Nat Comput (2010) 9:329–345
DOI 10.1007/s11047-009-9133-0

Algorithms on ensemble quantum computers

P. Oscar Boykin · Tal Mor · Vwani Roychowdhury · Farrokh Vatan

Published online: 30 May 2009

Abstract In ensemble (or bulk) quantum computation, all computations are performed on an *ensemble* of computers rather than on a single computer. Measurements of qubits in an individual computer cannot be performed; instead, only expectation values (over the complete ensemble of computers) can be measured. As a result of this limitation on the model of computation, many algorithms cannot be processed directly on such computers, and must be modified, as the common strategy of delaying the measurements usually does not resolve this *ensemble-measurement problem*. Here we present several new strategies for resolving this problem. Based on these strategies we provide new versions of some of the most important quantum algorithms, versions that are suitable for implementing on ensemble quantum computers, e.g., on liquid NMR quantum computers. These algorithms are Shor's factorization algorithm, Grover's search algorithm (with several marked items), and an algorithm for quantum fault-tolerant computation. The first two algorithms are simply modified using a *randomizing* and a *sorting* strategies. For the last algorithm, we develop a classical-quantum hybrid strategy for removing measurements. We use it to present a novel quantum fault-tolerant scheme. More explicitly, we present schemes for fault-tolerant measurement-free implementation of Toffoli and $\sigma_z^{1/4}$, as these operations cannot be implemented "bitwise", and their standard fault-tolerant implementations require measurement.

P. O. Boykin
Department of Electrical and Computer Engineering, University of Florida,
Gainesville, FL 32611, USA

T. Mor (✉)
Computer Science Department, Technion, Haifa 32000, Israel
e-mail: talmo@cs.technion.ac.il

V. Roychowdhury
Electrical Engineering Department, UCLA, Los Angeles, CA 90095, USA

F. Vatan
Jet Propulsion Laboratory, California Institute of Technology,
4800 Oak Grove Drive, Pasadena, CA 91109, USA

Keywords Quantum algorithms · Ensemble (bulk) quantum computers · NMR · Fault tolerance computing

1 Introduction

Quantum computing is a new type of computing which uses the properties of quantum mechanics to construct fast algorithms to solve several important problems. For example, Shor's quantum algorithm (1997) for factoring large numbers is exponentially faster than any known classical algorithm. Similarly, by utilizing Grover's algorithm (1996), it is possible to search a database of size N in time $O(\sqrt{N})$, compared to $O(N)$ in the classical setting.

Nuclear Magnetic Resonance (NMR) computing, first suggested by Cory et al. (1997), and by Gershenfeld and Chuang (1997), is currently one of the most promising implementations of quantum computing. Several quantum algorithms involving only few qubits have been demonstrated in the laboratory setting (Cory et al. 1997; Gershenfeld and Chuang 1997; Cory et al. 1998; Jones et al. 1998; Nielsen et al. 1998; Vandersypen et al. 2001). In such NMR systems, each molecule is used as a computer. Different qubits in the computer are represented by spins of different nuclei. Many identical molecules (in fact, a macroscopic number) are used in parallel; hence, there is an ensemble of quantum computers. This model is called the ensemble or bulk quantum computation model. In such bulk models, each operation is applied to each computer in the ensemble. Qubits in a single computer cannot be measured, and only expectation values of each particular bit over all the computers can be read out.

The impossibility of performing measurements on the particular qubits of individual computers causes severe limitations on ensemble quantum computation. In particular, for quantum cryptography tasks, ensemble quantum computers appear to be useless. It was generally assumed that a rather simple strategy of delaying measurements can be used to bypass these limitations, in order to enable the implementation of *all* quantum algorithms. However, as we explain in Sect. 2 this assumption was not justified, and the *delaying-the-measurement* strategy usually is insufficient.

We first provide here two novel strategies, the *randomizing* strategy and the *sorting* strategy, that resolve the ensemble-measurement problem in most cases. These strategies are provided along with modest modifications (see Sect. 3) of two important algorithms, Shor's factoring algorithm and Grover's search of several items, to enable processing them on ensemble quantum computers. The modifications—although modest—are important, and furthermore, they have their price in terms of the required space (number of qubits) and time of the algorithms.

Although, in theory, polynomial slowdown can commonly be ignored, in practical quantum computing, where each qubit counts, the price might be extremely high. Specifically, current methods in NMR quantum computing are not scalable in the number of qubits: An exponential scalability problem exists due to working with pseudo-pure state and not with a real-pure state; There are also other, less severe, scalability problems, one is due to the difficulty in addressing specific qubits and another is due to "refocusing"—deleting undesired unitary evolutions that happen all the time in the NMR system. As a result, factoring, even of very small odd numbers such as $35 = 7 \times 5$, might be totally impossible (for NMR quantum computers) in the next 20–30 years, unless a drastically new approach will be used such as algorithmic cooling—see Boykin et al. (2002) and Elias

et al. (2007). We must clarify that it is well known in number theory, that the number 15 is an exception,[1] and indeed the experiment that succeeded to factorize 15 on an NMR quantum computer (Vandersypen et al. 2001), heavily relies on that, in order to (extremely) simplify the experimental setup. Such a simplification will not be possible (Izmerly O, private communication) for factorizing other numbers, such as 35.

Most important, we show in Sect. 4 that all these three strategies mentioned above are *insufficient* for fault tolerant computation, so that a vital modification is required. We develop in Sect. 5 a non-trivial strategy, a hybrid classical-quantum strategy for fault-tolerant computing, that resolves the ensemble-measurement problem for that case.

The understanding we gain in Sect. 5 is actually useful also for improving conventional (non-fault-tolerant) error correction on ensemble computers. See Sect. 6. Note however, that in that case the modification improves the algorithms but is not essential for the algorithms.

A remark: In this paper we restrict ourselves to issues related solely to the *ensemble-measurement problem*, and the results here are vital for bulk computation. However, in addition, the specific results obtained regarding universal and fault tolerant sets of gates might also be important for other implementations of quantum computing devices where delaying measurements is desired, such as the electron-spin-resonance transistor computing device of Vrijen et al. (2000).

2 The measurement in ensemble quantum computation

The measurement process in quantum mechanics can be described simply as follows: To measure the state of a qubit, say $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ in the computation basis $(|0\rangle; |1\rangle)$, one measures the Hermitian operator (the observable)

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

to get the outcome $\lambda_0 = 1$ with probability $|\alpha|^2$ and $\lambda_1 = -1$ with probability $|\beta|^2$. In an NMR ensemble model, the corresponding qubit in every computer is measured simultaneously, resulting in the expectation value, i.e., the outcome of the measurement is a signal of strength proportional to $|\alpha|^2 - |\beta|^2$.

The inability to measure bits in individual computers precludes using measurements as a method for resetting bits. A simple way to reset a bit is to measure it and flip it if the outcome is $|1\rangle$. Since each computer in the ensemble will have a different outcome, this is impossible on an ensemble computer. Algorithmic cooling (Boykin et al. 2002) has been proposed as a novel method for resetting bits in ensemble computing model, generalizing an earlier reversible initialization technique (Schulman and Vazirani 1999), by adding the concept of heat-bath cooling; see Boykin et al. (2002), Fernandez et al. (2004), Schulman et al. (2007) and Elias et al. (2007).

The measurement process lies at the heart of all quantum information processing and computing protocols and algorithms, and hence, needs to be carefully addressed in any proposed implementation scheme. Clearly, when the outcome of a measurement is

[1] We shall not explain this point here in details, but in brief, the reason that 15 is an exception is that the "order" r is then either 2 or 4 for any choice of a, namely $a^r = 1$ (modulu 15); An order that is (for many random choices of a) a power of 2 is extremely rare, and when the order satisfies this property it is easy to factorize the number also on a classical computer.

expected to be the same on each of the computers, the ensemble measurement is as good as the standard (single computer) measurement. Usually, this is not the case. In fact, to the best of our knowledge, the following two protocols cannot be implemented on an ensemble quantum computer due to the measurement issue:

Random number generator (RNG): One can easily create an RNG using a single qubit. To create a binomial probability distribution with parameter p, one prepares a state $\sqrt{p}|0\rangle + \sqrt{1-p}|1\rangle$, and measures in the computational basis to obtain the desired RNG. This, as far as we know, cannot be done on an ensemble quantum computer, where only the expectation value $p\lambda_0 + (1-p)\lambda_1 = p(+1) + (1-p)(-1) = 2p - 1$ can be monitored.

Teleportation: Standard teleportation can easily be performed on a three qubit quantum computer. Strictly speaking however, it cannot be performed on an ensemble quantum computer. This is because a direct Bell-state measurement of the ensemble quantum computer is computationally useless: each computer will yield a random result (of the Bell measurement), and on average the outcome is $(1/2)\lambda_0 + (1/2)\lambda_1$ for each of the two measured qubits; hence, there is no way to decide how to rotate the third qubit in each individual computer. Yet, a "fully-quantum teleportation" of the type suggested in Brassard et al. (1998) can be, and has been (Nielsen et al. 1998), performed on an ensemble quantum computer: in this fully-quantum teleportation, the measurement of an individual computer is never monitored, and a classically-controlled rotation of the third qubit is replaced by a quantum control operation, in which the control qubits dephase before being used.

To better appreciate the ensemble-measurement problem it is instructive to review the basic anatomy of a quantum computer. At a very high level, a quantum algorithm can be described as a set of unitary transformations to be applied to an n-qubit system, followed by a measurement of m of the qubits to obtain a classical m-bit output. The m-bit classical output is usually one of the following:

1. The m-bit output is one of many possible "correct" or "good" answers. Suppose that the classical post-processing (on a regular classical computer) is some many-to-one function that leads to the same final answer, no matter which of the intermediate m-bit answers was found by the quantum computer. [For example, Shor's algorithm yields a number, say c, that satisfies some conditions (with high probability), and once it satisfies these conditions, the order can be calculated from this number c via the continued fractions method. The order is the same for the various intermediate values of c (with high probability).] In ensemble quantum computing, even though all the computers do identical operations, they will have different outcomes after the measurement process, and one cannot get one correct answer by the reading process we described. This case was recognized and resolved in the seminal work of Gershenfeld and Chuang (1997). The most effective scheme is to simply delay (or even avoid, if possible) the measurements, and incorporate the post-measurement processing step into the quantum algorithm, as a controlled operation. Then only the final answer, which is identical on all computers, will be measured.
2. The m-bit output is either the "desired" or "good" solution or it is a spurious or "bad" candidate, and the whole process has to be repeated again. For example, in Shor's factorization algorithm the delayed-measurement process (described in the first item above) yields an integer, which a set of classical operations can process to verify whether it is the correct answer (i.e., the "order" of the input integer) or not. The probability that the output yields the correct answer is such that one is guaranteed to obtain such an output in a few number of repeated executions of the algorithm. As

before, in the case of ensemble quantum computing different computers will yield different results, and the task of identifying the correct answer (the order) needs to be addressed. Adapting the delayed-measurement technique of Gershenfeld and Chuang (1997) to this case means that one should also test quantumly if the resulting order on each computer is indeed the correct order, and then somehow get rid of the bad cases in which the result is a wrong order.

3. The m-bit output is one of many possible "correct" or "good" answers. For example, a database search using Grover's algorithm will return one of the entries satisfying the query. If there are multiple possible query hits, then every time the algorithm is run, it will return any one of the hits with equal probability. Thus, in ensemble quantum computing, even though all the computers do identical operations, they will have different outcomes after the measurement process, and one cannot get one correct answer by the reading process we described, nor by the modification suggested in Gershenfeld and Chuang (1997).

We fully resolve these last two cases in Sects. 3.1 and 3.2 via the randomizing and sorting strategies. Let us briefly describe these two new strategies, sorting and randomizing, that one can adopt to successfully overcome the measurement problem in ensemble quantum computers. This in turn will allow us to point out why these approaches will not work for the fault tolerant computing problem, addressed later on in this paper.

In case # 2 mentioned above the measurement output is processed to determine whether it yields the "desired" outcome or not. After the post-processing operations, the answer is yes or no, and all the computers with the "yes" answer, will have the *same answer*, i.e., the desired order. However, the computers with "no" answers, will have different answers after the post-measurement processing steps. In Sect. 3.1 we solve the problem posed by the interference due to the "bad" candidates by replacing bad results with random data, which will not interfere on average with the reading of the "good" result. (Alternatively, one might be able to control-repeat the computation in case the classical verification part shows that the algorithm yielded a bad output. Unfortunately, such strategy was not yet provided, and is probably much more complicated than the approach we provide here.)

In case # 3 mentioned above the algorithm has more than one correct final outcome and the measurement process directly yields one of the correct solution. In Sect. 3.2 we show how the problem of the multiple search outputs can be resolved. Our solution involves making multiple searches on the same computer and then sorting the results. This way with high probability the computers will have the same sorted list.

If the role of measurements was restricted to only these three cases described above then the three strategies mentioned here, delaying the measurement sorting and randomizing, would solve the measurement problem for ensemble quantum computation. However, there are some "hidden" uses of the measurement process, particularly involving error correction and fault tolerance, during the execution of the algorithm (i.e., during the part that we broadly described as involving only unitary operations) that are a lot harder to address.

The ensemble-measurement problem is much more acute in the case of fault-tolerant computations; Our most important result in this paper is to solve the problem of performing fault-tolerant ensemble quantum computation.

The schemes proposed so far for quantum fault tolerant computation provide an incomplete set of gates, i.e., a set of gates that is not universal for quantum computation. In order to complete the set to a universal set, the schemes use interactions with ancilla qubits, which are then measured—see Shor (1996), Knill et al. (1998) and Preskill (1998). Each such measurement is followed by an application of a unitary operation, U_j, that depends on

the outcome of the measurement (j). (See Sect. 4 for a review.) A direct scheme for removing such measurements (followed by the required unitary operations U_j), and replacing them by controlled operations, $\Lambda(U_j)$, will not in general be realizable. This is because, $\Lambda(U_j)$, might not be realizable by the incomplete set of fault tolerant gates. For example, if one attempts to remove measurements in Shor's scheme for fault tolerant realization of Toffoli gate (Shor 1996), then the corresponding controlled operations would itself require Toffoli gates! Sect. 5 describes how an analysis of error propagation and a careful usage of classical reversible circuits can allow one to delay measurements in a fault tolerant manner, and allow for fault tolerant NMR quantum computing, via this hybrid classical-quantum strategy.

3 Quantum algorithms

Here we study different known quantum algorithms that cannot be implemented directly on ensemble quantum computers and we provide modifications to make them suitable for such computers.

3.1 The factorization algorithm

In the Shor's factorization algorithm the aim is to factor a large number n. To do so, one uses a random number x and tries to find the least positive integer r such that $x^r \equiv 1 (\mathrm{mod}\, n)$. This least r is the *order* of x mod n, and n can be factored with a high probability, once r is known.

Shor's algorithm does not yield r directly (in the quantum process). Instead, another integer c is the actual outcome of the quantum protocol, from which the right r can *sometimes* be obtained by a classical algorithm. Let us call the outcome of the classical algorithm r'; in at least $O(1/\log \log n)$ fraction of the cases, the number r' is the desired r, and whether it is the case or not is checked via a classical algorithm. Let the probability of a correct result (on an individual computer) be p_r. While the order r (for a given x and n) is unique, the result c and the calculated r' are not unique. Having several good outcomes c_i does not cause a problem (as noted by in Gershenfeld and Chuang 1997), since the quantum computer can perform a classical algorithm which calculates r from any of the possible c_i. However, this operation by itself is not sufficient, since many of the computers (probably, most of them) give an outcome r' which is not the correct r. When expectation values are measured for the jth bit, the correct result r_j happens with small probability p_r, and hence it is obscured by the wrong results r'_j.

One potential situation, which could lead to a simple resolution, is if the wrong-r results are well distributed (e.g., totally random); in such a case, on the average these wrong-r results will cancel out (e.g., average to yield zero) and will not obscure the correct result. Let us show that this is not always the case, and that the bad results are not always averaged to zero, and hence the good result sometimes is indeed obscured.

The output c of the quantum process in Shor's algorithm is used to calculate the order r (Shor 1997). For this, the integers d' and r' are found such that

$$\left|\frac{c}{q} - \frac{d'}{r'}\right| \leq \frac{1}{2q}$$

where $n^2 < q \leq 2n^2$, and q is a power of 2. Then the fraction d'/r' is unique. The integer r' is the output of the algorithm as the desired order (which is actually r). To continue, let $\alpha(c)$

be the unique integer such that $-q/2 \le \alpha(c) \le q/2$ and $rc \equiv \alpha(c) (\mathrm{mod}\ q)$. One of the possible situations that leads to incorrect answer is that the output c of the quantum process satisfies the condition

$$\left|\frac{c}{q} - \frac{d}{r}\right| \le \frac{1}{2q}$$

and d and r *are not* relatively prime. Then the answer, instead of r, would be a divisor of r. The probability that such event occurs is (see Shor 1997) approximately $4(r - \phi(r))/(\pi^2 r)$. This probability can be some constant far away from zero. For example, if $r = 2^s 3^t$, then $\phi(r) = r/3$ and the probability the algorithm provides a divisor of r is ≈ 0.135.

Let us now present a modified factorization protocol that bypasses this ensemble-measurement problem. The idea is to replace an additional part of the classical protocol, a part which verifies that r is indeed the order, by a quantum one. Also, a simple (but crucial) modification of the protocol is required. Let the register holding the result (r or r') be called s_1. Let us use an additional register s_2 of the same number ℓ of qubits as s_1. Let the register s_2 be in the state

$$H|0\rangle \otimes H|0\rangle \otimes \cdots \otimes H|0\rangle = \frac{1}{2^{\ell/2}} \sum_{x \in \{0,1\}^\ell} |x\rangle, \tag{1}$$

where $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Now we augment the quantum factorization algorithm with the following procedure. When the original factorization algorithm finishes, test the result in the register s_1 to see whether it gives the correct value of the order r. If the result on the ith computer is indeed the order then nothing is to be done and the outcome r is kept in s_1. Whenever the result is an incorrect value r', swap the contents of the registers s_1 and s_2 so the outcome r' is replaced by the state $H|0\rangle \otimes \cdots \otimes H|0\rangle$ which yields a completely randomized outcome once it is measured. Now, a measurement of the jth bit on s_1 will give the correct result if the string holds the state r or it yields zero (on average) if the string originally contained the wrong result r'.

Although the strength of the good signal may be small, there are enough computers running in parallel to read it since in the worst case, it is only logarithmically small.

3.2 The search algorithm

Certain search operations in a database can be done more efficiently on a quantum computer than on a classical computer (Grover 1996). Here the search means to find some item x in the database such that x satisfies some predefined condition T; i.e., we are looking for the solutions of $T(x) = 1$. The analysis of Boyer et al. (1998) shows that if the size of the database is N and the number of solutions are t, Grover's algorithm, with high probability, can find a solution in time $O(\sqrt{N/t})$. When there is only one solution, this algorithm yields the desired result also on an ensemble computer.

However, when several (say $t \ge 2$) different items satisfy the required condition, the protocol will randomly yield one of them. Therefore, in this case the algorithm is not suitable for ensemble computation. We show here how this algorithm can be modified such that ensemble computation still provides a correct solution with high probability.

We assume t, the number of solutions, is *known* and *constant* (the general case will be studied in the Sect. 3.2). We first consider the case $t = 2$. When processed on an ensemble-measurement computer, only expectation values are obtained, and the two outcomes

partially obscure each other to yield zero (as the average expected value) for jth bit of the answer if the jth bits of the two solutions are different.

To solve this problem we suggest to hold several (say m) computers in one molecule. After each computer in the molecule finishes Grover's algorithm, the procedure is continued by sorting the outputs of different computers in an increasing order. The algorithm then compares the first and the last results, and if they are equal then both are replaced by a randomized data (see Eq. 1) as in the modified Shor's algorithm. Once the first and last computers hold different outcomes, we are promised that the small solution is always the first, and that the large solution is always the last. Thus, we can obtain both solutions via measurements of the expectation values of the qubits.

The probability that the first and the last solutions are the same is $\frac{1}{2^m}$, so the final outcome is obtained with probability exponentially close to one. Even without applying the randomization to the bad outcomes, the expected outcomes are still readable.

When $t > 2$, we apply the same procedure (without randomization to the bad outcomes). We still reorder the solutions so that the minimal solution is in the first position. However, we might obtain different minimal solutions for different molecules. The probability of failing to obtain the global minimum solution in the first position is $(1 - \frac{1}{t})^m$, and as long as it is small (say less than $e^{-\lambda}$, which holds if $m > \lambda t$) the protocol can work properly. Note that this modified algorithm still works in time $O(\sqrt{N/t})$.

Only the smallest and largest solutions can be obtained by the above method. If one needs the other solutions, these can easily be obtained via similar methods, once some solutions are already known.

3.2.1 Search algorithm: the case of unknown number of solutions

Now we consider the most general case. Here we do not assume any condition on t, the number of solutions; it can be known or unknown, large or even zero. Our method is based on a binary search. We also utilize the following fact established in Boyer et al. (1998): Let $\mathcal{B}$ be a database of size M; then the search algorithm, with high probability, starting with the input $\frac{1}{\sqrt{M}} \sum_{x \in \mathcal{B}} |x\rangle$ in time $O(\sqrt{M})$ can determine whether there is any solutions in $\mathcal{B}$ or not.

Without loss of generality, we can assume that the database is represented as the members of the unit cube $V = \{0, 1\}^n$. So $M = 2^n$. For any string $\alpha = (\alpha_1, \ldots, \alpha_k) \in \{0, 1\}^k$, let V_α be the subset of V consisting of all strings $(\alpha_1, \ldots, \alpha_k, x_{k+1}, \ldots, x_n)$; i.e., V_α contains all strings in V that start with α. Thus $|V_\alpha| = 2^{n-k}$. For example, $V_0 = \left\{(0, \alpha_2, \ldots, \alpha_n) : (\alpha_2, \ldots, \alpha_n) \in \{0, 1\}^{n-1}\right\}$.

Our algorithm first checks whether there is a solution or not. If there is no solution then it stops. Otherwise it runs in n stages. The output of the stage j is a database $\mathcal{B}_j$ of size 2^{n-j} which contains a solution. At the end $\mathcal{B}_n = \{\xi\}$, where ξ is a solution. The algorithm starts with the database $\mathcal{B}_0 = V$. It checks whether there is any solution in V_0. If there is a solution then $\mathcal{B}_1 = V_0$, otherwise $\mathcal{B}_1 = V_1$. In a general stage $j + 1$, the input is of the form $\mathcal{B}_j = V_{\alpha_j}$ where $\alpha_j \in \{0, 1\}^j$, and there is a solution in $\mathcal{B}_j$. Then the algorithm checks whether there is a solution in $V_{\alpha_j 0}$, if so then the output of this stage is $\mathcal{B}_{j+1} = V_{\alpha_j 0}$, otherwise the output is $\mathcal{B}_{j+1} = V_{\alpha_j 1}$. This completes the description of our search algorithm. It is easy to check that this algorithm always provides the *first* solution in the lexicographic order. So we have presented a quantum search algorithm that always gives a unique output, no matter how many solutions are there. This is an algorithm which can be implemented on an ensemble–measurement computer. Note that the running time of this algorithm is

$$O\left(\sqrt{2^n}+\sqrt{2^{n-1}}+\cdots+\sqrt{2}\right)=O\left(\sqrt{M}\right).$$

4 Review of fault tolerant quantum computing

The idea of quantum fault tolerant computation (Shor 1996; Aharonov and Ben-Or 1996; Knill et al. 1998; Kitaev 1997; Preskill 1998) can be described briefly as follows. Suppose that we have a (noise-less) quantum circuit C which we want to simulate by a noisy quantum computer. On the noisy quantum computer, instead of circuit C we perform a fault tolerant circuit $\widetilde{C}$. The physical bits $|0\rangle$ and $|1\rangle$ are replaced by logical bits $|0\rangle_L$ and $|1\rangle_L$, where these are some entangled states of a block of physical qubits. While C operates on physical qubits representing the data, in the circuit $\widetilde{C}$ all operations are performed on logical qubits which are error-correction-encoded data, i.e., each data qubit or a set of data qubits is represented as a block of qubits that belongs to some quantum error-correcting code. Then each operation of C performed by a gate g_j is simulated by a procedure (sub-circuit) $\widetilde{g_j}$ in the circuit $\widetilde{C}$ such that in $\widetilde{g_j}$ each computation transforms codewords to codewords. In order to avoid accumulation of errors, after each computation in $\widetilde{g_j}$ a correction procedure is performed to correct any error that is introduced in that computation. Thus, in the fault tolerant circuit $\widetilde{C}$ each computation step is followed by a correction step.

The operations on the encoded qubits introduce a large number of additional gates and qubits, and, unless one is careful, it is possible that more errors are introduced than can be corrected by the code. To avoid any such catastrophic accumulation of errors, it is desirable that the operations in the fault tolerant circuits prevent "spreading of errors" by making sure that each gate error causes at most a single error in each block. It is useful now to review how errors propagate in quantum circuits. For example, consider the CNOT (controlled-not) gate which performs the operation $|a\rangle_c|b\rangle_t \mapsto |a\rangle_c|a\oplus b\rangle_t$ in the computation basis; for the rest of this paper, we shall drop the subscripts c (control) and t (target) and designate the control bit as the one on the left side. Clearly, applying the CNOT operation from one bit to many target bits can propagate one bit error from the control bit to all the target bits. On the other hand, applying CNOT from many control bits to one target bit can propagate one phase error from the target bit to all the control bits. It is easy to observe this "back" propagation of the phase errors: if a phase error happens on the second (target) qubit in the state $(|0\rangle+|1\rangle)\otimes(|0\rangle+|1\rangle)$ and a CNOT is applied after, we will get

$$(|0\rangle+|1\rangle)\otimes(|0\rangle-|1\rangle)\xrightarrow{\text{CNOT}}|0\rangle\otimes(|0\rangle-|1\rangle)+|1\rangle\otimes(|1\rangle-|0\rangle)=(|0\rangle-|1\rangle)\otimes(|0\rangle-|1\rangle)$$

which results in a phase error in the control qubit. Hence, fault tolerant computation requires that this gate be applied only in the case where the control qubit $|a\rangle$ and the target qubit $|b\rangle$ belong to different blocks.

This error-propagation phenomenon is also true for other controlled operations, and this motivated a *sufficient* condition for fault tolerance: only perform bit-wise or transversal[2] operations on qubits within a code. It is, however, *not* a *necessary* condition for fault

[2] By transversal operations, we mean operations that act on at most one qubit in any code block. For instance, a gate applied on the first bit of one codeword and the first bit of a second codeword, and on the second bit of one codeword and the second bit of a second codeword, and so fourth.

tolerance, and careful constructions may allow one to apply control gates from many control bits onto one target bit, without destroying the fault tolerant computation.

Therefore, to achieve a quantum fault tolerant computation, it is enough to show that a *universal* set of quantum gates can be constructed with only bit-wise and transversal operations on a quantum code. Quantum fault tolerant schemes usually (see, e.g., Shor 1996 and Preskill 1998) depend on measurements to ensure that the set of the operations permissible on encoded data is actually a universal set. Recall that we cannot depend on measurements in ensemble computers, but must still create a universal set to achieve fault tolerance. Some of the gates in the universal set do not require measurements, e.g., the operations H, $\sigma_z^{1/2}$, and CNOT. [For a class of codes called CSS codes (Shor 1996), H, σ_z, and CNOT can simply be achieved by performing the same gate bit-wise on the individual qubits (e.g., H is achieved on code words via applying H on individual qubits), while the bit-wise $\sigma_z^{1/2}$ yields a $\sigma_z^{-1/2}$ logical gate, hence requires an additional step of bit-wise σ_z, to yield the desired logical gate.] In previous works (Shor 1996; Boykin et al. 1999, 2000) at least one gate in the universal set requires measurements. That's bad for ensemble computers. We now present tools that will allow us to create a measurement-free universal quantum fault tolerant set of gates.

5 Measurement-free quantum fault tolerant gates

There is always a simple scheme that potentially allows one to postpone measurements of ancilla qubits in quantum computation. Unfortunately, the simple scheme never works in the case of generating universal fault tolerant gates. In fault tolerant computation and error recovery, often a measurement is followed by an operation U_j[3] *if* the outcome of the measurement is 1. As explained in Sect. 2, the scheme for delaying the measurement can be successfully implemented only if the measurement followed by a U_j operation can be replaced by controlled-U_j (denoted $\Lambda(U_j)$) in the set of *available* measurement-free operations; i.e., control operations which can be implemented on encoded data fault tolerantly and directly without using any measurements. However, in the schemes proposed so far, the required control operations $\Lambda(U_j)$ are not implementable in a direct fault tolerant manner. For instance, in Shor's fault tolerant set of gates (Shor 1996), a measurement is required for the preparation of a Toffoli gate, but a Toffoli gate is required if we want to delay that measurement. This is because the measurement is followed by a CNOT operation, and hence can only be replaced by a controlled-CNOT, which is a Toffoli gate. This seems like a catch-22 situation![4]

The solution comes from the vital observation that some operations need protection only from the bit errors, and do not need to use full quantum codes: by replacing the quantum ancilla (in a logical basis $|0\rangle_L$ and $|1\rangle_L$) by a "classical ancilla" in a "classical" basis $|\mathbf{0}\rangle = |0\rangle \cdots 0$ and $|\mathbf{1}\rangle = |1\rangle \cdots 1$, we can use the classical ancilla to perform $\Lambda(U_j)$ in a fault tolerant manner. This can be done in the two cases where the Toffoli gate is required for Shor's fault tolerant set of gates, and the $\Lambda(\sigma_z^{1/2})$ gate required for the basis of Boykin

[3] U_j can be performed fault tolerantly using the given, non-universal, set of operations.

[4] Similarly, in the fault tolerant universal set of gates suggested in Boykin et al. (1999), the generation of the $\sigma_z^{1/4}$ gate without measurements leads to a catch-22 problem; a $\sigma_z^{1/2}$ gate (which follows the measurement) needs to be replaced by a $\Lambda(\sigma_z^{1/2})$ gate, which is not available as long as the $\sigma_z^{1/4}$ gate is not available.

et al. (1999). One can interpret the classical basis as the classical repetition code. We call the ancilla in these states "classical" since we are not concerned with phase errors on these bits. A classical error-correction code can correct bit errors in the classical ancilla. Despite the fact that phase errors are not corrected in the classical ancilla, we found that the use of such a classical ancilla is still good enough for our purpose.

5.1 Replacing measurements of encoded ancilla qubits

In the following, we shall replace the measurement of the quantum ancilla followed by the operation U acting on the quantum data, by a sequence of operations: we copy the two basis states of a quantum ancilla into a classical ancilla, we perform classical error correction on the classical ancilla, and we use the classical ancilla as a control bit for performing the operation $\Lambda(U_j)$ with the quantum data as the target bit.

The measurement of the quantum ancilla in the original protocol can be done as follows (Preskill 1998): measure each of the physical qubits, and perform a classical error correction on the outcome of this measurement to determine the state of the ancilla. For example, if the 7-bit CSS code (Shor 1996) is used to encode data, then a measurement will yield a (possibly corrupted) codeword of a classical 7-bit Hamming code. After classical error correction, if the parity of the codeword is even, then the ancilla has collapsed to the state $|0\rangle_L$; otherwise, it has collapsed to the state $|1\rangle_L$. Classical error correction is enough to protect the output bit b, because phase errors before a measurement will not change the outcome probabilities.

In Fig. 1, we represent a circuit that computes operation $\mathcal{N}_1$ for the seven-bit CSS code, where $\mathcal{N}_1$ stands for the operator of Eq. 2 with only one bit of the classical ancilla. The ancilla bits labeled syndrome are used to prevent the spread of one bit error from the quantum ancilla into the classical bit. These bits are exactly the parity check of the syndrome of the 7-bit Hamming code. Only two errors (in any of the inputs, the gates or the time steps) shall yield an error in the classical bit.

The circuit $\mathcal{N}_1$ flips the bit b if the quantum ancilla (acting here as a control bit) is $|1\rangle_L$, and does nothing otherwise. This circuit operates properly as long as there is up to one bit error in the quantum data (there can actually be an unlimited number of phase errors). Note that phase errors in the lower part will spread to the quantum ancilla. This is of no consequence, however, since the quantum ancilla never interacts with the quantum data in later stages. Bit errors in the quantum ancilla are important, since the process is repeated n

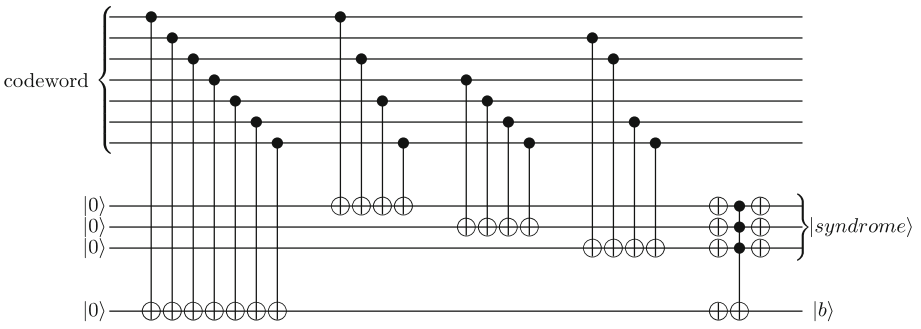

Fig. 1 The operation $\mathcal{N}_1$. Note that the circuit shows the generation of only one classical target bit $|b\rangle$; the operations on the last bit have to be repeated to generate multiple target bits

times; hence, bit errors, created in the quantum ancilla at initial stage of $\mathcal{N}_1$, will spread errors into the next bits of the classical ancilla. Fortunately, bit errors are not transmitted from the classical to quantum section, and the quantum ancilla cannot be disturbed by a bit error in bits of the classical ancilla or the syndrome ancilla.

As a step toward removing the measurement from the original protocol, we propose a new gate that copies an encoded quantum ancilla word onto a classical ancilla:

$$\mathcal{N}: \begin{cases} |0\rangle_L \otimes |\mathbf{0}\rangle & \longrightarrow \quad |0\rangle_L \otimes |\mathbf{0}\rangle, \\ |0\rangle_L \otimes |\mathbf{1}\rangle & \longrightarrow \quad |0\rangle_L \otimes |\mathbf{1}\rangle, \\ |1\rangle_L \otimes |\mathbf{0}\rangle & \longrightarrow \quad |1\rangle_L \otimes |\mathbf{1}\rangle, \\ |1\rangle_L \otimes |\mathbf{1}\rangle & \longrightarrow \quad |1\rangle_L \otimes |\mathbf{0}\rangle. \end{cases} \tag{2}$$

Let $\mathcal{N}$ be a unitary operation that implements the above transformation. In the next section we show that the complete $\mathcal{N}$ operation can be done fault tolerantly. In Sects. 5.4 and 5.5, the operation $\mathcal{N}$ will enable us to construct gates for universal fault tolerant computation, without measurement.

5.2 The operation $\mathcal{N}$: quantum-to-classical controlled-NOT

The $\mathcal{N}$ operation "copies" the encoded quantum bit onto the encoded classical ancilla. The repetition code can only correct bit errors in the classical ancilla, but one must be sure that the classical ancilla can still be used to perform $\Lambda(U_j)$ without putting the quantum data in jeopardy. Perhaps counter-intuitively, this is not a problem, since phase errors are transmitted from *target* bit to *control* bit, hence cannot be transmitted from the classical ancilla (control) to the quantum data (target). This leads to the most interesting aspect of our scheme: the data in the classical repetition code, or any classical function of this data, *can* act as control bits in a bit-wise controlled-U operation onto quantum data.

In the complete $\mathcal{N}$ circuit, the $\mathcal{N}_1$ computation on the bottom four bits is repeated at most n times, where n is the number of qubits in a codeword. At each repetition stage, the syndrome bits are discarded, and another bit b_i is created ($i \in \{1, \ldots, n\}$). In principle, the syndrome bits could be ignored, reset, or measured. These bits will not affect the operation beyond their use as a form of error detection in the codeword. The bits b_i are then corrected (to yield the classical 0 or 1) using a majority vote.

In order to reduce the number of operations (and hence improve the fault tolerant threshold), we only need to use a repetition code that will successfully recover from k' errors. Once this number k' is equal to, or greater than, the number of errors, k, that the *quantum* code can correct for, we may stop. For a probability p of an error (per gate, per input bit, and per delay line), the resulting error rate of this circuit is $O(p^2)$, as required for fault tolerant computation. The threshold can easily be calculated by counting the potential places for two errors, and the threshold can be much improved by enhancing the parallelism, and by repeating $\mathcal{N}_1$ only $2k + 1$ times (e.g., with the 7-bit quantum code, that is $n = 7$, which corrects $k = 1$ error, it is enough to repeat the circuit 3 times, correct the outcome using a majority vote, and then copy the result into seven bits).

Later, in Sects. 5.4 and 5.5, we show cases where, indeed, the operations between the classical ancilla and the quantum data can be performed bit-wise, while the same operations cannot be performed bit-wise between quantum ancilla and the quantum data (as the naive solution of delaying measurements would have suggested).

Note that the quantum data may add phase errors to the repetition code, but that is of no concern to us, since the classical repetition code also loses phase coherence in the

measured case. If there are t bit errors in the repetition code, it will result in t errors in the quantum data. Fortunately, bit errors *are* corrected in the repetition code. Hence, *the operation $\mathcal{N}$ enables one to create universal bases without measurement.*

5.3 Creating the special states required for fault tolerant universal computation, without using a measurement

In Sects. 5.4 and 5.5 we describe how to construct gates to produce a universal set without measurement. In both of those sections, we will make use of "special states" which enable the construction of the gate. In this section we describe a general method to produce these special states under general circumstances. Once presented, the descriptions in Sects. 5.4 and 5.5 become much simpler.

Assume that a quantum code of length n is used for encoding data. Suppose that $U \in U(2^l)$ [for our purpose it is enough to consider up to three qubits ($l = 3$) operations], and $\widetilde{U} = U^{\otimes n}$ is the unitary operation on the codewords obtained by applying U bit-wise. Suppose that $\widetilde{U}$ has eigenvectors $|\phi_0\rangle$ and $|\phi_1\rangle$ such that

$$\widetilde{U}|\phi_0\rangle = |\phi_0\rangle \qquad \text{and} \qquad \widetilde{U}|\phi_1\rangle = -|\phi_1\rangle.$$

Then the quantum circuit in Fig. 2 outputs the eigenvector $|\phi_0\rangle$ if the input state is $\alpha|\phi_0\rangle + \beta|\phi_0\rangle$ for any α, β. In Fig. 2, $\widetilde{U}_{\text{flip}}$ is a unitary operation that maps $|\phi_0\rangle$ on $|\phi_1\rangle$ and vice versa. The operations $\Lambda(\widetilde{U})$ (i.e., the controlled-$\widetilde{U}$), and H are applied bit-wise.

This scheme is practical if it is possible to prepare a state $\alpha|\phi_0\rangle + \beta|\phi_1\rangle$, where the values of α and β do not matter. In this circuit the first line is a single parity bit, and each of the second and third inputs are blocks of n qubits, containing the cat-states lines and the special state lines, respectively. The third gate, the CNOT gate which we call here P, is a parity gate which calculates the parity of the cat-state lines and puts the result in the parity bit. This is done by a sequence of CNOTs from each control bit onto one target bit. The figure only demonstrates the creation of one parity bit $|\phi_0\rangle$ in an unprotected manner as far as a bit error in the parity bit is concerned. The real circuit is a bit different: The operations $\Lambda(\widetilde{U}), H$ and P, are repeated n times, each time with fresh cat-states and a fresh parity bit (but on the same special state's lines). Then a majority vote is calculated on the parity bits, in order to reduce the probability that an error in a cat state or in the parity bit will ruin the result. Then the n parity results are corrected, so that the probability of two errors becomes low [that is, of order $O(p^2)$]. Finally, the parity result is used to control $\widetilde{U}_{\text{flip}}$ in a bit-wise manner, so that the special state is created via a fault tolerant operation.

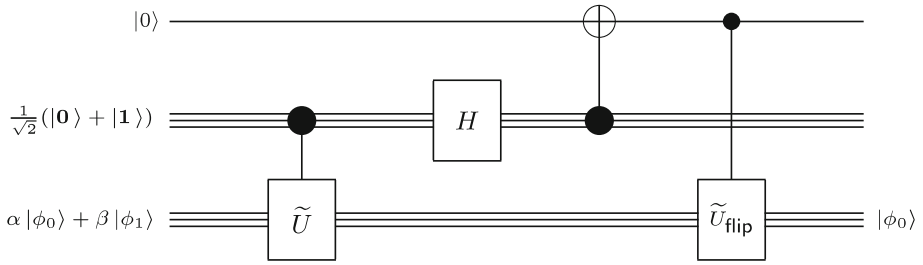

Fig. 2 Preparing an eigenvector

5.4 Fault tolerant $\sigma_z^{1/4}$ without measurement

We show here a modified version of the original method for implementing $\sigma_z^{1/4}$ on codewords (Boykin et al. 1999) which does not use measurements. Using the method described in Sect. 5.3, we need to prepare the following state

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle_L + e^{\frac{i\pi}{4}}|1\rangle_L\Big).$$

This state can be prepared with a circuit of form given in Fig. 2. For this purpose, let $\widetilde{U} = e^{\frac{i\pi}{4}}\sigma_x\sigma_z{\sigma_z}^{1/2}$ and $|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle_L - e^{\frac{i\pi}{4}}|\psi_1\rangle_L)$. Then $\widetilde{U}|\psi_0\rangle = |\psi_0\rangle$, $\widetilde{U}|\psi_1\rangle = -|\psi_1\rangle$, and $U_{\mathsf{flip}} = \sigma_z$. Finally, see that $|\overline{0}\rangle = \frac{1}{\sqrt{2}}(|\psi_0\rangle + |\psi_1\rangle)$. Note, as required in Sect. 5.3, both $\widetilde{U}$ and U_{flip} are in the directly fault tolerant set. Hence we have all the requirements of the previous section, and thus we may use that method to create $|\psi_0\rangle$.

Now we are ready to describe the fault tolerant $\sigma_z^{1/4}$ without measurement. The circuit in Fig. 3 shows the fault tolerant implementation of $\sigma_z^{1/4}$ on a codeword $|x\rangle_L$. In this circuit, $\mathcal{N}$ is the unitary operation defined in (2). Apart from replacing the standard measurements by the $\mathcal{N}$ circuit, this figure is exactly the same as the one drawn in Boykin et al. (1999) to implement the $\sigma_z^{1/4}$ gate. In this figure each input in fact denotes a block of qubits, and operations are bit-wise.

5.5 Fault tolerant Toffoli without measurement

The more conventional (and more complicated) set of universal fault tolerant gates contain the Toffoli instead of the $\sigma_z^{1/4}$. We show explicitly how to implement Toffoli on encoded data without using any measurement. This scheme is a modified version of Shor's original method for implementing Toffoli on codewords (Shor 1996), and is similar to the one applied to $\sigma_z^{1/4}$.

In Shor's method (as in the other basis we have shown) a preparation of a special state is required, hence we first prepare the state

$$|\mathsf{AND}\rangle = \frac{1}{2}(|000\rangle_L + |010\rangle_L + |100\rangle_L + |111\rangle_L), \tag{3}$$

without using measurement, based on our scheme presented in Sect. 5.3.

To get $|\mathsf{AND}\rangle$ we let $U = \Lambda(\sigma_z) \otimes \sigma_z$, and we chose

$$|\overline{\mathsf{AND}}\rangle = \frac{1}{2}(|001\rangle_L + |011\rangle_L + |101\rangle_L + |110\rangle_L).$$

Then $\widetilde{U}|\mathsf{AND}\rangle = |\mathsf{AND}\rangle$, $\widetilde{U}|\overline{\mathsf{AND}}\rangle = -|\overline{\mathsf{AND}}\rangle$, $U_{\mathsf{flip}} = I \otimes I \otimes \sigma_x$, and

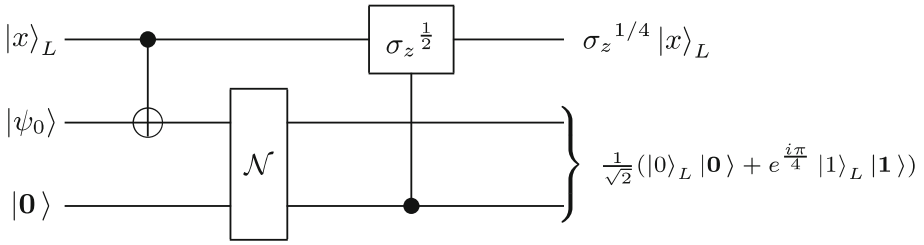

Fig. 3 Fault tolerant $\sigma_z^{1/4}$ without measurement

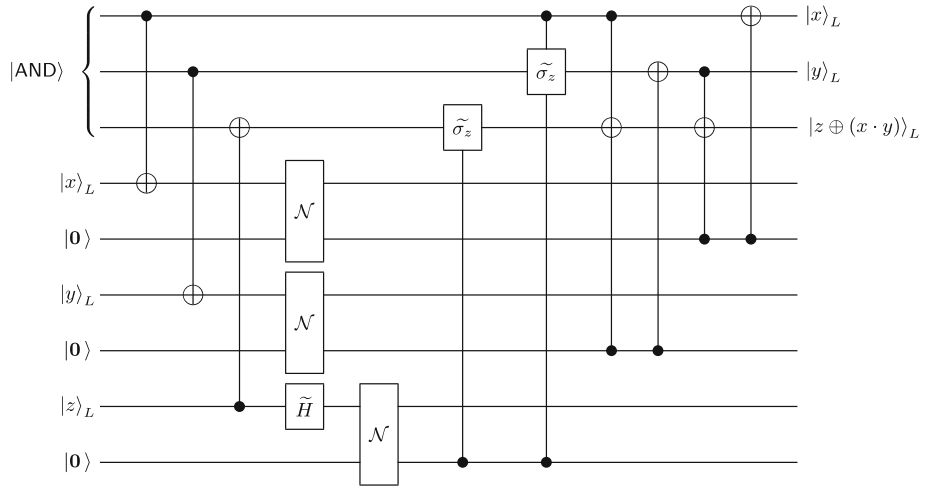

Fig. 4 Fault tolerant Toffoli without measurement

$$\frac{1}{\sqrt{2}}(|\mathsf{AND}\rangle + |\overline{\mathsf{AND}}\rangle) = (\widetilde{H} \otimes \widetilde{H} \otimes \widetilde{H})|000\rangle_L.$$

Note, as required in Sect. 5.3, both $\widetilde{U}$ and U_{flip} are in the directly fault tolerant set. Hence we have all the requirements of the previous section, and thus we may use that method to create $|\mathsf{AND}\rangle$.

A different solution to this step was given (independently) by Aharonov and Ben-Or (1997) (see, especially, the Quant-ph extended version). Our procedure for constructing the fault tolerant Toffoli gate is presented in Fig. 4. In this circuit $\mathcal{N}$ is the unitary operation defined in (2); apart from replacing the standard measurements by our $\mathcal{N}$ circuit, this figure is exactly the same as the one drawn by Preskill (1998) to describe Shor's way of obtaining the Toffoli gate. Note that in this figure each input represents a block of qubits and operations on these blocks are defined in the natural way. Also note that the first three top outputs of this circuit are in a tensor product with the rest of the outputs.

6 Error recovery in the error correction process

Standard error correction can be viewed as a computation with more than one good answer, and thus belongs to case (1) discussed in Sect. 2. On different computers in the ensemble the syndrome of the error will be different, and thus is not unique. In the standard error correction prescription, measurement is used to collapse the ancilla qubits containing the information as to which error occurred (the syndrome). Then these syndrome bits are processed by a classical reversible algorithm to determine the errors, and a unitary operation to correct the error is applied to the data qubits by the output bits of the classical algorithm. In the measurement-free case, the ancilla qubits need not be measured, and the classical subroutine (following the measurement) could be incorporated into the original quantum algorithm.

The standard error correction operations require the use of a universal set of classical gates (e.g. NOT, CNOT, Toffoli). As in Sect. 5, for the classical part of the computation we do not care about phase errors, and as such we do not need the full power of quantum fault tolerance in this part of the computation. Hence, the techniques of Sect. 5 can be applied so that the classical subroutine is carried out on a classical code. The state of the ancilla qubits can be first copied onto a classical repetition code using the $\mathcal{N}$ gate. Now classical reversible computation can be performed on the repetition code and then a control operation can be performed on the quantum data to correct the errors.

Since phase errors from the classical sub-circuit will not propagate to the quantum data, using repetition codes to correct any bit errors in the sub-circuit is sufficient. The observation that phase errors cannot propagate from the "classical" part of the computation allows one to fault tolerantly replace quantum Toffoli gates by classical ones in the error recovery process.

7 Concluding remarks

To summarize, various conventional algorithms cannot run on ensemble (bulk) computers, such as NMR quantum computers, since individual qubit measurement is not available. We explained in details why running various algorithms on ensemble computers is not always straightforward. We modified Shor's algorithm, Grover's algorithm (in the case of more than one solution), and fault tolerance protocols so that they can run on ensemble computers.

A very preliminary version of this work appears in the Los-Alamos archive, quant-ph/9907067. A partial version describing the algorithm for fault-tolerant quantum computing appears in the DSN'04 conference proceedings (Boykin et al. 2004).

In a prior work, addressing fault tolerant computation, Aharonov and Ben–Or (1996) have observed that the measurements required for fault tolerant computation can be substituted by reversible classical circuits performing controlled operations. D. Aharonov also sent us a manuscript (Aharonov D, private communication) with results regarding Toffoli gate which are very similar to those obtained here. Knill et al. (1996) followed a different approach that potentially does not require measurements. However, to the best of our knowledge, a proof of universal fault tolerant computation via their approach is not available. In particular, a measurement-free implementation of the Hadamard gate using that approach has not been demonstrated. Finally, Peres (1998) also discusses the possibility of measurement-free encoding and decoding procedures in quantum error-correction. However, in his scheme the quantum information is transformed to a single qubit, and his method is not suitable for fault tolerant computation.

Acknowledgements We are thankful to Dorit Aharonov for many helpful remarks. This work was supported in part by grants from the Revolutionary Computing group at JPL (contract #961360), and from the DARPA Ultra program (subcontract from Purdue University #530-1415-01). The work of T.M. was supported in part by the Israeli Ministry of Defense, by the Institute for Future Defense Research, and by the Israel Science Foundation—FIRST (grant #4088103).

References

Aharonov D, Ben-Or M (1996) Polynomial simulations of decohered quantum computers. In: Proceedings of the 37th annual symposium on foundations of computer science, FOCS'96. IEEE Computer Society Press, Los Alamitos, pp 46–55

Aharonov D, Ben-Or M (1997) Fault tolerant quantum computation with constant error. In: Proceedings 29th annual ACM symposium on theory of computing, ACM, New York, pp 176–188. Extended version: Quant-ph/9906129

Boyer B, Brassard G, Hoyer P, Tapp A (1998) Tight bounds on quantum searching. Fortschr Phys 46 (4–5):493–505

Boykin PO, Mor T, Pulver M, Roychowdhury VP, Vatan F (1999) On universal and fault–tolerant quantum computation. In: Proceedings of the 40th annual symposium on foundations of computer science, pp 486–494. Quant-ph/9906054

Boykin PO, Mor T, Pulver M, Roychowdhury VP, Vatan F (2000) A new universal and fault-tolerant quantum basis. Inf Process Lett 75:101–107

Boykin PO, Mor T, Roychowdhury VP, Vatan F, Vrijen R (2002) Algorithmic cooling and scalable NMR quantum computers. Proc Natl Acad Sci USA 99(6):3388–3393

Boykin PO, Mor T, Roychowdhury VP, Vatan F (2004) Fault tolerant computation on ensemble quantum computers. In: Proceedings of the 2004 international conference on dependable systems and networks (DSN'04), IEEE Computer Society Press, Los Alamitos, pp 157–166

Brassard G, Braunstein SL, Cleve R (1998) Teleportation as a quantum computation. Phys D 120(1–2): 43–47

Cory DG, Fahmy AF, Havel TF (1997) Ensemble quantum computing by nuclear magnetic resonance spectroscopy. Proc Natl Acad Sci USA 94:1634–1639

Cory D, Price M, Mass W, Knill E, Laflamme R, Zurek W, Havel T, Somaroo S (1998) Experimental quantum error correction. Phys Rev Lett 81:2152–2155

Elias Y, Fernandez JM, Mor T, Weinstein Y (2007)Optimal algorithmic cooling of spins. In: Proceedings of the 6th international conference on Unconventional Computation (UC'2007), Kingston, Canada. Lecture Notes in Computer Science (LNCS), vol 4618, Springer, Berlin, pp 2–26

Fernandez JM, Lloyd S, Mor T, Roychowdhury VP (2004) Algorithmic cooling of spins: a practicable method for increasing polarization. Int J Quant Inf 2(4):461–477

Gershenfeld N, Chuang IL (1997) Bulk spin-resonance quantum computation. Science 275(5298):350–356

Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the 28th annual ACM symposium on theory of computing, STOC'96. ACM Press, New York, pp 212–219

Jones JA, Mosca M, Hansen RH (1998) Implementation of a quantum search algorithm on a quantum computer. Nature 393:344–346

Kitaev AY (1997) Quantum computation: algorithms and error correction. Russ Math Surv 52(6):1191–1249

Knill E, Laflamme R, Zurek WH (1996) Accuracy threshold for quantum computation. Quant-ph/9610011

Knill E, Laflamme R, Zurek HW (1998) Resilient quantum computation: error models and threshold. Proc R Soc Lond A 454(1969):365–384

Nielsen MA, Knill E, Laflamme R (1998) Complete quantum teleportation using nuclear magnetic resonance. Nature 396:52–55

Peres A (1998) Quantum disentanglement and computation. Superlattices Microstruct 23(3–4):373–379

Preskill J (1998) Reliable quantum computers. Proc R Soc Lond A 454(1969):385–410

Schulman LJ, Vazirani U (1999) Molecular scale heat engines and scalable quantum computation. In: Proceedings of the 31st annual ACM symposium on theory of computing, STOC'99. IEEE Computer Society Press, Los Alamitos, pp 322–329. Quant-ph/9804060, 1998: Scalable NMR quantum computation

Schulman LJ, Mor T, Weinstein Y (2007) Physical limits of heat-bath algorithmic cooling. SIAM J Comput 36(6):1729–1747

Shor PW (1996) Fault-tolerant quantum computation. In: Proceedings of the 37th annual symposium on foundations of computer science, FOCS'96. IEEE Computer Society Press, Los Alamitos, pp 56–65

Shor PW (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J Comput 26(5):1484–1509

Vandersypen LMK, Steffen M, Breyta G, Yannoni CS, Sherwood MH, Chuang IL (2001) Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. Nature 414:883–887

Vrijen R, Yablonovitch E, Wang K, Jiang HW, Balandin A, Roychowdhury V, Mor T, DiVincenzo D (2000) Electron-spin-resonance transistors for quantum computing in silicon-germanium heterostructures. Phys Rev A 62(1):012306-1–012306-10