acta physica slovaca vol. 45 No. 3, 205 – 216 June 1995

QUANTUM COMPUTATION[1]

Adriano Barenco, Artur Ekert[2]
Clarendon Laboratory, University of Oxford, Parks Road Oxford OX1 3PU, U.K.

Received 25 April 1995, accepted 10 May 1995

We describe how physics of computation determines computational complexity. In particular we show how quantum phenomena lead to qualitatively new modes of computation. The power of quantum computation is illustrated by Shor's quantum factoring algorithm.

1. Computation and Physics

Computers are physical objects and computations are physical processes. Quantum computers are machines that rely on characteristically quantum phenomena, such as quantum interference and quantum entanglement in order to perform computation.

The classical theory of computation usually does not refer to physics and as the result it is often falsely assumed that its foundations are self-evident and purely abstract. Although in the sixties Landauer [1] pointed out the physical nature of information, it was not until the first works on quantum computation by Deutsch [2] and Feynman [3] that the fundamental connection between the laws of physics and computation was properly emphasised. Recent developments in the theory of quantum computational complexity [4-7] provide a vivid example of this connection.

Computers solve problems following a precise set of instructions that can be mechanically applied to yield the solution to any given instance of a particular problem. A specification of this set of instruction is called an algorithm. Examples of algorithms are the procedures taught in elementary schools for adding and multiplying whole numbers; when these procedures are mechanically applied, they always yield the correct result for any pair of whole numbers. However, any operation on numbers is performed by physical means and what can be done to a number depends on the physical representation of this number and the underlying physics of computation. For example, when numbers are encoded in quantum states then quantum computers, *i.e.* physical devices whose unitary dynamics can be regarded as the performance of computation, can accept states which represent a coherent superposition of many different numbers

[1]Presented at the **3rd central-european workshop on quantum optics**, Budmerice castle, Slovakia, 28 April - 1 May, 1995

[2]World Wide Web: `http://eve.physics.ox.ac.uk/QChome.html`

(inputs) and evolve them into another superposition of numbers (outputs). In this case computation, *i.e.* a sequence of unitary transformations, affects simultaneously each element of the superposition allowing a massive parallel data processing albeit within one piece of quantum hardware. As the result quantum computers can efficiently solve some problems which are believed to be intractable on any classical computer [2,4-7].

In this paper we illustrate the relevance of the underlying physics of computation by describing the difference between classical and quantum computation. We have chosen *factorisation* as an example that sets the efficiency of quantum computation ahead of any classical data processing. Indeed the contrast is striking. Shor [7] has recently shown that quantum computers can efficiently factor big integers and compute discrete logarithms. These tasks are believed to be intractable on any classical computer !

Finally let us mention also that factorisation is not of purely academic interest. It is the problem which underpins security of many classical public key cryptosystems, for example, RSA [8] — the most popular public key cryptosystem named after the three inventors, Rivest, Shamir, and Adleman — gets its security from the difficulty of factoring large numbers. Hence for the purpose of cryptoanalysis the experimental realisation of quantum computation is a most interesting issue.

2. Complexity of Factoring

Consider an integer N with L decimal digits. Factoring N means finding its prime factors *i.e.* finding whole numbers $\{p\}$ such that any p divides N with the remainder 0.

One way to calculate the prime factors is to try to divide N by $2, 3, \ldots \sqrt{N}$ and to check the reminder. This method is very time consuming. It requires about $\sqrt{N} \approx 10^{L/2}$ divisions, hence the time it takes to execute this algorithm increases exponentially with L. Even if the computer can perform as much as 10^{10} divisions per second it would take about a second to factor a 20 digit number, about a year to factor a 34 digit number and more than the estimated age of the Universe (10^{17}s) to factor a 60 digit long number!

Although the problem of finding an efficient algorithm for factoring large numbers was worked on by famous mathematicians such as Fermat and Legendre, only recently (since the invention of public key cryptosystems in the seventies) a significant progress has been made in designing good factoring algorithms. The best algorithms such as the Multiple Polynomial Quadratic Sieve [9] and the Number Field Sieve [10] have an execution time that grows as a subexponential function of L ($\sim \exp[L^{1/3}]$). Although they are much faster than the trial division method still they cannot be regarded as efficient algorithms.

For an algorithm to be efficient (and usable), the time it takes to execute the algorithm must increase no faster than a polynomial function of the size of the input (L in our case). If the best algorithm we know for a particular problem has the execution time (viewed as a function of the size of the input) bounded by a polynomial then we say that the problem belongs to class P. Problems outside class P are known as *hard* problems. Thus we say, for example, that multiplication is in P whereas factoring is not in P and that is why it is a hard problem (for more information about computational complexity see for example [11] or [12]).

The snag is that the complexity classes such as P are defined with respect to classical computation. Classical algorithms for factorisation are not known to belong to P, however, there exists an efficient quantum factoring algorithm [7] !

3. Quantum Registers

We start our description of quantum computation with the basic unit of information namely a single bit. If a physical object can be put into two different, distinguishable states then this object can represent two different numbers. We call any two–state system a physical bit; when the system is quantum and the two states are two orthogonal *quantum* states, we refer to it as a *quantum bit* or simply a *qubit*. Any two–state quantum system is a potential candidate for a qubit. Both a single classical bit and a qubit can represent at most two different numbers, however, qubits are different because apart from the two orthogonal basis states, which we label as $|0\rangle$ and $|1\rangle$, they can also be put into infinitely many other states of the form $|\Psi\rangle = c_0|0\rangle + c_1|1\rangle$.

Let us mention in passing that although a qubit can be prepared in an infinite number of different quantum states it cannot be used to transmit more that one bit of information. This is because no detection process can reliably differentiate between nonorthogonal states [13-16]. However, information encoded in nonorthogonal states and in quantum entanglement can be used in systems known as quantum cryptography [17-20] or quantum teleportation [21].

Consider now a register composed of m physical qubits. There are 2^m different orthogonal quantum states of this register therefore the register can represent 2^m different numbers, *e.g.* from 0 to $2^m - 1$. The most general (pure) state of this register can be written as

$$|\Psi\rangle = \sum_x c_x |x\rangle, \qquad (1)$$

where number x is represented in the register in binary form

$$|x\rangle = |x_{m-1}\rangle \otimes |x_{m-2}\rangle \otimes \ldots |x_1\rangle \otimes |x_0\rangle \qquad (2)$$

according to the decomposition

$$x = \sum_{i=0}^{m-1} 2^i x_i, \qquad x_i = 0 \text{ or } 1. \qquad (3)$$

Note that Eq. 1 describes the state in which several different values of the register are present *simultaneously*; this quantum feature has no classical counterpart. In order to prepare a specific number in the register we have to perform m elementary operations on each qubit to set it into one of the two orthogonal state $|0\rangle$ or $|1\rangle$. However, in quantum computers m elementary unitary transformations performed bit by bit can also prepare the register in a coherent superposition of all 2^m numbers that can be stored in the register. Take the register initially in $|0\rangle \otimes |0\rangle \otimes \ldots |0\rangle$ state and apply the unitary operation

$$A = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad (4)$$

to each qubit. The resulting state of the register is an equally weighted superposition of all 2^m numbers,

$$|\Psi\rangle = \overbrace{A\,|0\rangle \otimes A\,|0\rangle \otimes \ldots A\,|0\rangle}^{m \text{ times}} = \frac{1}{2^{m/2}} \sum_{x=0}^{2^m-1} |x\rangle . \tag{5}$$

It is quite remarkable that in quantum registers m elementary operations can generate a state containing all 2^m possible numerical values of the register. In contrast, in classical registers m elementary operations can only prepare one state of the register representing one specific number. It is this ability of creating quantum superpositions which makes the "quantum parallel processing" possible. If after preparing the register in a coherent superposition of several numbers all subsequent computational operations are unitary (*i.e.* preserve the superpositions of states) then with each computational step the computation is performed simultaneuosly on all the numbers present in the superposition.

This type of computation is particularly useful for problems which, in classical case, involve performing the same computation several times for different input data. Estimating the period of a given function $f(x)$ is a typical example as it requires evaluating function $f(x)$ many times for different x.

4. Computing Functions

Let us describe now how quantum computers compute functions. For this we will need two quantum registers of length m and n. Consider a function

$$f:\ \{0,1,\ldots 2^m-1\} \longrightarrow \{0,1,\ldots 2^n-1\}, \tag{6}$$

where m and n are natural numbers.

A classical computer computes f by evolving each labeled input, $0,1,\ldots 2^m-1$ into a respective labeled output, $f(0), f(1), \ldots f(2^m-1)$. Quantum computers, due to the unitary (and therefore reversible) nature of their evolution, compute functions in a slightly different way. In order to compute functions which are not one–to–one and to preserve the reversibility of computation, quantum computers have to keep the record of the input. Here is how it is done.

We will use the two quantum registers; the first register to store the input data, the second one for the output data. Each possible input x is represented by $|x\rangle$ — the quantum state of the first register. Analogously, each possible output $y = f(x)$ is represented by $|y\rangle$ — the quantum state of the second register. Vectors $|x\rangle$ belong to the 2^m–dimensional Hilbert space $\mathcal{H}_1$, and vectors $|y\rangle$ belong to the 2^n–dimensional Hilbert space $\mathcal{H}_2$ ($\mathcal{H}_1$ and $\mathcal{H}_2$ are tensorial products of the Hilbert spaces of respectively m and n qubits). States corresponding to different inputs and different outputs are orthogonal, $\langle x|x'\rangle = \delta_{xx'}$, $\langle y|y'\rangle = \delta_{yy'}$. The function evaluation is then determined by the evolution of the two registers,

$$|x\rangle\,|0\rangle \xrightarrow{U_f} |x\rangle\,|f(x)\rangle . \tag{7}$$

We can always prepare specific x in the first register and read the value $f(x)$ from the second register. It was shown that as far as the computational complexity is concerned a reversible function evaluation, *i.e.* the one that keeps track of the input, is as good as a regular, irreversible evaluation [22]. This means that if a given function can be computed in polynomial time it can also be computed in polynomial time using a reversible computation. The computation we are considering here is not only reversible but also quantum and we can do much more than computing values of $f(x)$ one by one. We can prepare a superposition of all input values as a single state and by running the computation U_f *only once*, we can compute *all* of the 2^m values $f(0), \ldots, f(2^m-1)$,

$$\left(\frac{1}{2^{m/2}} \sum_{x=0}^{2^m-1} |x\rangle\right) |0\rangle \xrightarrow{U_f} \frac{1}{2^{m/2}} \sum_{x=0}^{2^m-1} |x\rangle\,|f(x)\rangle . \tag{8}$$

It looks too good to be true so where is the catch? How much information about f does the state

$$|f\rangle = \frac{1}{2^{m/2}}(|0\rangle\,|f(0)\rangle + |1\rangle\,|f(1)\rangle + \ldots + |2^m-1\rangle\,|f(2^m-1)\rangle) \tag{9}$$

contain?

Unfortunately no quantum measurement can extract all of the 2^m values $f(0)$, $f(1)$, … , $f(2^m-1)$ from $|f\rangle$. However, there are measurements that provide us with information about joint properties of all the output values $f(x)$ such as, for example, periodicity. We will see in the following sections, how a periodicity estimation can lead to fast factorisation.

5. Factoring and Periodicity

The factorisation problem is related to finding periods of certain functions. In particular one can show that finding factors of N is equivalent to finding a period of $f_N(x)$, where

$$f_N(x) = a^x \bmod N \tag{10}$$

and a is *any* randomly chosen number which is coprime with N. The result of this operation is the remainder after the division of a^x by N. The function is periodic and the period r, which depends on a and N, is called the order of a modulo N. For example, the increasing powers of 2 modulo 15 go like $1, 2, 4, 8, 1, 2, 4, 8, 1, \ldots$ and so on — the order of 2 modulo 15 is 4 (for more information see, for example [23]).

Knowing r *i.e.* the order of a modulo N, we can factor N provided r is even and $r \bmod N \neq -1$. When a is chosen randomly the two conditions are satisfied with probability greater than half. To factor N it is enough to calculate the greatest common divisor of $a^{r/2} \pm 1$ and N. Fortunately an easy and very efficient algorithm to compute the greatest common divisor has been known since 300 BC. The algorithm, known as the Euclidean algorithm, is described in Euclid's *Elements*, the oldest Greek treatise in mathematics to reach us in its entirety (try your elementary school textbooks as a reference). The result of taking this greatest common divisor, written as $(a^{r/2} \pm 1, N)$, is a factor of N.

To see how this method works let us consider a very simple example of factoring 15. Firstly we select a, such that $(a, N) = 1$; *i.e.* a could be any number from the set $\{2,4,7,8,11,13,14\}$. Let us pick up $a = 11$ and let us compute the order of 11 modulo 15. Values of 11^x mod 15 for $x = 1,2,3,\ldots$ go as $11,1,11,1,11,\ldots$ giving $r = 2$. Then we compute $a^{r/2}$ which gives 11 and we find the largest common factor $(11 \pm 1, N)$ *i.e.* $(10,15)$ and $(12,15)$ which gives 5 and 3, the two factors of 15. Respective orders modulo 15 of elements $\{2,4,7,8,11,13,14\}$ are $\{4,2,4,4,2,4,2\}$ and in this particular example any choice of a except $a = 14$ leads to the correct result. For $a = 14$ we obtain $r = 2$, $a^{r/2} \equiv -1$ mod 15 and the methods fails.

Classically finding r is as time consuming as finding factors of N by the trial divisions, however, if we employ quantum computation r can evaluated very efficiently. Shor [7] describes a quantum algorithm which provides the order r of a randomly chosen a and which runs in polynomial time *i.e.* requires poly(log N) steps. Let us now outline the main features of this algorithm.

6. Measuring Periodicity

Suppose you want to find period r of $f_N(x)$ where $x = 0,1,2,\ldots M-1$ for some large $M = 2^m$ $(M \approx N^2$ and the values $f(0), f(1), \ldots f(r)$ are all different). Here is an efficient quantum method. First we choose a computational basis (which we label $\{|x\rangle\}$ for the first register and $\{|f_N(x)\rangle\}$ for the second register) and compute function $f_N(x)$ in a quantum way:

$$\frac{1}{\sqrt{M}}\left(\sum_0^{M-1} |x\rangle\right)|0\rangle \xrightarrow{U_{f_N}} \frac{1}{\sqrt{M}}\sum_0^{M-1} |x\rangle\,|f_N(x)\rangle\,. \tag{11}$$

Function $f_N(x) = a^x$ mod N can be computed efficiently. Next perform a measurement in the computational basis to determine the bit values in the second register. Suppose the outcome of this measurement is $f_N(l)$ for a least l $(f_N(l) = f_N(jr+l)$ for $j = 0,1,2,\ldots)$. The post measurement state is

$$|\phi_l\rangle = \frac{1}{\sqrt{A+1}}\sum_{j=0}^{A} |jr+l\rangle\,|f_N(l)\rangle\,. \tag{12}$$

where A is the greatest integer less than M/r. Thus in the first register we have a uniform superposition of labeled basis states where the labels have been chosen with period r $(l, l+r, l+2r, \ldots, l+Ar)$. From this state we wish to extract the information about the periodicity r.

The extraction of r will be achieved by applying to the first register the quantum discrete Fourier transform *i.e.* the unitary transformation (DFT) which acts on a M dimensional Hilbert space and is defined relative to a chosen basis $|0\rangle, \ldots, |M-1\rangle$ by:

$$DFT : |x\rangle \longmapsto \frac{1}{\sqrt{M}}\sum_{y=0}^{M-1} \exp(2\pi ixy/M)\ |y\rangle \tag{13}$$

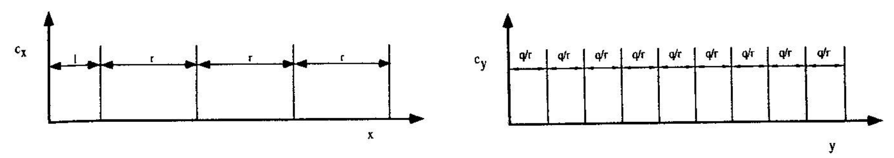


Fig. 1. c_x and the modulus of its Fourier transform c_y.

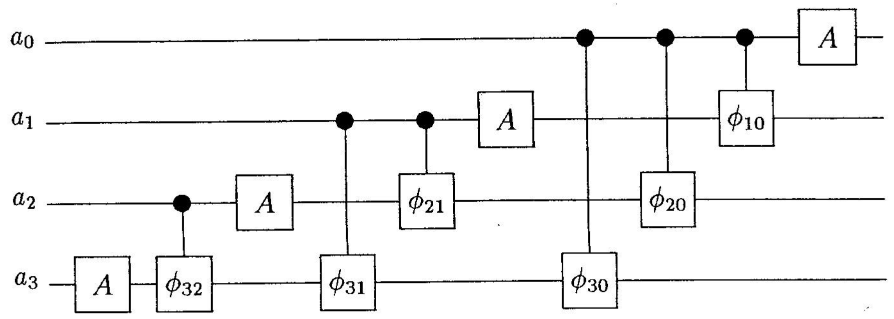


Fig. 2. Network effecting a DFT on a four–bit register, the phases that appear in the operations $B(\phi_{jk})$ are related to the "distance" of the qubits upon which **B** acts, namely $\phi_{jk} = \pi/2^{j-k}$.

The reason for calling this particular unitary transformation the discrete Fourier transform becomes obvious when you notice that in the transformation

$$DFT : \quad |\phi_{in}\rangle = \sum_x c_x\,|x\rangle \longmapsto |\phi_{out}\rangle = \sum_y c_y\,|y\rangle \tag{14}$$

the coefficients c_y are the discrete Fourier transforms of c_x's *i.e.*

$$c_y = \frac{1}{\sqrt{M}}\sum_x \exp(2\pi ixy/M)c_x. \tag{15}$$

There exists an efficient quantum algorithm for DFT which is a quantum analog of the Fast Fourier Transform algorithm (for details see [24,25]).

To see qthe principle of how this works, we consider first the simplified situation where r divides M exactly. Write $A = M/r - 1$. The final state corresponding to (12) is then

$$|\phi_{in}\rangle = \sqrt{\frac{r}{M}}\sum_{j=0}^{A} |jr+l\rangle = \sum_{x=0}^{M-1} c_x\,|x\rangle\,, \tag{16}$$

where $c_x = \sqrt{(r/M)}\delta_{x,jr+l}$ for $j = 0, 1, \ldots A$ is a periodic function of x (see Fig. 1).
Performing DFT on $|\phi_{in}\rangle$ gives

$$|\phi_{out}\rangle = \sum_y c_y\,|y\rangle\,, \tag{17}$$

where the amplitude of c_y is

$$c_y = \frac{\sqrt{r}}{M}\sum_{j=0}^{A}\exp\left(\frac{2\pi i(jr+l)y}{M}\right) = \frac{\sqrt{r}}{M}\left[\sum_{j=0}^{A}\exp\left(2\pi i\frac{jry}{M}\right)\right]\exp\left(2\pi i\frac{ly}{M}\right). \tag{18}$$

The term in the square bracket on the r.h.s. is zero unless y is a multiple of M/r,

$$c_y = \begin{cases} \exp(2\pi i l y/M)/\sqrt{r} & \text{if } y \text{ is a multiple of } M/r \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

i.e. the Fourier transform of a state with period r is a state with period M/r:

$$|\phi_{out}\rangle = \frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}\exp(2\pi i l y/M)\left|j\frac{M}{r}\right\rangle \tag{20}$$

Note that the Fourier transform "inverts" the periodicity of the input ($r \to M/r$) and has a translation invariance property which washes out the shift l (see Fig. 1).

Now we perform a bit by bit measurement on the first register to learn y which can only be a multiple $\lambda M/r$ with $\lambda = 0, \ldots, r-1$ chosen equiprobably. From the relation $y/M = \lambda/r$, knowing y and M and assuming that λ and r do not have any common divisor apart from 1 we can determine r by canceling y/M down to an irreducible fraction. Finally from r we calculate prime factors of N. The two essential computations *i.e.* the evaluation of f_N and the quantum discrete Fourier transform can be performed efficiently so that the whole algorithm takes only about $(\log N)^3$ steps !

Shor's algorithm is a randomized algorithms which runs successfully only with probability $1-\epsilon$ and we know when it is successful. It produces a candidate factor of N and must be followed by a trial division to check whether the result is a factor or not. If $\epsilon > 0$ is independent of the input N. By repeating the computation k times, we get probability $1-\epsilon^k$ of having at least one success. This can be made arbitrarily close to 1 by choosing a fixed k sufficiently large. Furthermore if a single computation is efficient then repeating it k times will also be efficient since k is independent of N . Thus the success probability of any efficient randomized algorithm of this type may be amplified arbitrarily close to 1 while retaining efficiency. Indeed we may even let the success probability $1-\epsilon$ decrease with N as $1/\text{poly}(\log N)$ and k increase as $\text{poly}(\log N)$ and still retain efficiency while amplifying the success probability as close to 1 as desired. Shor's quantum factoring algorithm is of this type; it is based on an efficient algorithm which provides a factor of the input N with probability which decreases as $1/\text{poly}(\log N)$.

The randomness in the algorithm is due to certain mathematical results concerning the distribution of prime and coprime numbers. For example, for λ being chosen at random it follows from number theory that the probability of λ and r to have no common divisor apart from 1 is greater than $1/\log r$ for largish r (see for example [23,26]). Also if we abandon the assumption that r divides M (very unlikely and adopted here only for pedagogical purposes) the Fourier transform of c_x will not produce sharp maxima as in Fig.1. which may contribute to possible errors while reading y from the register. Subsequent estimation of r is calculated using additional mathematical approximation techniques (continued fraction expansion).

If we try to factor bigger and bigger numbers N it is enough to repeat the computation $\approx poly(\log N)$ times to amplify the success probability as close to 1 as we wish. This gives an efficient determination of r and an efficient method of factoring any N !

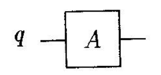
7. From Quantum Computation to Quantum Networks

An open question has been whether it would ever be practical to build physical devices to perform such computations, or whether they would forever remain theoretical curiosities. Like classical computers, quantum computers can be built out of logic gate networks. In the case of a quantum computer, a logic gate can be thought as a unitary operation that acts only on the space of a restricted number of qubits, one for a one-bit gate, two for a two-bit gate, etc. Deutsch [27] described quantum networks composed of elementary logic gates connected together by wires and showed that there exists a universal quantum gate from which any quantum computation can be built. More recently Barenco [28] and independently Sleator and Weinfurter [29] (1994) proved that a single two-bit gate suffices to implement the Deutsch gate. Finally it has been shown that almost any non-trivial two-bit gate is universal [30,31].

Quantum logic gates perform elementary unitary operations on qubits. In this section we will illustrate how complex quantum operations, such as the quantum discrete Fourier transform discussed above, can be implemented as a network consisting of only one- and two-bit gates.

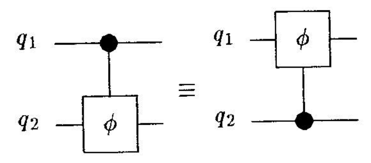
Consider the single qubit gate **A** performing the unitary transformation

$$A = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{21}$$



where the diagram on the right provides a schematic representation of the gate **A** acting on a qubit q. Consider also the two-bit gate $\mathbf{B}(\phi)$ acting on qubits q_1 and q_2 and performing the operation

$$B(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix} \tag{22}$$



in the Hilbert space $\mathcal{H} = \mathcal{H}_{q_1} \otimes \mathcal{H}_{q_2}$ of the two qubits with the basis $\{|0\rangle|0\rangle, |0\rangle|1\rangle, |1\rangle|0\rangle, |1\rangle|1\rangle\}$ (the diagram on the right shows the structure of the gate). The gate $\mathbf{B}(\phi)$ performs a conditional phase shift *i.e.* multiplication by phase factor $e^{i\phi}$ but only if the two qubits are both in their $|1\rangle$ states.

The two gates can be used to implement the efficient quantum DFT on a register of any size. For example, consider a four–bit register with qubits $a_0, \ldots a_3$. The network in Fig. 2. follows step by step the classical algorithm of a DFT (see for instance [32]), and perform the operation

$$|a\rangle \longmapsto \frac{1}{\sqrt{2^4}} \sum_{c=0}^{2^4-1} \exp(2\pi i ac/2^4)\,|b\rangle\,, \tag{23}$$

where $|b\rangle$ represents the value c read *reversing* the order of the bits *i.e.*

$$b = \sum_{i=0}^{3} 2^i c_{3-i} \quad \text{with } c_k \text{ given by} \quad c = \sum_{k=0}^{3} 2^k c_k. \tag{24}$$

A general case of L qubits requires a trivial extension of the network following the same sequence pattern of gates **A** and **B**.

Each gate operates for a fixed period of time (the clock time of the computer) and the number of gates needed to complete the full quantum DFT grows only as a quadratic function of the size of the register. (The transformation on the L–qubit register requires L operations **A** and $L(L-1)/2$ operations **B**, in total $L(L+1)/2$ elementary operations). Thus the quantum DFT can be performed efficiently. Moreover, it can be even simplified. Note that in the network shown in Fig. 2., the operations $\mathbf{B}(\phi)$ that involve distant qubits a_j and a_k, *i.e.* qubits for which $|j-k|$ is big (and therefore $\phi = \pi/2^{k-j}$ approaches zero), are close to unity. Therefore when performing the quantum DFT on registers of size L, one can neglect operations **B** on distant qubits (more precisely on qubits a_j and a_k for which $|j-k| > \log_2(L)+2$) and still retrieve the periodicity of coefficients c_x.

The network of gates for the quantum DFT enables the efficient implementation of the second part of Shor's algorithm. The first part requires an efficient quantum evaluation of the function $f_N(x) = a^x \bmod N$. The computation of $f_N(x)$ is "easy" *i.e.* the number of gates does not grow faster than a polynomial in the size of the input. The respective network is constructed by combining networks which perform addition and multiplication in a reversible and unitary way.

8. Practicalities

It remains an open question which technology will be employed to build first quantum computers. The conditional quantum dynamics which supports quantum logic gates and quantum networks can be implemented in lots of different ways ranging from the Ramsey atomic interferometry [33] to ions in ion traps [34]. However, in order to perform a successful quantum computation one has to maintain a coherent unitary evolution until the completion of the computation. In practice qubits, registers, and the whole machine interact with the environment causing decoherence. If the state of the whole machine is described by a density matrix in a computational basis

$$\rho(t) = \sum_{a,b} \rho_{ab}(t)\,|a\rangle\langle b|\,. \tag{25}$$

then a typical interaction with the environment in a thermal equilibrium destroys the off–diagonal elements ($\rho_{ab}(t) \to 0$, $a \neq b$) and changes the diagonal elements ($\rho_{aa}(t) \to \rho_{aa}^{\text{thermal}}$). When the off–diagonal elements (which are responsible for interference) vanish quantum computers lose their unique power. Simple theoretical models of decoherence [35,36] show that the probability of a successful computation in a single run decreases exponentially with the input size ($\approx \log N$) which implies that decoherence cannot be efficiently dealt with by simply increasing number of runs. What we need is some form of "quantum error correction" to stabilise the computation. A theoretical possibility for one such stabilising technique is outlined in [37].

From the experimental point of view one may try to reduce the effect of decoherence by employing technologies which allow the performance of many elementary computational steps within the decoherence time (see [38] for interesting numerical estimations regarding several selected physical realisations of qubits). Although the current technologies cannot support even a very simple quantum factorisation we hope that the world-wide experimental efforts will make practical quantum computation possible in a not too distant future. It should be stressed, however, that from the fundamental standpoint it is irrelevant when exactly the first non–trivial quantum computer is built — what matters is that quantum computation tell us about a connection between physics and computation making the two branches of science inseparable. The philosphical implication of this fusion are nothing but trivial and are discussed at length by Deutsch [39].

Acknowledgements This work was partially supported by the Royal Society. A. B. acknowledges the financial support of the Berrow's fund at Lincoln College, Oxford.

References

[1] R. Landauer: *IBM J. Res. Dev.* **5** (1961) 183;

[2] D. Deutsch: *Proc. R. Soc. London A* **400** (1985) 97;

[3] R. Feynman: *Int. J. Theor. Phys.* **21** (1982) 467;

[4] D. Deutsch, R. Jozsa: *Proc. R. Soc.Lond. A* **439** (1992) 553;

[5] E. Bernstein,U. Vazirani: in *Proc. 25th ACM Symposium on the Theory of Computation*, (1993) p. 11;

[6] D.S. Simon: in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser (IEEE Computer Society Press, Los Alamitos, CA), (1994) p. 116;

[7] P.W. Shor: in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser (IEEE Computer Society Press, Los Alamitos, CA), (1994) p. 124;

[8] R. Rivest, A. Shamir, L. Adleman: *On Digital Signatures and Public–Key Cryptosystems*, MIT Laboratory for Computer Science, Technical Report, MIT/LCS/TR-212 (January 1979).

[9] R.D. Silverman: *Math. Comp.* **48** (1987) 329;

[10] A.K Lenstra, H.W. Lenstra Jr., M.S. Manasse, J.M. Pollard: in *Proc. 22nd ACM Symposium on the Theory of Computing*, (1990) p. 564;

[11] D. Welsh: *Codes and Cryptography* (Clarendon Press, Oxford, 1988);

[12] C.H. Papadimitriou: *Computational Complexity* (Addison-Wesley, New York 1994);

[13] A.S. Holevo: *Problemy Peredachi Informatsii* **9** (1979) 3; (this journal is translated by IEEE under the title *Problems of Information Transfer*).

[14] E.B. Davies: *IEEE Trans. Inform. Theory* **IT 24** (1978) 596;

[15] C.A. Fuchs, C.M., Caves: *Phys .Rev. Lett* **73** (1994) 3047;

[16] R. Jozsa, B. Schumacher: *J. Mod. Optics* **41** (2343) 1994;

[17] S. Wiesner: *Sigact News* **15(1)** (1983) 78;

[18] C.H. Bennett, G. Brassard: in *Proceedings of the IEEE international Conference on Computers, Systems, and Signal Processing, Bangalore, India* 175, (IEEE, New York, 1984).

[19] A. Ekert: *Phys. Rev. Lett.* **71** (1993) 4287;

[20] C.H. Bennett: *Phys. Rev. Lett.* **68** (1992) 3121;

[21] C.H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, W.K. Wootters: *Phys. Rev. Lett.* **70** (1993) 1895;

[22] C.H. Bennett: *SIAM J. Comput.* **18(4)** (1989) 766;

[23] M.R. Schroeder: *Number Theory in Science and Communication* (Springer-Verlag, Berlin, 1984);

[24] D. Coppersmith: *IBM Research Report No. RC19642* (1994) ;

[25] A. Ekert, R. Jozsa: *to appear in Rev. Mod. Phys.* (1995) ;

[26] G.H. Hardy, E.M. Wright: *An Introduction to the Theory of Numbers*, 4th ed. (Oxford University Press, Oxford, 1965);

[27] D. Deutsch: *Proc. R. Soc. London A* **425** (1989) 73;

[28] A. Barenco: *to appear in Proc. R. Soc. London A* (1995) ;

[29] T. Sleator, H. Weinfurter, H.: *preprint* (1994) ;

[30] D. Deutsch, A. Barenco, A. Ekert: *to appear in Proc. R. Soc. London A* (1995) ;

[31] S. Lloyd: *Los Alamos National Laboratory preprint* (1994) ;

[32] D.E. Knuth: *The Art of Computer Programming, Volume 2: Seminumerical Algorithms* (Addison-Wesley, New York, 1981);

[33] M. Brune, P. Nussenzveig, F. Schmidt-Kaler, F. Bernardot, A. Maali, J.M. Raimond, S.Haroche: *Phys. Rev. Lett* **72** (1994) 3339;

[34] C.I. Cirac, P. Zoller P.: *University of Innsbruck preprint* (1994) ;

[35] W. Unruh: *Phys. Rev. A* **51** (992) 1994;

[36] G.M. Palma *et al.*: *University of Oxford preprint* (1995) ;

[37] A. Berthiaume, D. Deutsch, R. Jozsa: in *Proceedings of the Workshop on the Physics and Computation—PhysComp '94* (IEEE Computer Society Press, Dallas, Texas, 1994);

[38] D. DiVincenzo: *Phys. Rev. A* **50** (1995) 1015;

[39] D. Deutsch: *The Fabric of Reality* (Viking-Penguin Publishers, London, to appear early 1996);