subcircuit U is approximated by an $ApproxU$ module. We implement the algorithm for a molecule of molecular weight 10, with up to 12 bits of precision for the energy estimate, and with simplified assumptions for φ. It is difficult to obtain a closed-form analytical expression for the number of gates and timesteps in the phase kickback module. However, from our algorithm implementation we can obtain empirical values for these numbers with varying precision. These values are plotted in Fig. 14. This implementation, unlike that of Shor's, does not result in a polynomial function for the size of the phase kickback module. The lack of customized and efficient specifications of U^{2^k} circuits has resulted in exponential growth from repetitive application of the U module, as mentioned in Eqn. 10.

Fig. 15 shows the total number of gates as a result of varying the required precision in the overall GSE algorithm, showing the exponential growth effect.

ACKNOWLEDGMENTS

The authors thank Pawel Wocjan for useful discussions. This work was partly supported by NSF grant PHY-1415537.

REFERENCES

[1] D. S. Abrams and S. Lloyd. Quantum Algorithm Providing Exponential Speed Increase for Finding Eigenvalues and Eigenvectors. *Physical Review Letters*, 83(24):5162, 1999.

[2] D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the 29th annual ACM Symposium on Theory of Computing*, STOC '97, 1997.

[3] H. Ahmadi and C.-F. Chiang. Quantum Phase Estimation with Arbitrary Constant-Precision Phase Shift Operators. *Quantum Info. Comput.*, 12(9-10):864–875, Sept. 2012.

[4] A. Barenco, A. Ekert, K.-A. Suominen, and P. Törmä. Approximate Quantum Fourier Transform and Decoherence. *Physics Rev A.*, 54(1):139–146, Jul 1996.

[5] A. Bocharov, Y. Gurevich, and K. M. Svore. Efficient Decomposition of Single-Qubit Gates into V Basis Circuits. *Physical Review A*, 88(1):012313, 2013.

[6] P. O. Boykin et al. On Universal and Fault-Tolerant Quantum Computing: A Novel Basis and a New Constructive Proof of Universality for Shor's Basis. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 486–494. IEEE, 1999.

[7] D. Cheung. Improved Bounds for the Approximate QFT. In *Proceedings of the Winter International Synposium on Information and Communication Technologies*, WISICT '04, pages 1–6. Trinity College Dublin, 2004.

[8] C.-F. Chiang. Selecting Efficient Phase Estimation with Constant-Precision Phase Shift Operators. *Quantum Information Processing*, 13(2):415–428, 2014.

[9] A. M. Childs et al. Exponential Algorithmic Speedup by a Quantum Walk. In *Symposium on Theory of Computing*. ACM, 2003.

[10] C. M. Dawson and M. A. Nielsen. The Solovay-Kitaev algorithm. *Quantum Info. Comput.*, 2006.

[11] L. K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Symposium on Theory of Computing*. ACM, 1996.

[12] A. W. Harrow et al. Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters*, 103(15):150502, 2009.

[13] A. JavadiAbhari et al. Scaffold: Quantum Programming Language. Technical report, Princeton University, NJ, USA, 2012.

[14] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi. Scaffcc: A Framework for Compilation and Analysis of Quantum Computing Programs. In *ACM Conference on Computing Frontiers*, 2014.

[15] A. Y. Kitaev. Quantum Measurements and the Abelian Stabilizer Problem. *arXiv preprint quant-ph/9511026*, 1995.

[16] A. Y. Kitaev. Quantum Computations: Algorithms and Error Correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997.

[17] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, 2002.

[18] V. Kliuchnikov, D. Maslov, and M. Mosca. Practical Approximation of Single-Qubit Unitaries by Single-Qubit Quantum Clifford and T Circuits. *arXiv preprint arXiv:1212.6964*, 2012.

[19] V. Kliuchnikov, D. Maslov, and M. Mosca. Asymptotically Optimal Approximation of Single Qubit Unitaries by Clifford and T Circuits Using a Constant Number of Ancillary Qubits. *Physical review letters*, 110(19):190502, 2013.

[20] B. P. Lanyon, J. D. Whitfield, G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, et al. Towards Quantum Chemistry on a Quantum Computer. *Nature Chemistry*, 2(2):106–111, 2010.

[21] A. K. Lenstra et al. *The Development of the Number Field Sieve*, volume 1554. Springer, 1993.

[22] M. Oskin et al. A Practical Architecture for Reliable Quantum Computers. *IEEE Computer*, 35(1):79–87, 2002.

[23] A. Paetznick and K. M. Svore. Repeat-Until-Success: Non-Deterministic Decomposition of Single-Qubit Unitaries. *arXiv preprint arXiv:1311.1074*, 2013.

[24] A. Papageorgiou, I. Petras, J. Traub, and C. Zhang. A Fast Algorithm for Approximating the Ground State Energy on a Quantum Computer. *Mathematics of Computation*, 82(284):2293–2304, 2013.

[25] A. Pavlidis and D. Gizopoulos. Fast Quantum Modular Exponentiation Architecture for Shor's Factoring Algorithm. *Quantum Information and Computation*, 14:0649–0682, 2014.

[26] P. Selinger. Efficient Clifford+T Approximation of Single-Qubit Operators. *Quantum Info. Comput. (arXiv preprint arXiv:1212.6253)*, 2014.

[27] P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Foundations of Computer Science*. IEEE, 1994.

[28] Sqct: Single Qubit Circuit Toolkit - https://code.google.com/p/sqct/, May 2014.

[29] A. Steane. Multiple-Particle Interference and Quantum Error Correction. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, 1996.

[30] K. M. Svore, M. B. Hastings, and M. Freedman. Faster Phase Estimation. *Quantum Info. Comput.*, 14(3-4):306–328, Mar. 2014.

[31] K. Temme, T. Osborne, K. Vollbrecht, D. Poulin, and F. Verstraete. Quantum metropolis sampling. *Nature*, 471(7336):87–90, 2011.

[32] J. D. Whitfield et al. Simulation of Electronic Structure Hamiltonians Using Quantum Computers. *Molecular Physics*, 109(5):735, 2010.