QuRE: The Quantum Resource Estimator Toolbox

Martin Suchara
and John Kubiatowicz
Computer Science Division
Univ. of California Berkeley
Berkeley, CA 94720, U.S.A.
{msuchara, kubitron}@cs.berkeley.edu

Arvin Faruque
and Frederic T. Chong
Computer Science Department
Univ. of California Santa Barbara
Santa Barbara, CA 93106, U.S.A.
{afaruque, chong}@cs.ucsb.edu

Ching-Yi Lai
and Gerardo Paz
Dept. of Electrical Eng. Systems
University of Southern California
Los Angeles, CA 90089, U.S.A.
{laiching, pazsilva}@usc.edu

***Abstract*—We describe QuRE, the *Quantum Resource Estimator*. QuRE is a layout estimation tool that estimates the cost of practical implementations of quantum circuits in a variety of competing physical quantum technologies and with a variety of strategies for fault tolerant encoding. For each specified algorithm, QuRE estimates quantities such as number of physical qubits, execution time, probability of success of the computation, and physical gate counts for elementary quantum gate types of a specified technology. Out of the box, QuRE supports estimation for six physical quantum technologies, seven quantum algorithms, and with error correction using the Steane [1], [2], Bacon-Shor [3], Knill [4] or surface [5], [6] error correction codes. Moreover, QuRE is extendable and can easily accommodate other choices. After describing QuRE, we use it to investigate the tradeoff between concatenated and surface error correction coding techniques, demonstrating the existence of a crossover point for the Ground State Estimation Algorithm [7].**

Keywords*—*Quantum Computation, Resource Estimation.

I. Introduction

Quantum computing promises a revolution in our ability to solve extremely difficult problems, such as factorization [8] and quantum mechanical simulation [7]. Although the construction of a practical quantum computer is hindered by numerous factors, we believe that the design space for a future quantum computer should be explored *now*, helping to sort out the plethora of proposed technologies, fault tolerance techniques, and other implementation choices. Since quantum computing algorithms are often expressed via the *quantum circuit model* [9], investigating the practicality of implementing a particular quantum algorithm in a particular technology boils down to investigating the cost and reliability of a partitioned, placed, and fault-tolerant version of the quantum circuit.

Such *complete instantiations* of quantum circuits are much more challenging to investigate than classical circuits for many reasons. One is the impracticality of simulating the actual *values* flowing through the circuits which have an exponential amount of state. Instead, one must simulate the errors flowing through the circuit. Another challenge arises from the unreliability of all current quantum computing technologies: quantum circuits must be *encoded* in a *quantum error correction* code (QECC), where individual quantum values never leave the code space – even when operated upon. These encoded versions of the circuit are typically much larger than the original non-encoded circuit and, worse, the level and placement of encoding depends on the length of the computation.

TABLE I. Logical gate counts and parallelization factors for the Ground State Estimation Algorithm [7].

Gate	Count	Parallelism
CNOT	7.64×10^{10}	1.5
H	3.64×10^{10}	6
$P_{\|0\rangle}$	55	55
$\mathcal{M}_Z$	5	1
Z	1.21×10^{10}	3
S	1.21×10^{10}	3
Rot	6.46×10^{9}	1.5

Consequently, as with classical *computer aided design* techniques, investigation of the final behavior of a quantum computing circuit can span the gamut from extremely detailed encoding, layout, scheduling, and simulation to high-level *estimates* of the rough behavior based on aggregate or statistical models. This spectrum of investigative tools provides a speed vs accuracy trade-off, with different tools appropriate to different stages of the design process. Since quantum circuits can have 10^{15} to 10^{22} gates [10], we clearly need at least one tool that can provide quick but accurate estimates.

In this paper, we describe QuRE, the *Quantum Resource Estimator*[1]. QuRE is an early-stage *estimation* tool that estimates the cost of a practical implementation of various quantum circuits in a variety of competing physical quantum technologies and with a variety of strategies for fault tolerant encoding. QuRE starts with a statistical description of a particular quantum algorithm (the "logical circuit"), such as shown in Table I. It then estimates the cost of implementing this circuit in a given technology using a given QECC encoding.

Much of the complexity of QuRE arises in accounting for the implementation of a fault tolerant version of the circuit, given the length of the circuit and reliability of the target

This work was supported by the IARPA QCS program (document numbers D11PC20165 and D11PC20167). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government. Martin Suchara is currently at IBM Research.

[1]QuRE is pronounced like "query."

technology. For instance, the process of encoding the logical circuit may involve addition of a number of *ancilla* states to help with the encoding process. Such ancilla states may be produced by *ancilla factories*, structures whose parameters must be carefully analyzed.

QuRE produces detailed outputs that go beyond reporting basic quantities such as the running time, number of physical qubits, and gate or instruction counts. For example, we also report the size of the ancilla factories, number of gates and time needed to prepare each ancillary state, etc. The only inputs that must be provided to the toolbox are a succinct description of the quantum technologies, and information about the algorithm in the format shown in Table I.

A key benefit of the QuRE toolbox is its ability to quickly compare resource estimates in a number of scenarios. The toolbox supports estimation for six physical quantum technologies, seven quantum algorithms, and with error correction using the Steane [1], [2], Bacon-Shor [3], Knill [4] or surface [5], [6] QECCs. Moreover, the toolbox can be extended easily.

The QuRE toolbox can evaluate the effect of various parameters. For example, we observed that with increasing physical error rate, the resource requirements of topological QECCs increase much more moderately than those of concatenated codes. Consequently, for a typical real-world sized quantum algorithm, the surface code requires fewer resources than the Steane code when the physical error rate is above approximately 1×10^{-9}, but the Steane code performs better below this error rate. This allows us to determine which QECC should be used. We also observed how the composition of the quantum gates differs based on the choice of the code, a necessary first step for optimizing the most frequently occurring operations in the hardware.

The main contributions of this work are as follows:

- Development of a comprehensive resource estimation methodology for a variety of quantum technologies, algorithms, and QECCs.
- Release of the QuRE software toolbox that automatically estimates a range of quantities, including number of qubits, time to completion, success probability, and gate counts for each physical gate type.
- Providing framework for performance comparisons of current or future physical quantum technologies, algorithms, and QECCs.

The paper is organized as follows. Section II provides a high-level overview of the functionality of the QuRE toolbox. We describe how the toolbox estimates the cost of error correction with concatenated QECCs in Section III. In Section IV we repeat the error correction overhead analysis for topological codes. Finally in Section V we illustrate the functionality of the QuRE toolbox by reporting the resource requirements of the ground state estimation algorithm, and show a crossover point where the choice of the optimal QECC depends on the properties of the physical architecture.

Due to space constraints, this paper only presents the highlights of the methodology. More details, including descriptions of all studied algorithms, physical technologies, and all numeric results appear as a separate technical report [10].

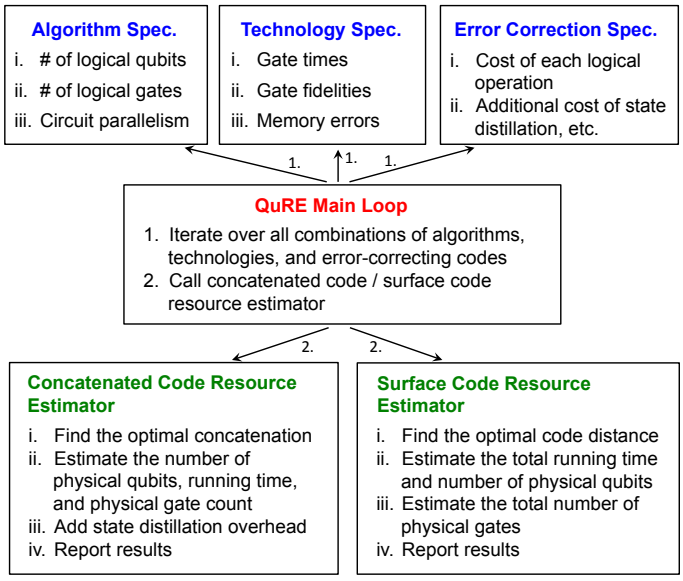

Fig. 1. A schematic view of the key modules of the QuRE toolbox.

II. Functionality of the QuRE Toolbox

QuRE is implemented in Octave [11] and uses modular design to allow easy extendability. Figure 1 shows a schematic view of the major modules of QuRE. At the heart of the tool is the **Main Loop**, which iterates over all specified quantum algorithms, quantum technologies, and quantum error correction codes. For each combination, the Main Loop calls appropriate modules that load information about the algorithm, technology and error correction code.

An **Algorithm Specification** module provides information about the number of logical qubits the particular algorithm needs. *Logical* qubits are defined as the fault-tolerant qubits built of a greater number of unreliable physical qubits. Number of logical gates and circuit parallelism are also specified. Section V-A illustrates how these specifications were obtained for the ground state estimation algorithm, one of the algorithms preloaded in the toolbox.

A **Technology Specification** module describes properties of a particular physical quantum technology. It specifies the time needed to carry out each physical gate, the error of the worst gate, and information about memory error rate per unit time. Note that by *physical* gate we shall understand a non-fault-tolerant quantum gate that is provided by the technology by executing one or more instructions. More details about three of the quantum technologies preloaded into QuRE are in Section V-B.

An **Error Correction Specification** module is provided for each analyzed error correction code. It quantifies the time and the number of physical (unreliable) gates needed to implement a logical (fault-tolerant) gate of each type. The module also quantifies any additional overhead that may be caused e. g. by magic state distillation, ancilla preparation, or any other operations pertinent to the particular error correction code. The methodology used to analyze the Steane and surface codes is described in Sections III and IV, respectively.

The **Concatenated Code Resource Estimator** reports the resources needed by an algorithm, technology, and concatenated code loaded by the Main Loop. First, the module determines the minimum concatenation level that is sufficient to complete the algorithm successfully with high probability. Then it evaluates the resources needed to carry out fault-tolerant operations at that concatenation level, multiplies them by the number of operations used by the algorithm, and evaluates additional resources needed for magic state distillation.

The **Surface Code Resource Estimator** reports the resources needed by the surface code and works analogously to the Concatenated Code Resource Estimator.

III. Concatenated Code Overhead

Here we describe how QuRE estimates the cost of error correction with concatenated codes. We use the Steane code [1], [2] as a representative. The techniques for evaluating other concatenated codes are similar. In Subsection III-A we describe the cost of constructing reliable gates out of unreliable components. Subsection III-B describes the tiled qubit layout that places qubits in two dimensions. QuRE uses tiles as basic building blocks that are concatenated in a recursive fashion. To quantify the total number of physical qubits, QuRE determines an appropriate concatenation level, as shown in Subsection III-C. Finally, in Subsection III-D we describe how QuRE quantifies the cost of elementary operations.

A. The Steane Code

To achieve fault-tolerance, the Steane code uses seven unreliable physical qubits to encode a single logical qubit. To increase the strength of the error correction, the Steane code can be concatenated, each level m encoded qubit block is built using seven level $m-1$ qubits. Level 1 blocks are built of seven level 0 physical qubits.

Quantum circuits consist of quantum gates of various types. For an overview see e.g. [9]. QuRE must estimate the cost of each fault-tolerant gate type. We will denote fault-tolerant quantum gates at the m-th level of concatenation by appending the concatenation level: $X_{(m)}$, $Y_{(m)}$, $Z_{(m)}$, $CNOT_{(m)}$, $H_{(m)}$, $S_{(m)}$, $T_{(m)}$. Note that at the lowest level $m = 0$ the gates correspond to the instructions of the quantum computer. Quantum states can be also measured in the X and Z basis, and the measurement will be denoted by $\mathcal{M}_{X(m)}$ and $\mathcal{M}_{Z(m)}$ or the symbol of a measurement. Qubit initialization is denoted by $P_{|0\rangle(m)}$ and $P_{|+\rangle(m)}$. The symbol $\mathcal{EC}_{(m)}$ represents error correction that repairs encoded state.

Most operations on states encoded with the Steane code are transversal, meaning that an operation U can be performed by recursively applying operation U on each of the seven sub-blocks. This is illustrated in Figure 2. Transversal gates include Pauli, S, H, $CNOT$ (controlled not) gates, and measurements.

In order to have a complete operation set, concatenated codes need at least one non-transversal operation. The Steane code uses the non-transversal T gate [12] in Figure 3. Non-transversal gates are expensive and difficult to analyze. The QuRE toolbox uses a system of recursive equations to accurately express the cost of such gates.

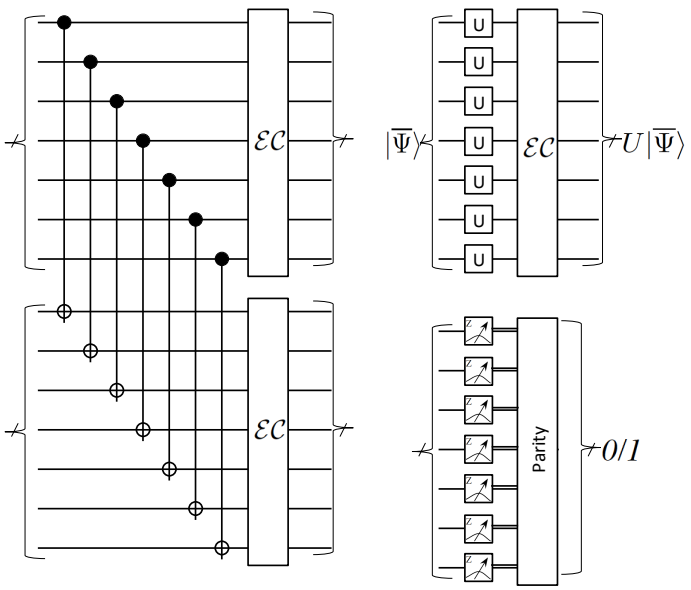

Fig. 2. Fault-tolerant implementation of the $CNOT$ gate (on the left), the single qubit gates X, Y, Z and S (top right corner), and Z basis measurement (bottom right).

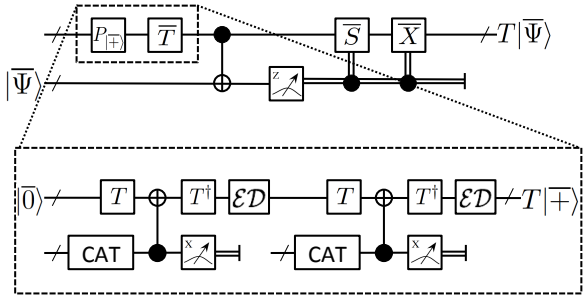

Fig. 3. Example of a complicated gate – fault-tolerant T gate cannot be implemented transversally.

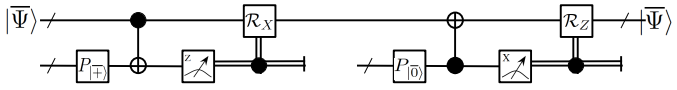

Fig. 4. Illustration of the Steane-EC syndrome extraction method.

The error correction operation $\mathcal{EC}$ must be performed after applying each gate, as well as periodically on idle qubits. The error correction operation [2], [13] is shown in Figure 4. Bit flip errors (X operation) are corrected by first preparing an ancilla in the $|+\rangle$ state, applying a controlled not gate between the data and ancilla, and measuring the ancilla in the Z basis. Depending on the outcome of the measurement, the data is either correct, or the X operation is applied. This is denoted by $\mathcal{R}_X$. Phase shifts (Z operation) are corrected similarly.

B. The Tiled Qubit Layout

Since the two-qubit $CNOT$ gates can be only applied to physically adjacent qubits, QuRE must take the cost of communication into account. QuRE assumes that the state of one of the two interacting qubits is transferred using a

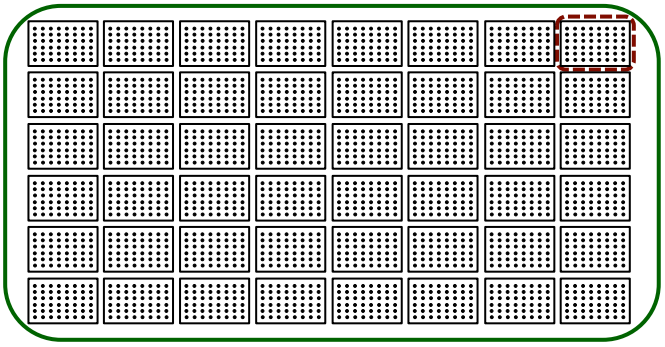

Fig. 5. One tile at the second concatenation level (bounded by the large solid box). A tile at the first level of concatenation consists of the qubits inside the smaller dashed box.

sequence of $SWAP$ operations before applying the $CNOT$. Moreover, additional "dummy" qubits must be introduced because two data qubits cannot be swapped directly. Svore et al. [14] describe an optimized qubit layout and a sequence of operations that attempt to minimize the overhead due to communication. We use this optimized layout in QuRE.

QuRE stores each fault-tolerant logical qubit in a two-dimensional tile. The tile contains additional space to store an ancilla, verification qubits to verify success of some state preparations, and "dummy" qubits to facilitate communication. Communication between tiles is achieved by swapping. For example, to perform a $CNOT$ operation on two neighboring tiles, the data qubits from the second tile are moved into the first one, the $CNOT$ operation is performed, and then the qubits return to their original locations. The tile consists of 6×8 smaller tiles at the next lower concatenation level. An example at the second concatenation level is show in Figure 5. It follows that an algorithm with n logical qubits requires $n(6 \times 8)^l$ physical qubits with l levels of concatenation.

C. Finding the Optimal Concatenation Level

To quantify the number of qubits and cost of all operations, QuRE must find the minimum concatenation level l sufficient to correct errors with high probability. QuRE follows the method from [9]. Let p be the failure probability of a physical gate. The probability that a circuit introduces two errors, which is an unrecoverable error, is $O(p^2) = cp^2$. Here $c = 1/p_{th} = 1/3.61 \times 10^{-5}$ corresponds to the threshold of the code [14]. With l levels of concatenation the failure probability becomes $(cp)^{2^l}/c$. With N gates and error probability at most $\epsilon = 0.5$, each gate must be accurate to ϵ/N so it suffices to find l satisfying: $\frac{(cp)^{2^l}}{c} \leq \frac{\epsilon}{N}$.

D. Quantifying Resources with m Levels of Concatenation

The QuRE toolbox estimates the number of physical gates and the time needed to perform each logical operation at level m of concatenation and sums up the costs. Because concatenated codes have a hierarchical structure, it is convenient to use recursion to express the gate counts and timing. The QuRE toolbox follows the optimized movement strategy described by Svore et al. [14] to count the number of $SWAP$ operations needed to move qubits during computation. Note that it is necessary to distinguish "horizontal" and "vertical" $CNOT$ and $SWAP$ operations which act on encoded blocks of qubits that are adjacent along the horizontal and vertical axes, respectively. These operations denoted $hCNOT_{(m)}$, $vCNOT_{(m)}$, $hSWAP_{(m)}$, and $vSWAP_{(m)}$ all have different costs for $m \geq 1$.

Here we only express the cost of the one-qubit transversal operation X and the number of gates in the error correction block. We refer readers interested in the cost of all operations calculated by QuRE to our technical report [10].

Fault-Tolerant Pauli Gates and Hadamard: The X gate is implemented by applying the X operation transversally to the seven data blocks in the tile, followed by error correction:

$$ops(X_{(m)}) = 7X_{(m-1)} + \mathcal{EC}_{(m)} \tag{1}$$

The X operation can be applied to the seven data blocks in parallel, thus the time to perform the operation is:

$$time(X_{(m)}) = X_{(m-1)} + \mathcal{EC}_{(m)} \tag{2}$$

Error Correction: QuRE obtains the cost of the fault-tolerant error correction by counting the operations in the circuit in Figure 4. A number of blocks in the tile must be moved to perform the $CNOT$ gates, which can only act on adjacent blocks. This causes the relatively high number of $SWAP$ operations in the estimate. Fault-tolerant state preparations of the ancillas also require additional gates, such as Hadamards, $CNOT$s and measurements, not shown in Figure 4. To simplify the already complicated model, QuRE assumes that ancilla preparation always succeeds.

$$\begin{aligned} ops(\mathcal{EC}_{(m)}) = {} & 14hCNOT_{(m-1)} + 28vCNOT_{(m-1)} \\ & + 7H_{(m-1)} + 18hSWAP_{(m-1)} + 15vSWAP_{(m-1)} \\ & + 8P_{|+\rangle(m-1)} + 12P_{|0\rangle(m-1)} + 20\mathcal{M}_{X(m-1)} \end{aligned} \tag{3}$$

IV. Topological Code Overhead

Here we describe the calculation performed by QuRE to estimate the cost of error correction with the surface code, the most commonly studied representative of topological error correction codes.

Subsection IV-A describes how the surface code performs error correction and how QuRE estimates its cost. In Subsection IV-B we introduce a tiled qubit layout that helps QuRE accurately estimate the number of physical qubits, and in Subsection IV-C we discuss how QuRE determines an appropriate tile size. Subsection IV-D explains how to perform the controlled not operation, and Subsection IV-E illustrates how QuRE quantifies the number of gates and time needed to perform the controlled not operation.

A. The Surface Code

We included the surface code [6] in the QuRE toolbox because it can tolerate higher error rates than concatenated codes. Another of its advantages is locality – all operations can be performed locally without the need to move qubits. Comparison with concatenated codes is therefore interesting.

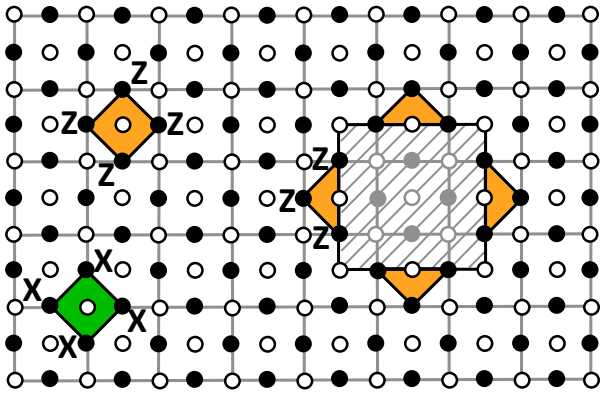

Fig. 6. Lattice of the surface code. A few syndromes and a rough hole are shown.

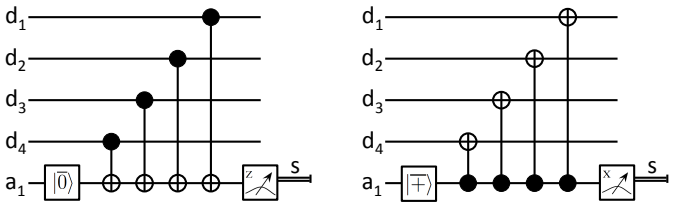

Fig. 7. The syndrome measurement circuits. Initialization, four CNOTs and a measurement are used to obtain each syndrome.

QuRE assumes that the surface code places qubits on a regular grid shown in Figure 6. Black circles represent data qubits that encode the quantum state, and white qubits represent ancillas that are used for error syndrome extraction.

There are two types of syndromes, plaquette and site syndromes, obtained by measuring four qubits in the Z and X bases, respectively. A few of the syndromes are depicted in Figure 6. Syndrome measurement is the most expensive operation because it is performed continuously. The syndromes are extracted using the circuits in Figure 7 which each require one ancilla. There are enough ancillas in Figure 6 to allow simultaneous measurement of all syndromes.

By default, the surface code only encodes two logical qubits [6]. To increase the number of encoded qubits, we need to introduce holes into the surface. Figure 6 shows an example of a single "rough" hole. Notice that the plaquette stabilizers at the boundary of the hole were deformed into weight three ZZZ stabilizers. "Smooth" holes can be introduced as well, they are shifted by half a cell size, and are surrounded by XXX stabilizers. One additional logical qubit is encoded by introducing a *pair* of smooth or rough holes.

B. The Tiled Qubit Layout

To count the number of physical qubits, we introduce a tiled qubit layout. Each tile contains a pair of rough holes, and therefore represents one logical qubit. This is shown in Figure 8. The logical Z operation is performed by applying the Z gate to a chain of physical qubits connecting the pair of holes, and the logical X operation is performed by applying X gates to a chain encircling one of the holes in the pair. Therefore, to prevent errors due to accidental application of these operations, the holes must be sufficiently big and far apart. Error correction improves with increasing tile size.

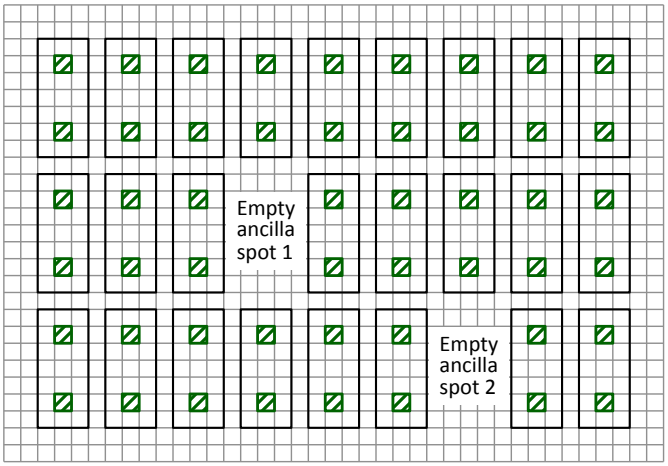

Fig. 8. The tiled layout of the surface code. Each tile contains two rough holes (dashed fill) that represent a logical qubit.

QuRE estimates the number of tiles as follows. One tile is needed for each logical qubit in the quantum algorithm. In addition, k pairs of logical ancillas ($2k$ tiles) are required for parallel execution of k $CNOT$ gates. Figure 8 shows one pair of such ancillas. Additional space is also needed to distill and store magic states which are required to implement S and T quantum gates [15]. We describe the estimation of this additional space in [10].

C. Finding the Optimal Code Distance

We need the tiles to be sufficiently large to guarantee success of the computation with constant probability (say 50%). QuRE uses the method of Jones [16] to find an appropriate code distance d, the minimum distance of two holes at any time during the computation, as well as a lower bound on the circumference of all holes. To allow enough space for the controlled not operation, as described later, the QuRE toolbox assumes that the size of each square in Figure 8 is $d \times d$.

Jones [16] estimates d by solving $\frac{0.5}{N} \geq C_1 \left(C_2 \frac{p}{p_{th}}\right)^{\lfloor \frac{d+1}{2} \rfloor}$ for d where $0.5/N$ is the desired success probability divided by the number of logical gates in the algorithm, p is the physical gate error rate, $p_{th} \approx 0.01$ is the threshold of the surface code, and $C_1 \approx 0.13$ and $C_2 \approx 0.61$ are constants.

D. The Controlled Not Operation

Interestingly, the surface code does not need any $SWAP$ operations to perform a $CNOT$ gate between logical qubits, even if they are far apart. $CNOT$ operations can be done easily between smooth and rough hole pairs by a braiding procedure where one of the smooth holes is "grown" and "shrunk" to move around a rough hole in the other hole pair [17]. This operation is illustrated in Figure 9. Since our tiled layout only contains rough holes, we need to estimate the cost of a rough-rough $CNOT$ between two rough hole pairs. This operation is a bit more complicated and requires additional ancilla space to

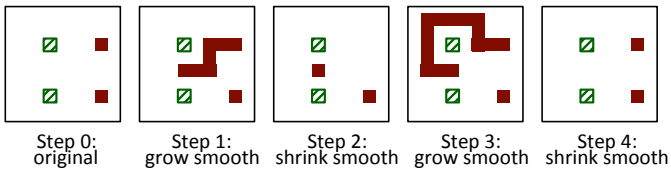

Fig. 9. A smooth-rough $CNOT$ gate. The time evolution of the rough holes (dashed fill) and smooth holes (solid fill) is shown.

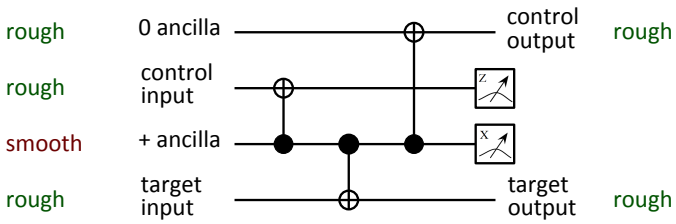

Fig. 10. A rough-rough $CNOT$ gate uses three smooth-rough CNOTs.

perform rough to smooth qubit conversion. This is illustrated in Figure 10. First the two ancillas are initialized, and then three $CNOT$s are performed by braiding.

E. Quantifying the Resources

The QuRE toolbox estimates resources in three key steps. First the number of physical qubits is calculated. Second, the actual time needed to run the entire algorithm is determined. Third, the number of physical gates is estimated.

Physical qubits are calculated as follows. First, we obtain the number of tiles by adding the number of logical qubits, ancillas to support $CNOT$ operations, and additional space for magic state distillation. We multiply this number by the number of physical qubits needed to build a single tile.

The total running time is obtained by adding the cost of all elementary logical operations. Here we derive the running time of the logical $CNOT$. For other operations see [10].

To estimate the running time of $CNOT$, we first need to consider more elementary operations. To perform error correction ($\mathcal{EC}$), all syndromes must be measured d times. The syndromes can be measured in parallel using the circuit in Figure 7, and therefore the time to perform error correction is:

$$time(\mathcal{EC}) = (max(P_{|0\rangle(0)} + \mathcal{M}_{Z(0)}, P_{|+\rangle(0)} + \mathcal{M}_{X(0)}) + 4CNOT_{(0)}) \times d. \quad (4)$$

The $CNOT$ gate between a smooth and rough qubit can be performed in time $4 \times time(\mathcal{EC})$ by the braiding procedure shown in Figure 9 because the error correction operation must be applied after each hole expansion or contraction. A rough logical qubit can be measured in the Z basis and a smooth qubit can be measured in the X basis in time $\mathcal{M}_{X(0)} + \mathcal{EC}$.

Finally, the time needed to perform a rough-rough $CNOT$ shown in Figure 10 must be the sum of the time of three smooth-rough $CNOT$s and the two measurements done in parallel (the ancilla state preparations are performed offline):

$$time(CNOT) = 3 \times 4 \times \mathcal{EC} + \mathcal{M}_{X(0)} + \mathcal{EC}. \quad (5)$$

QuRE obtains the total physical gate count in two steps. First, an estimate is obtained by calculating the number of gates needed to continuously perform the error correction operation in the entire surface of the code for the time period needed to run the algorithm. This is done by multiplying three terms: the number of error correction cycles, the number of elementary cells in the surface, and the number of operations needed to error correct an elementary cell. The last term simply consists of the gates in the two syndrome extraction circuits of figure 7: $P_{|0\rangle(0)} + P_{|+\rangle(0)} + 8CNOT + \mathcal{M}_{X(0)} + \mathcal{M}_{Z(0)}$.

In the final step, a refinement is performed by including the small number of additional gates needed by logical operations.

V. Quantifying and Interpreting the Resource Requirements

After introducing the ground state estimation algorithm in Subsection V-A, we describe three realistic models of physical technologies that are built into QuRE. Estimating the resource requirements with the Steane and surface error correcting codes leads to surprising conclusions. In Subsection V-C we show that fewer qubits are needed when the surface code is used to correct errors in unreliable technologies, but the Steane code outperforms the surface code when the physical error rate drops below certain threshold. Subsection V-C also illustrates the outputs of the QuRE toolbox.

A. The Ground State Estimation Algorithm

The ground state estimation algorithm [7] finds the ground state energy of a molecule. It is a popular benchmark, and is one of the algorithms preloaded in QuRE. The logical gate counts in the quantum circuit are studied in [18] as a function of the bit precision b of the result and the number of wave functions M that describe the ground state. In this work, we chose a problem instance that finds the ground state energy for the glycine molecule with $b = 5$ bits of precision. The glycine molecule needs $M = 50$ wave functions (10 wave functions are needed for each C, O, N atom and 2 for each H).

The algorithm needs 55 logical qubits. The required number of logical gates was already summarized in Table I. Note that the table shows the *logical* gates without error correction overhead, which is later estimated by QuRE. The parallelization factor indicates how many of the gates of a particular type can be performed in parallel. For example, we need 3.64×10^{10} H gates and 6 of these gates can be scheduled simultaneously on average. Rot denotes arbitrary one qubit rotations. QuRE decomposes these rotations into more elementary gates (H and T) using the method of Bocharov et al. [19]. The cost of this decomposition is significant, one rotation typically decomposes into 10^3 to 10^4 gates.

B. Models of the Physical Quantum Computer

Properties of quantum technologies and quantum control protocols were studied by Hocker et al. [20]. QuRE includes all twelve technologies from [20], and here we introduce three of them. These three technologies are among the most

TABLE II. THE GATE TIMES (IN NS) FOR ALL SUPPORTED GATES.

Gate	Superconductors	Ion Traps	Neutral Atoms
CNOT	22	120,000	11,370
SWAP	17	10,000	34,120
H	6	6,000	2,991
$P_{\|+\rangle}$	100	16,000	3,991
$P_{\|0\rangle}$	106	10,000	1,000
$\mathcal{M}_X$	16	106,000	82,991
$\mathcal{M}_Z$	10	100,000	80,000
X	10	5,000	2,667
Y	10	5,000	2,667
Z	1	3,000	5,532
S	1	2,000	3,125
T	1	1,000	3,125

TABLE III. GATE AND MEMORY ERRORS.

Error	Superconductors	Ion Traps	Neutral Atoms
Gate	1.00×10^{-5}	3.19×10^{-9}	1.47×10^{-3}
Memory	1.00×10^{-5}	2.52×10^{-12}	$high$

promising candidates suitable for building a large-scale quantum computer, representing a range of properties that a future quantum computer may possess – very fast but error prone superconductors, slower but more reliable ion traps, and neutral atoms with only average speed and average error properties.

Superconductors [21], [22]: The building block for qubits is the Josephson junction. This technology suffers from relatively high errors due to radiation leakage into the Josephson junction, circuit defects, and engineering limitations [20].

Ion Traps [23]: This technology is based on a 2D lattice of ions confined by electromagnetic field. Lasers are applied to implement quantum gates. Errors are caused by intensity fluctuations of the laser, resulting in low gate errors.

Neutral Atoms [24]: Qubits are represented by ultracold atoms trapped by light waves in an optical lattice. The "ultracold" atom properties improve their noise resilience, but errors arise from atomic motion inside the lattice.

The durations of the physical gates are shown in Table II. Another important property of the technologies is reliability. Table III summarizes the error probability after applying the worst gate, as well as the probability of a bit flip per nanosecond on an idle qubit.

C. Numeric Results Obtained with the QuRE Toolbox

Table IV show the resources needed to do ground state estimation with the Steane and surface codes on the three quantum technologies. The table shows the following quantities:

- **Execution time:** the total time in ns needed to obtain the correct result with high probability.
- **No. qubits:** the total number of physical qubits needed to build the quantum computer.
- **No. gates:** the total number of gates executed by the quantum computer.
- **Dominant resource:** the most frequently occurring gate.
- **QECC distance or QECC concatenations:** the distance for the surface code and the number of concatenations for the Steane code.
- **Logical gate error:** the error probability after performing one logical operation on a single logical qubit.
- **Logical gate time:** the time in ns to perform the error-correcting operation on one logical qubit.
- **No. qubits per logical:** the size of a tile that stores one logical qubit.
- **No. gates per logical:** the number of gates required to perform the error-correcting operation on the tile.

The results in Table IV illustrate how the resource requirements increase as the reliability of the technology decreases. The number of physical gates needed to implement a logical gate protected by the surface code ranges from 9.60×10^3 for ion traps, increases to 3.11×10^4 for the less reliable superconducting technology, and 2.40×10^5 for the least reliable neutral atoms technology. The Steane code exhibits an even sharper increase for less reliable technologies. Note that the neutral atoms technology cannot be protected by the Steane code because its error rate is too high.

The sharp increase of the resources needed by the Steane code is caused by the need to increase the concatenation level. The resources required by the surface code increase much more modestly with decreasing reliability, but there is certain minimum overhead even for low error technologies. As a result, the optimal choice of error correction may depend on the properties of the physical technology. For example, the Steane code requires three orders of magnitude fewer physical qubits than the surface code on the ion trap technology, but the cost reverses on the superconducting qubit technology.

To validate our methodology, we compare numeric results with the resource estimation for Shor's algorithm of Jones et al. [16]. Jones et al. assume that to factor a 1024-bit number, Shor's algorithm requires 1.68×10^8 Toffoli gates and $6,144$ logical qubits. They also assume that their quantum computer can execute an elementary gate in 32 ns, and the surface code is used for error correction. Under these assumptions, QuRE estimates that the Shor's algorithm requires 3.21×10^8 physical qubits and the execution time is 1.60 days. This is in excellent agreement with Jones et al. who estimate 4.54×10^8 physical qubits and execution time of 1.81 days.

VI. CONCLUSION

In this paper we describe the QuRE toolbox and illustrate its use. Moreover, we show that the number of physical qubits required to implement an algorithm can vary by *three orders of magnitude* depending on the type of quantum error correction code. Our results show that there is a *crossover point* between concatenated and topological error correction codes, and the optimal choice of error correction code can vary based on technology parameters. There are many more observations that can be made by inspecting the detailed outputs of the QuRE toolbox, which is available for download on our website.

Because QuRE is designed to *quickly* compare the properties of large quantum circuits, it has to use simplifying assumptions. However, QuRE uses a very detailed model of

TABLE IV. RESOURCE ESTIMATES OBTAINED WITH QURE.

Technology	**Neutral Atoms**	**Supercond. Qubits**	**Ion Traps**	
Gate error	$\mathbf{1 \times 10^{-3}}$	$\mathbf{1 \times 10^{-5}}$	$\mathbf{1 \times 10^{-9}}$	
Avg. gate time	**19,000 ns**	**25 ns**	**32,000 ns**	
Execution time	6.21×10^{21}	3.61×10^{18}	6.02×10^{21}	
No. qubits	4.21×10^{8}	5.46×10^{7}	2.45×10^{8}	
No. gates	6.07×10^{25}	2.82×10^{24}	7.45×10^{24}	
Dominant gate	$CNOT$	$CNOT$	$CNOT$	
QECC distance	25	9	5	**Surface code**
Logical gate error	3.26×10^{-15}	1.10×10^{-17}	9.58×10^{-22}	
Logical gate time	1.29×10^{5}	2.10×10^{2}	5.96×10^{5}	
No. qubits per logical	8.13×10^{4}	1.05×10^{4}	3.21×10^{3}	
No. gates per logical	2.40×10^{5}	3.11×10^{4}	9.60×10^{3}	
Execution time	N/A	1.51×10^{23}	1.55×10^{22}	
No. qubits	N/A	1.40×10^{10}	1.27×10^{5}	
No. gates	N/A	1.04×10^{36}	1.48×10^{25}	
Dominant gate	N/A	$SWAP$	$SWAP$	
QECC concatenations	N/A	5	2	**Steane code**
Logical gate error	N/A	5.22×10^{-23}	1.64×10^{-20}	
Logical gate time	N/A	4.31×10^{8}	4.48×10^{7}	
No. qubits per logical	N/A	2.55×10^{8}	2.30×10^{3}	
No. gates per logical	N/A	1.25×10^{11}	2.21×10^{4}	

error correction, and we are confident in its ability to perform accurate estimates. One potential inaccuracy may arise due to oversimplified descriptions of the quantum algorithms at the input of the tool. More accurate descriptions of the algorithms are not available, and manipulating the original circuits with 10^{15} to 10^{22} gates is beyond the scope of a quick resource estimation tool. An interesting direction for future work would be to use succinct descriptions of quantum algorithms that better capture parallelism and communication.

REFERENCES

[1] A. M. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, pp. 793–797, Jul 1996.

[2] ——, "Efficient fault-tolerant quantum computing," *Nature*, vol. 399, pp. 124–126, May 1999.

[3] D. Bacon, "Operator quantum error-correcting subsystems for self-correcting quantum memories," *Phys. Rev. A*, vol. 73, no. 1, p. 012340, Jan 2006.

[4] E. Knill, "Quantum computing with realistically noisy devices," *Nature*, vol. 434, pp. 39–44, 2005.

[5] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics*, vol. 303, no. 1, p. 2, Jan 2003.

[6] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *J. Math. Phys.*, vol. 43, no. 9, p. 4452, Sep 2002.

[7] J. Whitfield, J. Biamonte, and A. Aspuru-Guzik, "Simulation of electronic structure Hamiltonians using quantum computers," *Molecular Physics: An International Journal at the Interface Between Chemistry and Physics*, vol. 109, no. 5, pp. 735–750, 2011.

[8] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.

[9] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[10] M. Suchara, A. Faruque, C.-Y. Lai, G. Paz, F. Chong, and J. Kubiatowicz, "QuRE: The Quantum Resource Estimation Toolbox," *UC Berkeley Technical Report*, 2013.

[11] "GNU Octave." [Online]. Available: www.gnu.org/software/octave

[12] X. Zhou, D. W. Leung, and I. L. Chuang, "Methodology for quantum logic gate construction," *Phys. Rev. A*, vol. 62, p. 052316, Oct 2000.

[13] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," *ArXiv quant-ph/0504218v3*, 2005.

[14] K. Svore, D. DiVincenzo, and B. Terhal, "Noise Threshold for a Fault-Tolerant Two-Dimensional Lattice Architecture," *arXiv:quant-ph/0604090v3*, 2006.

[15] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal clifford gates and noisy ancillas," *Phys. Rev. A*, vol. 71, p. 022316, 2005.

[16] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, "Layered architecture for quantum computing," *Phys. Rev. X*, vol. 2, p. 031007, Jul 2012.

[17] A. Fowler, A. Stephens, and P. Groszkowski, "High threshold universal quantum computation on the surface code," *Phys. Rev. A*, vol. 80, p. 052312, 2009.

[18] O. Theogarajan, A. Faruque, G. Long, and F. Chong, "Analysis of the Ground State Estimation Algorithm," *IARPA QCS Project report*, 2012.

[19] A. Bocharov and K. Svore, "A Depth-Optimal Canonical Form for Single-qubit Quantum Circuits," *arXiv: quant-ph/1206.3223*, 2012.

[20] D. Hocker, H. Rabitz, Y.-C. Zheng, T. Brun, A. Shafaei, and M. Pedram, "Evaluation of Quantum Control Protocols for Physical Machine Descriptions," *IARPA QCS Project report*, 2012.

[21] A. Blais, J. Gambetta, A. Wallraff, D. I. Schuster, S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf, "Quantum-information processing with circuit quantum electrodynamics," *Phys. Rev. A*, vol. 75, p. 032329, Mar 2007.

[22] J. Martinis, "Quantum-information processing with circuit quantum electrodynamics," *Quant. Inf. Processing*, vol. 8, pp. 81 – 103, Jun 2009.

[23] M. D. Barrett, T. Schaetz, J. Chiaverini, D. Leibfried, J. Britton, W. M. Itano, J. D. Jost, E. Knill, C. Langer, R. Ozeri, and D. Wineland, "Quantum information processing with trapped ions," in *Proceedings of ICAP*, 2004, pp. 350 – 358.

[24] M. Saffman, T. G. Walker, and K. Mølmer, "Quantum information with rydberg atoms," *Rev. Mod. Phys.*, vol. 82, pp. 2313–2363, Aug 2010.