Investigating the practical implementation of Shor's algorithm

Article *in* Proceedings of SPIE - The International Society for Optical Engineering · February 2005

DOI: 10.1117/12.583191

3 authors, including:

Austin G. Fowler
Google Inc.
164 PUBLICATIONS **11,495** CITATIONS

SEE PROFILE

Investigating the practical implementation of Shor's algorithm.

Simon J. Devitt, Austin G. Fowler and Lloyd C.L. Hollenberg

Centre For Quantum Computer Technology, School of Physics University of Melbourne, Victoria 3010, Australia

ABSTRACT

The publication in 1994 of Shor's algorithm, which allows factorisation of composite number N in a time polynomial in its binary length L has been the primary catalyst for the race to construct a functional quantum computer. However, it seems clear that any practical system that may be developed will not be able to perform completely error free quantum gate operations or shield even idle qubits from inevitable error effects. Hence, the practicality of quantum algorithms needs to be investigated to estimate what demands must be made of quantum error correction (QEC). Several different quantum circuits implementing the quantum period finding (QPF) subroutine, which lies at the heart of Shor's algorithm have been designed, but each tacitly assumes that arbitrary pairs of qubits can be interacted. While some architectures posses this property, many promising proposals are best suited to realising a single line of qubits with nearest neighbour interactions only. This paper will present a circuit suitable for implementing the QPF subroutine for such linear nearest neighbour (LNN) designs. We will then present direct simulation results showing for both the LNN circuit and for a circuit utilising arbitrary interactions, that the QPF subroutine is very sensitive to a small number of errors in the entire circuit.

Keywords: Quantum computing, Shor's algorithm, Error stability, Circuit design

1. INTRODUCTION

Since Shor's discovery of a factoring algorithm, which on a quantum computer can factor large numbers in polynomial time,[1] a large international effort is attempting to construct a practical quantum computer. Currently there are many different proposals for constructing such a device.[2,3] Many of these proposals allow arbitrary pairs of qubits to be interacted (non-LNN), however some proposals utilise a single line of qubits that are restricted to nearest neighbour interactions only. The design of quantum algorithms often assume the former in regards to circuit design. Determining whether these linear nearest neighbour (LNN) architectures can implement quantum algorithms in a practical way is a nontrivial and important question.

Not only is the efficient design of appropriate quantum circuits an important problem but the practical implementation of large scale algorithms and their stability when exposed to errors is necessary. Quantum error correction (QEC) and fault tolerant quantum computation (FTQC)[2] offer a framework to stabilise quantum circuits against errors. However, the complexity of FTQC requires a detailed analysis of large scale algorithms in order to accurately estimate the demands on such procedures. Implementing Shor's factoring algorithm is arguably the ultimate goal of much experimental work and is currently the best example of a large scale quantum circuit. This paper will detail some of the physical implications of implimenting Shor's algorithm on a quantum computer. We will detail an appropriate circuit needed for a LNN array of qubits and provide simulation results showing how the LNN and non-LNN circuits behave when exposed to a discrete error model. Using these results we will examine some of the many issues relating to the practical construction of an appropriate circuit required to factorise a large number. A more detailed discussion of the appropriate LNN circuit can be found in Ref.[4] while a detailed treatment of the error stability of Shor's algorithm can be found in Ref..[5]

Many circuits have been proposed in order to implement Shor's algorithm on a physical quantum computer, as summarised in table 1. Some circuits are optimised for conceptual simplicity, some for speed and some for utilising a minimum number of qubits. This investigation will focus on circuits that require a minimal number of qubits for two reasons. Firstly, scalability of quantum computer architectures implies that circuits requiring as few qubits as possible are desirable. Secondly,

Further author information: (Send correspondence to S.J.Devitt)
S.J.Devitt: E-mail: devitt@physics.unimelb.edu.au, Telephone: +64 3 8344 7129

Circuit	Qubits	Depth
Simplicity[6]	$\sim 5L$	$O(L^3)$
Speed[7]	$O(L^2)$	$O(L \log L)$
Qubits[8]	$\sim 2L$	$\sim 32L^3$
Tradeoff I[9]	$\sim 50L$	$\sim 2^{19}L^{1.2}$
Tradeoff II[9]	$\sim 5L$	$\sim 3000L^2$

Table 1. Number of qubits required and circuit depth of different implementations of Shor's algorithm. Where possible, figures are accurate to leading order in L.

classical simulations of quantum circuits become difficult when manipulating large numbers of qubits. Beauregard[8] details an implementation for a quantum computer that requires $2L + 3$ qubits to factorise a composite number N of bit-length $L = \log_2(N)$. The circuit presented in this paper is based on this circuit. The LNN circuit shown requires $2L + 4$ qubits and $8L^4 + 40L^3 + 116\frac{1}{2}L^2 + 4\frac{1}{2}L - 2$ gates arranged in a circuit depth of $32L^3 + 80L^2 - 4L - 2$, where the depth is the number of layers of gates within the circuit. To leading order in L, the LNN circuit has identical gate counts and depth to the original Beauregard circuit provided one extra qubit is added to avoid doubly controlled gates. Several authors have previously examined the effect of errors on Shor's algorithm.[10–12] However, these simulations often limit the investigation to specific sections of the entire circuit, or to other sources of error such as phase drifts on idle qubits, imperfect gate operations or aspects relating to quantum chaos. This paper will examine the error stability for both LNN and non-LNN circuits when considering the required circuit in full.

This paper is structured as follows. In section 2 Shor's algorithm is briefly reviewed, the algorithm is broken down into a series of classical operations and a quantum period finding (QPF) subroutine. This QPF subroutine is the focus of all the circuits mentioned earlier, including the circuit presented in this paper. In section 3 the QPF subroutine is broken down into a series of simple steps. Sections 4 to 8 present, in order of increasing complexity, the LNN circuit required to implement the QPF subroutine. Section 9 then details the error model used and issues relating to simulations. Section 10 present simulation results looking at both the LNN circuit and original Beauregard circuit when exposed to errors. Sections 11 and 12 will relate these simulation results back to the demands on QEC concatenated codes and the practical implementation of Fault-Tolerant computation for Shor's algorithm.

2. SHOR'S ALGORITHM

As several papers detail the major steps of Shor's algorithm,[1, 13, 14] we will only give a brief overview for the sake of completeness. We first consider a given composite number $N = N_1 N_2$ which has a binary length $L = \log_2(N)$. In order to factorise this number we consider the function $f(k) = x^k \text{mod} N$, where $k = 1, 2, 3, ...$ and x is a randomly chosen integer such that $1 < x < N$ and $\text{gcd}(N, x) = 1$ (gcd $\equiv$ greatest common divisor). The QPF subroutine of Shor's algorithm is designed to determine the period of $f(k)$. i.e, to find the integer $r > 0$ such that $f(r) = 1$. This QPF subroutine is the quantum component of Shor's algorithm. The complete algorithm is composed of both the QPF subroutine and several pre and post processing operations that can be performed in polynomial time using classical techniques. These classical steps are detailed by several authors,[1, 2, 14] however they are not important for this investigation and hence will not be discussed. Once the QPF subroutine returns a value for the period of $f(k)$ the factors of N can be calculated as $N_1 = \text{gcd}(f(r/2) - 1, N)$ and $N_2 = \text{gcd}(f(r/2) + 1, N)$. This is conditional on r being even and $f(r/2) \neq N - 1$. It can be shown[2] that a randomly chosen x will result in the QPF algorithm returning non-trivial factors of N, with probability $1 - 1/2^t$. Where t is the number of distinct prime factors of N. Given N is a product of two primes, we find that in the absence of errors the QPF subroutine still determines a useful value of r, for a random choice of x, with probability $= 0.75$.

Next, we detail how we define success of the QPF subroutine. The underlying steps to factorise a number of binary length L first requires the initialisation of $3L$ qubits to the state $|0\rangle_{2L}|0\rangle_L$. The actual number of qubits used in our circuit is less than this since we replace the $2L$ qubit register with a single master control qubit that is sequentially measured $2L$ times. This replacement does not effect the following analysis which is performed on the full $3L$ qubit computer. For clarity we have broken these $3L$ qubits into a $2L$ qubit k register and a L qubit f register. The next step requires a

Hadamard transform to be performed on each qubit in the k register

$$|0\rangle_{2L}|0\rangle_L \longrightarrow \frac{1}{2^L}\sum_{k=0}^{2^{2L}-1}|k\rangle_{2L}|0\rangle_L. \tag{1}$$

Step three is to apply the function $f(k)$ on the f register, conditional on the values of the k register. The resultant state of the computer is

$$\frac{1}{2^L}\sum_{k=0}^{2^{2L}-1}|k\rangle_{2L}|0\rangle_L \longrightarrow \frac{1}{2^L}\sum_{k=0}^{2^{2L}-1}|k\rangle_{2L}|x^k \mathrm{mod} N\rangle_L. \tag{2}$$

Next we measure the f register. This step can actually be omitted. We present it here to show explicitly how the period r appears within this procedure. After measurement we obtain

$$\frac{1}{2^L}\sum_{k=0}^{2^{2L}-1}|k\rangle_{2L}|f(k)\rangle_L \longrightarrow \frac{\sqrt{r}}{2^L}\sum_{n=0}^{2^{2L}/r-1}|k_0+nr\rangle_{2L}|f_0\rangle_L, \tag{3}$$

where r is the period of f, f_0 is the measured value and k_0 is the smallest value of k such that $f_0 = f(k_0)$. We now apply a quantum Fourier transform (QFT) to the k register. The state of the computer after the application of the QFT becomes

$$\frac{\sqrt{r}}{2^{2L}}\sum_{j=0}^{2^{2L}-1}\sum_{n=0}^{2^{2L}/r-1}\exp(\frac{2i\pi}{2^{2L}}j(k_0+nr))|j\rangle_{2L}|f_0\rangle_L. \tag{4}$$

If we now measure the k register, we will return a value of j with probability given by

$$Pr(j,r,L) = \left|\frac{\sqrt{r}}{2^{2L}}\sum_{n=0}^{2^{2L}/r-1}\exp\left(\frac{2i\pi}{2^{2L}}jnr\right)\right|^2. \tag{5}$$

Equation (5) is strongly peaked at certain values of j. If the period r perfectly divides 2^{2L} then (5) can be evaluated exactly, with the probability of observing $j = c2^{2L}/r$ for $0 \leq c < r$ being $1/r$, and 0 if $j \neq c2^{2L}/r$ [figure (1a)]. If r is not a perfect divisor of 2^{2L}, then the peaks of equation (5) become slightly broader [figure (1b)]. In this case, classical methods can be utilised in order to determine r from the values measured. Various authors[2,14] show how a continued fraction method can be used to determine r, given several measured integer values around these non-integer peaks. Hence, we define the probability of success s for Shor's algorithm as

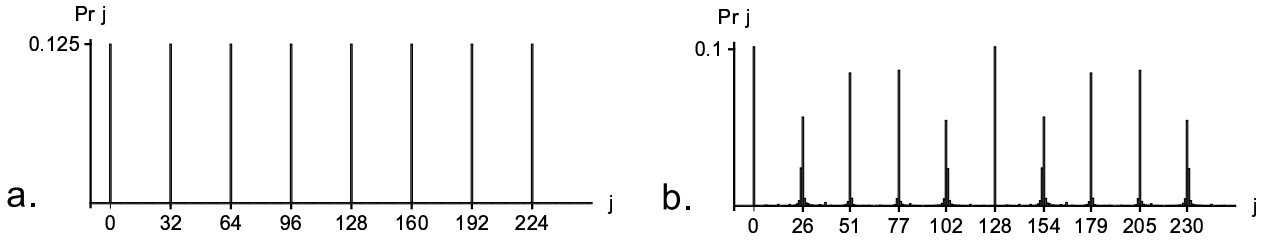

Figure 1. a.) Probability of measuring j when r is a perfect divisor of 2^{2L}. b.) Peaks become slightly broader if r is not a perfect divisor of 2^{2L}.

$$s(L,r) = \sum_{\{useful\ j\}} Pr(j,L,r), \tag{6}$$

where $\{$useful $j\}$ is the set, $\{j = \lfloor c2^{2L}/r \rfloor, \lceil c2^{2L}/r \rceil, 0 < c < r\}$ and $\lfloor\ \rfloor$ $\lceil\ \rceil$ denote rounding down and up respectively. Using this definition of s we can determine the period r provided we can run the algorithm $O(1/s)$ times.

3. DECOMPOSING SHOR'S ALGORITHM

The first step in designing an appropriate circuit to implement the QPF subroutine requires breaking it into a series of four steps that can be implemented as quantum circuits. 1. Hadamard transform. 2. Modular exponentiation. 3. Quantum Fourier transform. 4. Measurement. Steps 1,3 and 4 can be implemented quite easily, while the modular exponentiation step is the only one that requires further decomposition. The calculation of $f(k) = x^k \text{mod} N$ is firstly broken up into a series of controlled modular multiplications.

$$f(k) = \prod_{i=0}^{2L-1} (m^{2^i k_i} \text{mod} N), \tag{7}$$

where k_i denotes the ith bit of k. If $k_i = 1$ the multiplication $m^{2^i} \mod N$ occurs, and if $k_i = 0$ nothing happens.

There are many different ways to implement controlled modular multiplication.[6–9] The methods of Ref.[8] require the fewest qubits and will be used in this paper. To illustrate how each controlled modular multiplication proceeds, let $a(i) = m^{2^i} \mod N$ and

$$x(i) = \prod_{j=0}^{i-1} (m^{2^j k_j} \mod N). \tag{8}$$

$x(i)$ represents a partially completed modular exponentiation and $a(i)$ the next term to multiply by. Let $|x(i), 0\rangle$ denote a quantum register containing $x(i)$ and another of equal size containing 0. Firstly, add $a(i)$ modularly multiplied by the first register to second register if and only if (iff) $k_i = 1$.

$$|x(i), 0\rangle \longrightarrow |x(i), 0 + a(i)x(i) \mod N\rangle = |x(i), x(i+1)\rangle. \tag{9}$$

Secondly, swap the registers iff $k_i = 1$.

$$|x(i), x(i+1)\rangle \longrightarrow |x(i+1), x(i)\rangle \tag{10}$$

Thirdly, subtract $a(i)^{-1}$ modularly multiplied by the first register from the second register iff $k_i = 1$.

$$|x(i+1), x(i)\rangle \longrightarrow |x(i+1), x(i) - a(i)^{-1}x(i+1) \mod N\rangle = |x(i+1), 0\rangle. \tag{11}$$

Note that while nothing happens if $k_i = 0$, by the definition of $x(i)$ the final state in this case will still be $|x(i+1), 0\rangle$.

The first and third steps described in the previous paragraph are further broken up into series of controlled modular additions and subtractions respectively.

$$0 + a(i)x(i) = 0 + \sum_{j=0}^{L-1} a(i)2^j x(i)_j \mod N, \tag{12}$$

$$x(i) - a(i)^{-1}x(i+1) = x(i) - \sum_{j=0}^{L-1} a(i)^{-1}2^j x(i+1)_j \mod N, \tag{13}$$

where $x(i)_j$ and $x(i+1)_j$ denote the jth bit of $x(i)$ and $x(i+1)$ respectively. Note that the additions associated with a given $x(i)_j$ can only occur if $x(i)_j = 1$ and similarly for the subtractions. Given that these additions and subtractions form a multiplication that is conditional on k_i, it is also necessary that $k_i = 1$. Further decomposition will be left for subsequent sections.

4. QUANTUM FOURIER TRANSFORM

The first circuit that needs to be described, as it will be used in all subsequent circuits, is the QFT.

$$|k\rangle \rightarrow \frac{1}{\sqrt{2^L}} \sum_{j=0}^{2^L-1} \exp(2\pi i jk/2^L)|j\rangle \tag{14}$$

Figure (2a) shows the usual circuit design for an architecture that can interact arbitrary pairs of qubits. Figure (2b) shows the same circuit rearranged with the aid of swap gates to allow it to be implemented on an LNN architecture. Dashed boxes indicate compound gates that can be implemented via the canonical decomposition.[15–17] Note that the general circuit inverts the most significant to least significant ordering of the qubits whereas the LNN circuit does not. Counting compound gates as one, the total number of gates required to implement a QFT on both the general and LNN

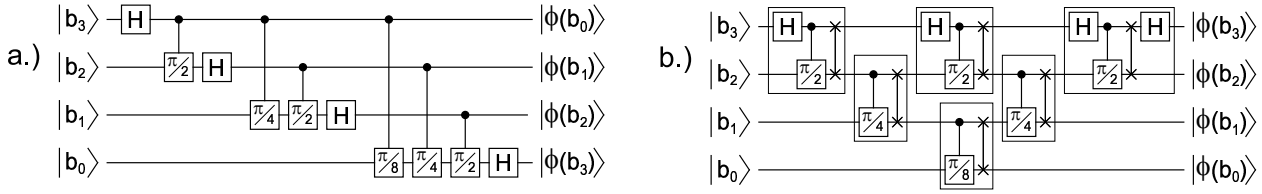

Figure 2. a.) Standard quantum Fourier transform circuit. b.) An equivalent linear nearest neighbour circuit.

architectures is $L(L-1)/2$. Assuming gates can be implemented in parallel, the minimum circuit depth for both is $2L-3$.

5. MODULAR ADDITION

Given a quantum register containing an arbitrary superposition of binary numbers, there is a particularly easy way to add a binary number to each number in the superposition.[8] By quantum Fourier transforming the superposition, the addition can be performed simply by applying appropriate single-qubit rotations as shown in figure (3a). Such an addition can also very easily be made dependent on a single control qubit as shown in figure (3b).

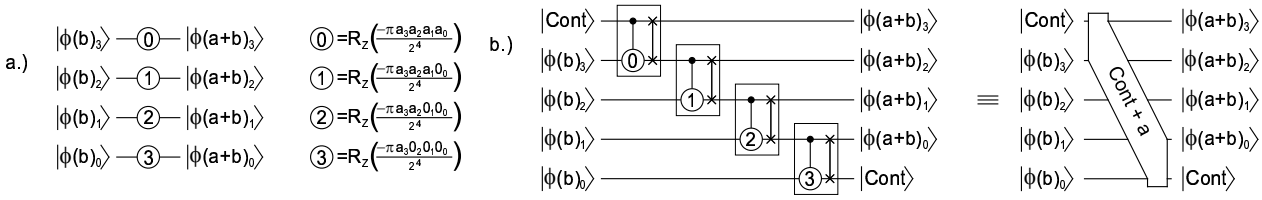

Figure 3. a.) Quantum Fourier addition. b.) Controlled quantum Fourier addition and its symbolic equivalent circuit.

Performing controlled modular addition is considerably more complicated as shown in figure (4). This circuit adds $2^j m^{2^i} \mod N$ to the register containing $\phi(b)$ to obtain $\phi(c) = \phi(b + 2^j m^{2^i} \mod N)$ iff both $x(i)_j$ and k_i are 1. The first five gates comprise a Toffoli gate that sets $kx = 1$ iff $x(i)_j = k_i = 1$. k_i and $x(i)_j$ are defined in equation (8) and equations (12-13) respectively. Note that the Beauregard circuit does not have a kx qubit, but without it the singly-controlled Fourier additions become doubly-controlled and take four times as long. The calculations of the gate count and circuit depth of the Beauregard circuit presented in this paper have therefore been done with a kx qubit included.

The next circuit element firstly adds $2^j m^{2^i} \mod N$ iff $kx = 1$ then subtracts N. If $b + (2^j m^{2^i} \mod N) < N$, subtracting N will result in a negative number. In a binary register, this means that the most significant bit will be 1. The next circuit element is an inverse QFT which takes the addition result out of Fourier space and allows the most significant bit to be accessed by the following CNOT. The MS (Most Significant) qubit will now be 1 iff the addition result was negative. If $b + (2^j m^{2^i} \mod N) > N$, subtracting N will yield the positive number $(b + 2^j m^{2^i}) \mod N$ and the MS qubit will remain set to 0.

We now encounter the first circuit element that would not be present if interactions between arbitrary pairs of qubits were possible. Note that while this "long swap" operation technically consists of L regular swap gates, it only increases

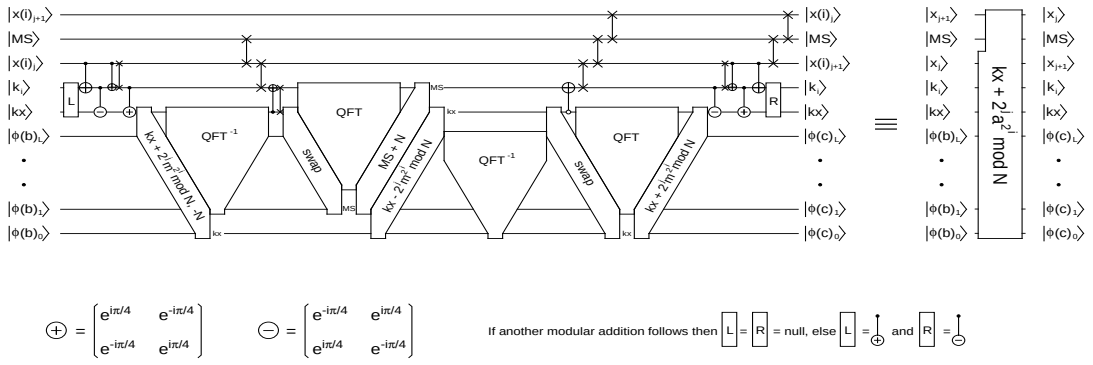

Figure 4. Circuit to compute $c = (b + 2^j m^{2^i}) \bmod N$. The diagonal circuit elements labelled swap represent a series of 2-qubit swap gates. Small gates spaced close together represent compound gates. The qubits $x(i)$ are defined in equation (8) and essentially store the current partially calculated value of the modular exponentiation that forms the heart of Shor's algorithm. The MS (Most Significant) qubit is used to keep track of the sign of the partially calculated modular addition result. The k_i qubit is the ith bit of k in equation (2). The kx qubit is set to 1 if and only if $x(i)_j = k_i = 1$.

the depth of the circuit by 1. The subsequent QFT enables the MS controlled Fourier addition of N yielding the positive number $(b + 2^j m^{2^i}) \bmod N$ if $MS = 1$ and leaving the already correct result unchanged if $MS = 0$.

While it might appear that we now done, the qubits MS and kx must be reset so they can be reused. The next circuit element subtracts $2^j m^{2^i} \bmod N$. The result will be positive and hence the most significant bit of the result equal to 0 iff the very first addition $b + (2^j m^{2^i} \bmod N)$ gave a number less than N. This corresponds to the $MS = 1$ case. After another inverse QFT to allow the most significant bit of the result to be accessed, the MS qubit is reset by a CNOT gate that flips the target qubit iff the control qubit is 0. Note that the long swap operation that occurs in the middle of all this to move the kx qubit to a more convenient location only increases the depth of the circuit by 1.

After adding back $2^j m^{2^i} \bmod N$, the next few gates form a Toffoli gate that resets kx. The final two swap gates move $x(i)_{j+1}$ into position ready for the next modular addition. Note that the L and R gates are inverses of one another and hence not required if modular additions precede and follow the circuit shown. Only one of the final two swap gates contributes to the overall depth of the circuit.

The total gate count of the LNN modular addition circuit is $2L^2 + 8L + 22$ and compares very favourably with the general architecture gate count of $2L^2 + 6L + 14$. Similarly, the LNN depth is $8L + 16$ versus the general depth of $8L + 13$.

6. CONTROLLED SWAP

Performing a controlled swap of two large registers is slightly more difficult when only LNN interactions are available. The two registers need to be meshed so that pairs of equally significant qubits can be controlled-swapped. The mesh circuit is shown in figure (5). This circuit element would not be required in a general architecture.

After the mesh circuit has been applied, the functional part of the controlled swap circuit (figure (6)) can be applied optimally with the control qubit moving from one end of the meshed registers to the other. The mesh circuit is then applied in reverse to UN-tangle the two registers. The gate count and circuit depth of a mesh circuit is $L(L-1)/2$ and $L-1$ respectively. The corresponding equations for a complete LNN controlled swap are $L^2 + 5L$ and $6L$. The general controlled swap only requires $6L$ gates and can be implemented in a circuit of depth $4L + 2$. The controlled swap is the only part of this implementation of Shor's algorithm that is significantly more difficult to implement on an LNN architecture.

7. MODULAR MULTIPLICATION

The ideas behind the modular multiplication circuit of figure (7) were discussed in section 3. The first third comprises a controlled modular multiply (via repeated addition) with the result being stored in a temporary register. The middle third implements a controlled swap of registers. The final third resets the temporary register. Note that the main way in which the

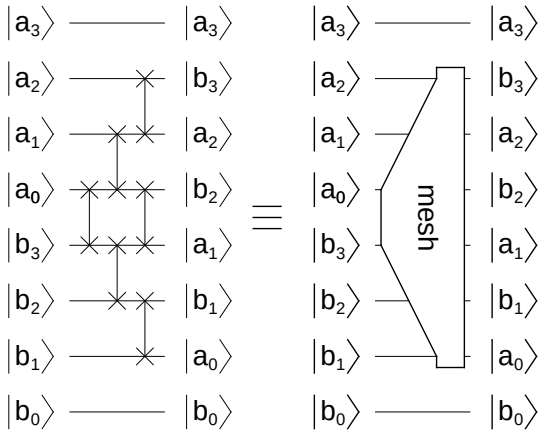

Figure 5. Circuit designed to mesh two quantum registers.

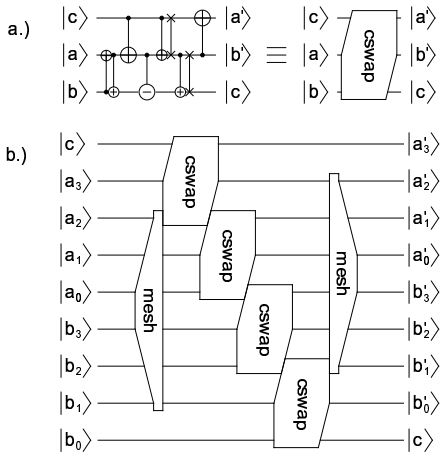

Figure 6. a.) LNN circuit for the controlled swapping of two qubits $|a\rangle$ and $|b\rangle$. The qubits $|a'\rangle$ and $|b'\rangle$ represent the potentially swapped states. b.) LNN circuit for the controlled swap. The effective depth of the cswap gate is 4.

performance of the LNN circuit differs from the ideal general case is due to the inclusion of the two mesh circuits. Nearly all of the remaining swaps shown in the circuit do not contribute to the overall depth. Note that the two swaps drawn within the QFT and inverse QFT are intended to indicate the appending of a swap gate to the first and last compound gates in these circuits respectively. The total gate count for the LNN modular multiplication circuit is $4L^3 + 20L^2 + 58L - 2$ versus the general gate count of $4L^3 + 13L^2 + 35L + 4$. The LNN depth is $16L^2 + 40L - 7$ and the general depth $16L^2 + 33L - 6$.

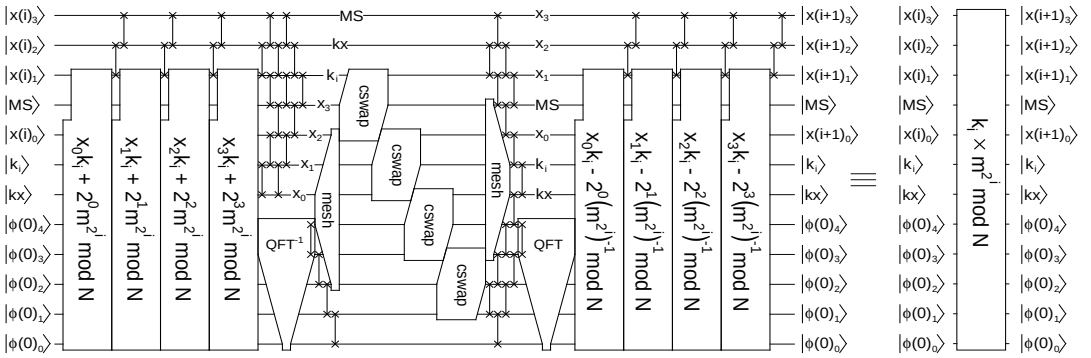

Figure 7. Circuit designed to modularly multiply $x(i)$ by m^{2^i} if and only if $k_i = 1$. Note that for simplicity the circuit for $L = 4$ has been shown. Note that the bottom $L + 1$ qubits are ancilla and as such start and end in the $|\phi(0)\rangle$ state. The swap gates within the two QFT structures represent compound gates.

8. COMPLETE CIRCUIT

The complete circuit for Shor's algorithm (figure (8)) can best be understood with reference to figure (2a) and the five steps described in section 2. The last two steps of Shor's algorithm are a QFT and measurement of the qubits involved in the QFT. When a 2-qubit controlled quantum gate is followed by measurement of the controlled qubit, it is equivalent to measure the control qubit first and then apply a classically controlled gate to the target qubit. If this is done to every qubit in figure (2a), it can be seen that every qubit is decoupled. Furthermore, since the QFT is applied to the k register and the k register qubits are never interacted with one another, it is possible to arrange the circuit such that each qubit

in the k register is sequentially used to control a modular multiplication, QFTed, then measured. Even better, after the first quit of the k register if manipulated in this manner, it can be reset and used as the second qubit of the k register. This one qubit trick[18] forms the basis of figure (8). The total number of gates required in the LNN and general cases are $8L^4+40L^3+116\frac{1}{2}L^2+4\frac{1}{2}L-2$ and $8L^4+26L^3+70\frac{1}{2}L^2+8\frac{1}{2}L-1$ respectively. The circuit depths are $32L^3+80L^2-4L-2$ and $32L^3 + 66L^2 - 2L - 1$ respectively. The primary result of this paper is that the gate count and depth equations for both architectures are identical to first order.

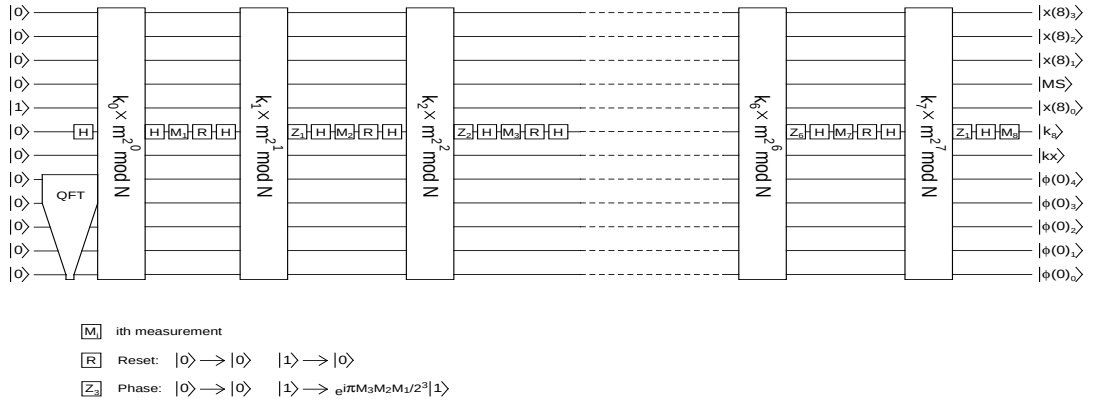

Figure 8. Circuit designed to implement the QPF subroutine. The single-qubit gates interleaved between the modular multiplications comprise a QFT that has been decomposed by using measurement gates to remove the need for controlled quantum phase rotations. Note that without these single-qubit gates the remaining circuit is simply modular exponentiation.

9. ERROR MODELS AND ANALYSIS

The error model used in this investigation was the set of discrete errors in which a single qubit $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ can experience a bit flip $X|\phi\rangle$, a phase flip $Z|\phi\rangle$, or both at the same time $XZ|\phi\rangle$. Each of the three types of discrete error gates has an equal probability $p/3$ of occurring [figure (9)]. Hence, we can examine how the success of the QPF subroutine

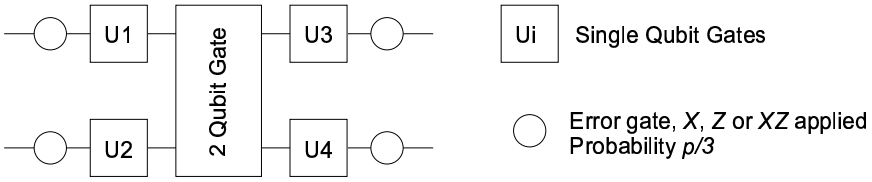

Figure 9. Error model used in simulation. All single qubit gates are combined with neighbouring two qubit gates. After each two qubit interaction, X,Z or XZ gates are applied to each qubit with equal probability $p/3$

s, depends on the probability of a discrete error occurring p, for various values of L. Within the simulations we assume perfect gate operations. All two qubit gates are approximated as having the same operational times (defined as a single time step) and all single qubit gates combined with neighbouring two qubit gates via the canonical decomposition.[15–17] Discrete error gates are then applied with probability p to each qubit after each time step.

Computational times are unfortunately the major restriction on how large a circuit we can simulate. The stochastic nature of both the algorithm itself and more importantly, the application of discrete error gates required a large number of statistical runs in order to calculate a consistent average value of s for each particular value of p. Errors may or may not occur in places within the circuit that effect the final result. Hence, for a given number of statistical runs the variance on

this set can be quite large. To see this in a more explicit way we can examine figure (10) which shows the effect of a single bit flip error on the value of s for the $L = 5$ circuit. Each horizontal block represents one of the 14 qubits, while each vertical slice represents a single time step where a bit flip error gate can occur. White areas indicate where the bit flip error has no effect on the success of the circuit, and successively darker regions show where bit flip errors begin to reduce the final value of s until the circuit completely fails. The image shown only represents a single modular multiplication gate and clearly shows the structure of the circuit shown earlier. This image makes it clear that the location of the error can play

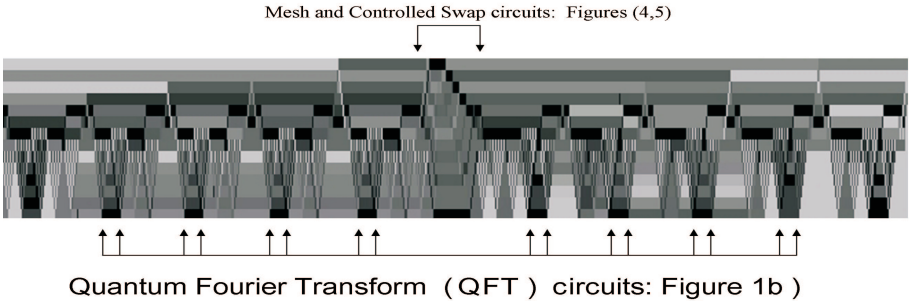

Figure 10. Map showing how the location of a single bit flip error plays a major role in the final output success of the circuit. This image is for L = 5 (14 qubits), and shows the first modular multiplication section of the circuit. Darker areas represent lower values for s.

a major role in the final value of s calculated, with various sections invariant to the bit flip error. The average value of s for this single section of the circuit when subjected to a single X error is $s = 0.34$.

10. SIMULATION RESULTS

The classical simulation algorithm employed to examine the QPF subroutine used a state vector representation. Matrix operations were performed in order to simulate both quantum gates and error operations. The total area of the circuit used (i.e. the number of possible locations where an error can occur) scales as n_p = number of qubits $\times$ depth $= (2L + 4) \times (32L^3 + 80L^2 - 4L - 2)$. The simulation results shown investigate the behaviour of the QPF subroutine at low discrete error rates, (i.e. $p \approx 1/n_p$). The simulations were performed in a half-stochastic, half-deterministic manner. We allowed the type and location of discrete errors to occur at random, however we now specified exactly how many errors could occur within a given run of the subroutine. For each value of L we found the maximum number of discrete errors that can occur such that the QPF subroutine just produces non-random output. For each value of L, the circuit produces random output when each j value has an equal probability of being measured (given by $P = 1/2^{2L}$). A more detailed investigation into the error behaviour for the QPF subroutine can be found in Ref..[5] Each $f(k)$ used for this section is shown in table (2). Within simulations we only examined the probability of obtaining the specific useful value $j = \lfloor 2^{2L}/6 \rfloor$. The error

$2L + 4$	$f(k) = x^k \text{mod} N$, with $r = 6$
14	8^kmod27
16	31^kmod63
18	10^kmod77
20	27^kmod247

Table 2. Functions used for various values of L. Note that for $2L + 4 = 14, 16, 22$ the functions used are not products of two primes. With some slight modifications to the classical post-processing, Shor's algorithm can still be used to factor such numbers. Since we are only investigating the reliability of the QPF subroutine, this is not relevant to our analysis.

behaviour of the circuit does not change by examining different values of j or r (refer to Ref.[5] for a more detailed analysis of how errors cause the circuit to fail). Figure (11) show the results of simulations. Each data point on this plot is the average value of a number of statistical runs as explained in section 9. Figure (11) shows us quite conclusive behaviour for the QPF subroutine at small error rates. Both the LNN and non-LNN will fail if between 10 and 40 errors occur (depending

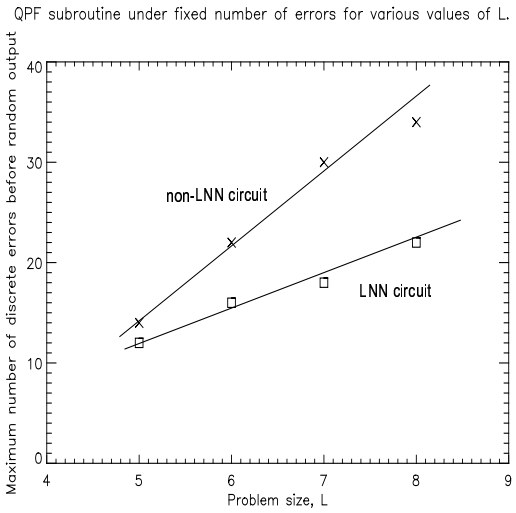

Figure 11. Plot detailing the maximum number of discrete errors possible before the QPF subroutine produces random output for various values of L. Both the LNN and non-LNN circuits are shown, with a slight increase in stability for the non-LNN plot.

on the value of L). These results show that if we require the QPF subroutine to produce non random output the physical error rate of the circuit must be kept approximately $p = \zeta/n_p$. Where ζ is a linear function of L. To maintain polynomial runtime of the QPF subroutine p cannot increase linearly with L, since the random output of the subroutine scales as $O(1/2^{2L})$. As a rough approximation an error rate of $p = 10/n_p$ should be needed for large values of L. The non-LNN circuit used in these simulations were conducted in an identical manner to the LNN circuit. As expected, the the error behaviour is slightly more robust than the LNN circuit. This can be attributed to the slight difference in both circuits and to the statistical nature in how the above results are simulated.[5] Additional swap operations that are generally canonically decomposed with neighbouring gates do not significantly change the succeptability of the LNN circuit.

11. IMPLICATIONS FOR QEC

Our simulations have shown that the QPF subroutine is highly sensitive to discrete errors. If we assume that the error behaviour for large values of L follows the same trend shown in these simulations, the maximum value for p such that polynomial run-time is preserved appears to be at most one order of magnitude above the $1/n_p$ lower bound. Using this information, we can now take a brief look at the demands on QEC. If we utilise the theoretical scaling behaviour[2] of k-level concatenated error correction codes such as the 7 qubit Steane code,[19] and assume that our quantum computer operates at a physical discrete error rate of approximately $p = 10^{-5}$, we can construct a table showing, for large values of L, how many layers of error correction are needed and the minimum number of physical qubits required. Note that the actual physical

L	p_R	$k(c = 10^4)$	$Q(c = 10^4)$	$k(c = 10^3)$	$Q(c = 10^3)$	$k(c = 10^2)$	$Q(c = 10^2)$
64	9.3×10^{-9}	3	45,276	2	6,468	2	6,468
128	5.8×10^{-10}	3	89,180	2	12,740	2	12,740
256	3.6×10^{-11}	3	176,988	3	176,988	2	25,284
512	2.3×10^{-12}	4	2,468,228	3	352,604	2	50,372
1024	1.4×10^{-13}	4	4,926,852	3	703,836	2	100,548

Table 3. Table showing QEC requirements for the QPF subroutine. L denotes the binary length of the number to be factored. p_R is the required error rate on each logical qubit $p_R = 10/64L^4$. k is the number of levels of concatenated correction. p_{logical} is the actual logical error rate for k levels of concatenated QEC using the scaling relationship $p_{\text{logical}} = (cp)^{2^k}/c$ with $p = 10^{-5}$ and $\{c = 10^4, c = 10^3, c = 10^2\}$. Q is the minimum number of qubits required, $Q = (2L + 4)7^k$.

error thresholds and scaling behaviour for the 7 qubit code is based on calculations used for a correction circuit utilising

arbitrary interactions. A LNN circuit has been simulated for a 5 qubit correction code demonstrating scaling behaviour for certain values of the physical error rate.[20] Simulations need to be conducted for the 7 qubit Steane code for an appropriate LNN circuit in order to estimate if the values calculated in table (3) remain accurate. Table (3) shows, using our current simulation data that in order to factorise numbers of binary length $L = 128$ upwards, we require approximately 2^{nd} to 3^{rd} level concatenated error correction. Since the error rate required for each logical qubit is highly dependent on the $1/n_p$ lower bound, by minimising the area of the circuit used for the QPF subroutine it may be possible to raise this lower bound and reduce the concatenation level to 1^{st} order. Table (3) shows the required levels of concatonation for various values of the threshold c. Careful circuit design is clearly needed in order to reduce this value as much as possible.

12. IMPLEMENTING LARGE SCALE ALGORITHMS.

The circuit and simulations detailed in this paper show that the implementation of large scale quantum algorithms is an extremely complicated task. Qualitatively we can consider some of the issues related to implementing Shor's algorithm on an actual quantum computer when attempting to factor numbers of large L using the techniques of QEC and Fault-Tolerant computation.[2] The results of the previous simulations demonstrate that in order to factor a 128-bit number 3^{rd} order concatenated error correction will be required (assuming $p = 10^{-5}$ and $c = 10^4$). There are several important issues that need to be addressed if we are to implement Fault-Tolerant QEC for these circuits. Construction of circuits appropriate for large values of L require extremely fine rotation gates within the Quantum Fourier Transform. Also when employing the 7 qubit code, arbitrary single and two qubit gates need to be constructed from a universal set of gates that can be implemented Fault-Tolerantly on encoded data. We will first begin with the small angle approximation when constructing circuits to implement the Quantum Fourier Transform (QFT). In order to construct an appropriate circuit to factorise a $L = 128$-bit number we require single and controlled phase rotations of order $\pi/(10^{38})$. This magnitude of control for a phase rotation is clearly impractical and generally unnecessary. Previous work[10] has examined the resultant stability when we replace the final QFT needed in Shor's algorithm (which is performed using the 'one qubit trick' in the circuit presented here) with an Approximate Quantum Fourier transform (AQFT) which simply removes these small angle rotations. It was shown that the algorithm still results in robust output when we remove all rotations finer than $\pi/128$. The need to construct arbitrary phase gates required by the AQFT is highly nontrivial when considering a circuit that requires the property of Fault-Tolerance. If we are considering qubits encoded by the 7 qubit code, then a useful universal set of gates are $H, X, Z, T, T^\dagger, S, S^\dagger$ and $CNOT$. Where S and T gates are given by,

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad\qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}. \tag{15}$$

This set of universal gates is useful since the 7 qubit code allows for straightforward Fault-Tolerant application of all gates except for T and $T^\dagger$. Using this set of universal operations, any single qubit gate can be constructed from $H, X, Z, T, T^\dagger, S$ and $S^\dagger$ through the Solovay-Kitaev theorem.[2,21] Given that any single qubit gate can be constructed from this set, the inclusion of a two qubit CNOT operation allows for the construction of any arbitrary two-qubit operation.[22,23] Recent work[10] has shown that in order to construct these arbitrary phase gates required by the AQFT, long gate sequences are often needed to generate gates that are a better approximation to the desired rotation than the identity gate. When examining these gate sequences it also becomes apparent that T and $T^\dagger$ gates are crucial in constructing arbitrary single qubit unitaries. The repeated presence of these gates in a sequence results in a large obstacle in relation to efficient Fault-Tolerant construction of arbitrary rotation gates from this universal set. As mentioned T and $T^\dagger$ gates are not simple gates to implement Fault-Tolerantly using the 7 qubit QEC code. Fault-Tolerant circuits for these gates are highly complex. Requiring a minimum of 19 qubits with circuit depths into the hundreds.[2] A single, approximate $\pi/128$ rotation gate (Which is required numerous times during Shor's algorithm for large L) requires a 46 gate sequence minimum, 23 of which are either T or $T^\dagger$. It seems implausible that practically implementing a circuit that requires three concatenated levels of this type of encoding is realistic in the near future. A much more realistic prospect is to abandon the concept of Fault-Tolerant computation for the time being and examine the applicability of either straight non-Fault-Tolerant computation, or a combination of Fault-Tolerant encoding and non-Fault-Tolerant operations when equivalent Fault-Tolerant operations are simply to costly to be effective. For example, some recent simulations[24] have shown that non-Fault-Tolerant encoding of a single logical $|0\rangle$ is far more reliable for realistic error rates. Further work is currently underway in order to investigate in more detail whether the concept of rigorous Fault-Tolerant QEC is truly practical for large scale quantum algorithms, at least in the near future.

13. CONCLUSION

In conclusion we have presented a circuit capable of implementing the QPF subroutine on a linear nearest neighbour array of qubits. To leading order our circuit involves $8L^4$ gates arranged in a circuit of depth $32L^3$ on $2L + 4$ qubits. Figures identical to that possible when interactions between arbitrary pairs of qubits are allowed. Detailed simulations show that the QPF subroutine for Shor's algorithm is highly sensitive errors. We have demonstrated that on average the algorithm can only tolerate discrete error rates approximately one order of magnitude above the single error lower bound for both the LNN circuit presented and the original Beauregard circuit. This suggests that substantial quantum error correction is necessary for factoring problems large enough to be interesting. Error rates of physical gate operations and operational times of complete circuits (with error correction) will need to be investigated further in order to be confident that a physical implementation of the QPF subroutine and hence Shor's algorithm is possible and performs in a manner that is practical.

14. ACKNOWLEDGEMENTS

The authors would like to thank the staff at the Melbourne centre for high performance computing. This work was supported by the Australian Research Council, and the Army Research Office under contract DAAD19-01-1-0653

REFERENCES

1. P. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logatithms on a Quantum Computer," *Society for Industrial and Applied Mathematics* **26**, p. 1484, 1997.
2. Nielsen and Chuang, *Quantum Computation and Quantum Information*, Cambridge, Second ed., 2000.
3. ARPA, "Quantum information science and technology roadmap project," *http://qist.lanl.gov* , 2004.
4. A.Fowler, S. Devitt, and L. Hollenberg, "Implementation of Shor's algorithm on a linear nearest neighbour qubit array," *Quantum Information and Computation* **4**, p. 237, 2004.
5. S.J.Devitt, A.G.Fowler, and L.C.L.Hollenberg, "Simulations of Shor's algorithm with implications to scaling and Qauntum Error Correction," *quant-ph/0404081* , 2004.
6. V.Vedral, A.Barenco, and A.Ekert, "Quantum Networks for Elementary Arithmetic Operations," *Phys. Rev. A* **54**, p. 147, 1996.
7. P. Gossett, "Quantum carry save arithmetic," *quant-ph/9808061* , 1998.
8. S. Beauregard, "Circuit for Shor's algorithm using 2n+3 qubits," *Quantum Information and Computation* **3**, p. 175, 2003.
9. C.Zalka, "Fast versions of Shor's Quantum Factoring algorithm," *quant-ph/9806084* , 1998.
10. A.Fowler and L. Hollenberg, "Scalability of Shor's algorithm with a limited set of rotation gates," *accepted by Physical Review A, quant-ph/0306018* .
11. L.F.Wei, X.Li, X.Hu, and F.Nori, "Phase-Matching approach to eliminate the dynamical phase error in Shor's algorithm," *quant-ph/0305039* , 2003.
12. D.Braun, "Quantum chaos and quantum algorithms," *quant-ph/0110037* , 2001.
13. E.Knill, R.Laflamme, H.N.Barnum, D.A.Dalvit, J.J.Dziarmaga, J.E.Gubernatis, L.Guruits, G.Ortiz, and W.H.Zurek, "From factoring to Phase Estimation," *Los Alamos Science* **27**, p. 38, 2002.
14. C. Lavor, L. Manssur, and R.Portugal, "Shor's Algorithm for Factoring Large Integers," *quant-ph/0303175* , 2003.
15. J.Zhang, J.Vala, S.Sastry, and K.B.Whaley, "Geometric theory of non-local two qubit operators," *Phys. Rev. A* **67**, p. 042313, 2003.
16. B. Kraus and J. Cirac, "Optimal creation of entanglement using two-qubit gate," *Phys. Rev. A* **63**, p. 062309, 2001.
17. Y. Makhlin, "Nonlocal properties of two-qubit gates and mixed states and optimization of quantum computers," *Quantum Information Processing* **1**, p. 243, 2002.
18. S. Parker and M. Plenio, "Efficient factorization with a single pure qubit and logN mixed qubits," *Phys. Rev. Lett.* **85**, p. 3049, 2000.
19. A. M. Steane, "Error correcting codes in quantm theory," *Phys. Rev. Lett.* **77**, p. 793, 1996.
20. A. Fowler, L. Hollenberg, and C. Hill, "Quantum error correction on linear nearest neighbour qubit arrays," *Phys. Rev. A* **69**, p. 042314, 2004.
21. A. Kitaev *Russ. Math. Surv.* **53**, p. 1991, 1997.

22. S. Bullock and I. Markov, “An arbitrary two-qubit computation in 23 elementary gates,” *Phys. Rev. A* **68**, p. 012318, 2003.
23. F. Vatan and C. Williams, “Optimal quantum circuits for general two-qubit gates,” *Phys. Rev. A* **69**, p. 032513, 2004.
24. S. Devitt, A.Fowler, and L. Hollenberg, “Practicallity of Fault-Tolerant Quantum Computation,” *In Preperation* , 2004.