Quantum Computation Tutorial

Brian Lederer - brianl@it.uts.edu.au

October, 2001

Abstract. Quantum computers (yet to be demonstrated) will derive their power by exploiting the basic quantum effect called *superposition* whereby a quantum system can be in a number of its basic states simultaneously; eg a photon, in an interferometer, can be travelling both horizontally and vertically simultaneously, or an electron in a magnetic field can have spin up and spin down simultaneously. When a quantum system, in a superposition state, interacts with some apparatus (and thereby undergoes a transition) the component states are all transformed simultaneously and independently of one another. This opens a door to a new kind of computational parallelism, called *quantum parallelism.* To realise this, at least conceptually, it is only necessary to take quantum systems consisting of one (or more) binary valued variables (such as photon direction, or electron spin) and, in analogy with a computer, treat each variable as a bit (called a *qubit)* and the collection as a register. A superposition then corresponds to the qubit or register having a number of values simultaneously. A transition corresponds to these values all being operated upon simultaneously (and the apparatus performing the operation involved corresponds to one or more *gates* making up a *circuit*). Since the number of such values goes up exponentially with the size of the register we get an enormous gain in computational power, compared with conventional computers. As demonstrated by Shor in 1994, a problem previously thought intractable, viz factorising large integers with hundreds of digits, becomes easy. Since the public key encryption system used by financial institutions is based on the difficulty of the factorisation problem, quantum computing has generated a lot of interest.

1. Why the fuss?

A popular lead-in is to show a graph of miniturisation (of computer components) vs time and claim that, if the trend to smallness continues, by about 2016 it will no longer be possible to hide quantum effects.

For example, a bit (or binary digit) is defined by the voltage (or charge) across a capacitor: a high value is interpreted to be 1 (or True), and a low value to be 0 (or False). To allow for variations in the charge, "high" is actually taken to be any value bigger than some intermediate value. In present day computers, "high" would involve millions of electrons and variations due to quantum uncertainties would not upset the sharp distinction required to realise a bit. Not so when things become very small.

So "if you can't beat them, why not join them?" Rather than being a counsel of despair, it turns out that if we try to exploit quantum effects, rather than fight them, we get a tremendous gain (a quantum leap!) in computing power, for some problems that are intractable with conventional (or classical) computers.

The most spectacular example is factorisation. This is where we are given a natural number, eg 15, and asked to find the factors (5 and 3) if such exist (ie if the number is *composite*, ie not *prime*). You might say, what's interesting or hard about this? What is interesting is that the public key encryption scheme, used by the banks etc, is based on the difficulty of factorising large numbers (with about 200 digits, rather than 2 digits as with 15). It would take billions of years, with conventional computers to factorise a 200 digit number and hence break an encryption scheme based on it. This is because, on average, you find a factor by brute force - searching through all numbers less than the given number (or, better, less than the square root of the given number): so you have to search through about 10 to the power 100 numbers! But amazingly Shor, in 1994, devised a method (algorithm) for doing so in a few seconds, if you had a quantum computer!

So this created quite a stir. And, in a recent SMH article on terrorism, describing the public availability of encryption software packages, it was mentioned that a quantum computer would be needed to crack the code.

But do quantum computers exist? The answer is "not yet". However, labs around the world are working furiously to build one. Our previous Head Of School (Computing Science) at UTS, on sabbatical in the USA last year, was taken into a room full of equipment in an IBM research lab and told that what was being worked on was a 5 bit quantum computer.

You might laugh at 5 bits: after all, the PC on your desk has megabytes of memory, where each byte has eight bits. But it turns out that the power of quantum computers comes, not so much from the number of bits, but from the fact that in a collection (or *register)* of quantum bits (called *qubits*) many values can be stored and operated upon simultaneously. (If you know anything about quantum mechanics - see below - you may be thinking that this is quantum *superposition* coming in here; and you'd be correct. A quantum system can be in two or more of its base states - register values here - simultaneously. A quantum card could fall left and right simultaneously! Likewise, Schrodinger's poor cat was both alive and dead simultaneously!).

For example, if we had a 4 qubit register (the case we will look at below) then it is possible to have all 16 possible values - 0000, 0001, 0010, 0011, ..., 1111 - in the register simultaneously. Then, if the problem we are solving requires the evaluation of some function at all these values (as will be the case for Shor's algorithm), we will be able to generate all the function values in a single step.

This is dramatically different from a parallel processor of the conventional kind where we have many processors working in parallel (your PC has a single processor), each on a register(s) containing a single value. With a quantum computer we have the complete opposite: a single processor working on a register containing many values simultaneously. Moreover the number of values that can be worked on simultaneously goes up *exponentially* with the size of the register: with a 4 qubit register there are 2 to the power 4 values, ie 16 values (as we've just seen); with a 5 qubit register 2 to the power 5 = 32 values, and so on. This ability to work on a large number of values in parallel is called *quantum parallelism* and marks a paradigm shift in computing.

The revolutionary nature of this can be seen in another example, which might amuse people working in the Complexity and Chaos area. I have recently received a paper showing how a quantum computer could be used to compute the orbit of a chaotic system. Classical computers fail in this task because of the fact that numerical errors get amplified exponentially in the chaotic region of parameter space: this is sensitive dependence on initial conditions. It is this effect that has led people to say that chaotic systems (like the weather) are deterministic but not predictable. But the paper claims that the exponential buildup of error can be offset by the exponential power of quantum parallelism, at least in the example cited!

2. A result in number theory

Shor's algorithm for fast factorisation uses a result known for a long while and appearing in Hardy's book on number theory written in the early part of the twentieth century.

Suppose you have a number, such as $N = 15$, for which you want the factors. Take any number *a relatively prime* with 15 such as $a = 2$ (2 and 15 have no common factor and so are relatively prime; whereas 6 and 15 are not relatively prime because they have 3 as a common factor). Now form successive powers of a modulo N (where "n modulo N" means the remainder on dividing n by N; eg 33 mod 15 = 3 because 15 goes twice into 33 with 3 left over):

$$2^0 \bmod 15 = 1 \bmod 15 = 1$$

$$2^1 \bmod 15 = 2 \bmod 15 = 2$$

$$2^2 \bmod 15 = 4 \bmod 15 = 4$$

$$2^3 \bmod 15 = 8 \bmod 15 = 8$$

$$2^4 \bmod 15 = 16 \bmod 15 = 1$$

$$2^5 \bmod 15 = 32 \bmod 15 = 2$$

$$2^6 \bmod 15 = 64 \bmod 15 = 4$$

$$2^7 \bmod 15 = 128 \bmod 15 = 8$$

$$2^8 \bmod 15 = 256 \bmod 15 = 1$$

...

The result is 1, 2, 4, 8, 1, 2, 4, 8, 1, ... What we observe is that the sequence is periodic with period $r = 4$. Then, according to the number theory result, the factors p and q of N are given by the formula

$$p, q = \text{gcd}\ (a^{r/2} \pm 1, N)$$

where *gcd* stands for "greatest common divisor" (eg gcd (14, 21) = 7).

In the present case this is $\text{gcd}(2^{4/2} \pm 1, 15) = \text{gcd}(4 \pm 1, 15)$ = gcd (3,15), gcd (5, 15) = 3 or 5, as required.

The same result will be obtained using another a value relatively prime with 15 (eg 7: the sequence goes 1, 7, 4, 13 and then repeats; the above formula gives gcd (48, 15), gcd (50, 15) = 3, 5).

In general we have to find the period r of the function:

$$f_N(x) = a^x \bmod N$$

With a conventional computer it turns out that there is just as much work finding the factors this way as in the brute force search; this is because, for a large number N, the period r will also be large and it will take quite a lot of evaluations to find it.

However, with a quantum computer, this result is interesting because the period is a global property of a function requiring lots of evaluations and this is the sort of thing a quantum computer is good at finding, as we shall see.

3. Bomb detection

So far we haven't said much about quantum mechanics, the theory that describes quantum systems and that is at the heart of all the speculation about quantum computers. Nor have we said anything about how a quantum bit might be realised. So let's start on that here. And to highlight the weirdness of what goes on at the quantum level, which you may be vaguely familiar with, we'll take a topical application (to continue the terrorist theme): how, with quantum mechanics, you can test whether a bomb is a dud or not without actually interacting with it!

To lead into this, first consider an *interferometer* - well known in the 19th century. This is based on and demonstrates the wave nature of light. By means of an arrangement of semi or fully silvered mirrors, a light beam can be split into two component beams and then recombined into a beam indistinguishable from the original one (see Figure 2).

Classical treatment

At BS1 (the semi-silvered mirror or beam splitter) the incoming horizontal beam, or wave, (H) is split into two diverging component beams: half goes into a reflected beam V travelling in the vertical direction and half is transmitted through the mirror in the horizontal direction (H):

H -> half H + half V

The two fully silvered mirrors M cause the beams to converge with (H -> V and V -> H) and at BS2 they are recombined. Here *constructive* interference in the direction H causes the original beam to be re-established and all the light goes into DH; whereas, in the V direction there is *destructive* interference and no light goes into DV. Constructive interference means that the reflected and transmitted waves in the H direction have their peaks overlapping perfectly; whereas destructive interference in the V direction means that the peaks are 180 degrees *out of phase*, ie peaks overlap troughs (because, in the V direction, BS2 inverts one of the waves, turning a peak into a trough). (For more on interference see Figure 3).

Quantum treatment

If we focus just on BS1 we come in contact with what is new. Here, according to Einstein, we have to imagine the incoming beams made up of *photons*, tiny packets of energy, all of the same amount, proportional to the frequency of the light (assuming a monochromatic beam). The photon is *indivisible* (as are some other fundamental particles, such as electrons, etc): so we can't say that a photon splits at BS1 into two half photons going in the V or H directions. Moreover, the photons go through BS1 *independently* and are not interacting with one another.

If we place the detectors just beyond BS1 and follow individual photons we find they go into DV or DH randomly in such a way that, on average, half arrive at DV and half at DH (Figure 4).

Quantum Mechanics handles this by saying that, whereas the incoming photon is in state H, the outgoing photon is in <u>both</u> states H and V, ie in what is termed a *superposition* of basic states. The effect of the detector is then to force the photon to *collapse* into one or other of its component or basic states. Whereas the effect of BS1 is completely *deterministic* (in creating the superposition), the effect of D in causing the collapse is *non-deterministic* in that the photon collapses randomly into one or other basic state; ie the measuring or observing process has an uncontrollable effect on the system. The probabilities with which the two possibilties occur are, however, determined: and these are one half in each case.

Mathematically, all this is handled neatly with the constructs *state vector* (and *state space).* The incoming state is denoted by |H> (termed a *ket* vector, using Dirac's notation); the superposition state is denoted |H> + |V> or, more correctly, $\frac{1}{\sqrt{2}}|H> + \frac{1}{\sqrt{2}}|V>$ (see Figure 5; note that the state space is 2 - dimensional). The effect of BS1 can be represented by:

$$|H> \xrightarrow{BS1} \frac{1}{\sqrt{2}}|H> + \frac{1}{\sqrt{2}}|V>$$

The $\frac{1}{\sqrt{2}}$'s (the *coefficients* of the vectors) are called *probability amplitudes*: to get the probability of collapse you have to square them. Thus, the effect of D is to project the state vector into one of its components:

either

$$\frac{1}{\sqrt{2}}|H> + \frac{1}{\sqrt{2}}|V> \xrightarrow{D} |H> \quad \text{(probability 1/2)}$$

or

$$\frac{1}{\sqrt{2}}|H> + \frac{1}{\sqrt{2}}|V> \xrightarrow{D} |V> \quad \text{(probability 1/2)}$$

Quantum interference

Let us return to the interferometer as a whole (and move the detectors back to their original positions beyond BS2). If we now look at BS2, we have the photon in the superposition state $\frac{1}{\sqrt{2}}|H> + \frac{1}{\sqrt{2}}|V>$ coming in to BS2 (the effect of the mirrors M is to convert |H> to |V> and vice versa so as not to change the overall state). BS2 acts on the individual components in the same way as BS1: the |H> component goes into the same superposition as before; the |V> component goes into a slightly different superposition:

$$|V> \xrightarrow{BS2} \frac{1}{\sqrt{2}}|H> - \frac{1}{\sqrt{2}}|V>$$

where the minus takes account of the phase change. The net effect of BS2 is:

$$\frac{1}{\sqrt{2}}|H> + \frac{1}{\sqrt{2}}|V> \xrightarrow{BS2} \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|H> + \frac{1}{\sqrt{2}}|V>\right) + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|H> - \frac{1}{\sqrt{2}}|V>\right)$$

Simplifying on the right hand side gives 1/2 |H> + 1/2|V> + 1/2 |H> - 1/2|V> = |H>. Hence, the photon goes into DH, as required.

Notice how at BS2, the apparatus acts on both component states of the superposition simultaneously. This is the beginning of the idea of quantum parallelism.

It’s also worth noting that, if we abandon the superposition, and just assume that the photon, at BS1, randomly goes in either direction (H or V), as in a coin toss, then we’ll fail to get the correct result. For, the same thing will happen at BS2: there’ll be another coin toss and the photon will go into either DH or DV when it should go only into DH.

Bomb detection

Figure 6 shows the interferometer with one of the M mirrors replaced by a bomb with M attached to a detonator. If the bomb is a dud and the detonator is jammed then M is the same as before and the photon goes into DH. But if the bomb is live and the detonator can move then the effect of the bomb is to act as a detector (observer) and force a collapse. If this is to $|H\rangle$, the "low road", the photon strikes the bomb mirror and the bomb goes off. If to $|V\rangle$, the "high road", we no longer have interference; rather, the situation at BS2 is now reminiscent of BS1 and leads to the photon going into DH or DV with probability 1/2 (ie we have a second collapse).

From the point of view of an observer seeing the overall outcome, if the photon arrives at DH s/he can’t conclude anything (it could be a dud bomb; or it could be a live bomb with collapse into $|V\rangle$). But if the photon goes into DV (and the bomb hasn’t gone off) s/he can conclude that the bomb is live (the bomb caused a collapse into the "high road" which, in turn, caused a collapse into DV. With a live bomb, this happens 25% of the time.

So, if we keep on testing the bombs, eventually we will detect a live bomb without destroying it. The weird thing is this: the photon doesn’t strike the bomb; but the very possibility of it doing so influences the physical outcome!!!

Philosophical

The mathematical formalism is quite seductive. A quantum system is always in (or represented by) some state in state space (called Hilbert space). The effect of the apparatus with which the system interacts (eg BS1) is simply to rotate the state vector to point in a new direction (but not to change the length of the vector which is always 1); this is called a unitary transformation (see below). A measurement causes the state vector to project into a subspace. As we run with this structure, we find empirically that it seems to fit all cases: the predictions based on it agree amazingly well with experiment.

However, all is not well philosophically. We have to face contradictions such as a photon going in two directions simultaneously. This has caused great angst over the years with the founders - such as Schrodinger, deBroglie, and Einstein - all throwing up their hands about quantum mechanics.

One way of coming to terms with this is to see quantum mechanics as exposing the limitations of our common sense constructs. In a similar way, relativity theory, in showing that simultaneity was relative to the observer, showed limitations in our concept of time.

As a lead-in to the next section I note that, in general, a quantum experiment involves the system being *prepared* in some initial state, undergoing an *evolution* and, finally, being *measured.* In a quantum computer, we re-interpret these stages as *input, processing,* and *output.*

4. Towards a quantum computer

One qubit gates

We can interpret the interferometer as a 1 qubit computer. We identify 0 or 1 with the direction of the photon, H or V. Bit values are now state vectors:

0 -> |0> (= |H>)

1 -> |1> (= |V>)

There is nothing in a conventional bit corresponding to a superposition:

? -> |0> + |1>

The action of the beam splitter on the qubit, eg

$$| 0 > \rightarrow \frac{1}{\sqrt{2}} | 0 > + \frac{1}{\sqrt{2}} | 1 >$$

$$| 1 > \rightarrow \frac{1}{\sqrt{2}} | 0 > - \frac{1}{\sqrt{2}} | 1 >$$

mirrors the action of a logical operator, or *gate*, in a conventional computer such as NOT:

0 -> 1

1 -> 0

By analogy, the beam splitter can be viewed as a 1 qubit quantum gate. This is called the Hadamaard gate, H.

It is traditional to show a circuit or box diagram for a gate with a wire coming in and out for the input/output and Figure 7 shows some of the gates we've met so far, including the quantum NOT.

Qubit realisations

We have interpreted the two photon directions through the interferometer as the two basis states of a qubit. Other realisations, typically where some property of a quantum system is allowed only two values (such as electron *spin* - up or down) are shown in Figure 8.

Unitarity and reversibility

If the basis vectors |0> and |1> are represented by column vectors, as in Figure 7, then the quantum gates (or operators) can be represented by matrices, as shown. You can see that all these matrices are symmetric (the entries on either side of the diagonal are equal). This exemplifies a general property that all quantum gates possess, called *unitarity,*.

Unitarity follows from the fact that the *length* of the state vector, defined to be the sum of the squares of the amplitudes, must always be 1 so as to enable the square of an amplitude to be interpreted as a probability (the probabilities for all possible outcomes of an observation must sum to 1). Thus, if *a* and *b* are the amplitudes (*a* and *b* were both $\frac{1}{\sqrt{2}}$ above) then the length of the general state vector |x>

$$| x > = a \,| 0 > + b \,| 1 >$$

is required to be 1, ie $a^2 + b^2 = 1$. (It should be mentioned that, in general, *a* and *b* are complex numbers; we're hiding that here). If U is a quantum gate acting on state |x> to produce a new state |y> then it can be shown, from the fact that |x> and |y> must have length 1, that U must be unitary.

It can also be shown that U must have an inverse U^{-1}, ie there exists a gate U^{-1} which, when acting on |y> produces |x>. This means that all quantum gates must be *reversible*, ie if we know the output of a quantum gate we can unambiguously say what the input was. That is, given |y>, we can't have two different states |x> and |z> going into |y> under U (because if you act on |y> with U^{-1} you must get the unique result |x> for the state going into |y> under U; see Figure 8). Example: in the case of H, the inverse of H is just H itself (see Figure 9).

Reversible AND

A conventional computer is made up of a small number of fundamental gates, implementing logical operators or connectives, such as NOT or AND. (It is possible to show that NOT and AND are *universal*: any circuit can be made up out of just these two components, in particular the basic arithmetical operations of addition and multiplication can be implemented in terms of them). NOT is reversible and so, as we've seen (Figure 7) it's easy to get a quantum gate corresponding to it. But AND, a 2 bit gate, is <u>not</u> reversible, as the truth table (Figure 9) shows: if we know that the output of AND is 1 we can unambiguously say that the input must have been 00; but, if the output is 0, the input might have been any one of 00, 01, or 10.

Since the quantum analogue of AND will be required to be reversible, as we've just seen, it helps if we can get a classical AND that is reversible. One way to do this, as Figure 10 shows, is to retain the input bits on output lines and to add an *ancillary* input bit (set to 0) which, on output, carries the AND value. Thus, whereas AND takes two input bits and produces a single output bit, reversible AND, ie AND_R, takes three input bits and produces three output bits. Inspection of the table for AND_R shows that, for every output there is a unique input (eg 111 output has input 110).

Quantum register

A register, in a conventional computer, is a collection of bits (say 32 in a PC) on which the processor can act in a single operation. We want to be able to mimic this with our quantum computer.

One way is simply to take a quantum system composed of identical subsystems, which may be allowed to interact. For example, we had the photon's direction through the interferometer giving us a single qubit. Suppose we have two photons simultaneously entering two interferometers (Figure 11). The state of the composite system in quantum mechanics is represented by the *tensor product*:

$$| 0 >_1 | 0 >_2$$

There are four possibilities for base states depending on whether the first or second photon is in state 0 or 1: $| 0 >_1| 0 >_2$, $| 0 >_1| 1 >_2$, $| 1 >_1| 0 >_2$, $| 1 >_1| 1 >_2$. These may be abbreviated as |00>, |01>, |10>, |11>, where the first digit stands for photon 1 and the second for photon 2. (Note that the two photon state space is now a composite of the single photon state spaces and is 4 dimensional). As with any quantum system, we can, with a composite system have a superposition state which, in general, looks like

$$a\,| 00 > + b\,| 01 > + c\,| 10 > + d\,| 11 >$$

(with the sum of the squares of the amplitudes *a, b , c, d* being 1).

Quantum parallelism

The interesting thing about such a superposition state is that all four possible values of the corresponding 2 bit classical register are simultaneously present (as noted in Section 1)! Moreover, these can be operated on simultaneously (as noted in Section 3). We see this as follows.

If we come back to AND_R, and allow 3 qubits, we have:

$$| x_1 x_2 > | 0 > \xrightarrow{AND_R} | x_1 x_2 > | AND(x_1, x_2) >$$

(eg |01>|0> -> |01>|0> and |11>|0> -> |11>|1>).

In general, if *x* stands for a register value (eg 010001) and *f* stands for a function that can act on *x* then, if we have a quantum *circuit* implementing *f* (a quantum circuit is a collection of gates) we have

$$| x > | 0 > \xrightarrow{f} | x > | f(x) >$$

This becomes the model of the way functions are computed in quantum computing (and, in Shor's algorithm below, *f* will become f_N).

But now, suppose we have the input to *f* prepared in a <u>superposition</u> of all possible register values (if the register size is *n* there will be $M = 2^n$ such values):

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} | x >$$

(eg, if the register size is $n = 4$ then M is 16 and this is 1/4 ($|0000> + |0001> + \ldots + |1111>$; this can be abbreviated to 1/4($|0> + |1> + |2> + \ldots + |15>$)). Then, if this superposition state is passed into f, then

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x>|0> \xrightarrow{f} \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x>|f(x)>$$

If we take f to be the function defined in Section 2 (factorisation) and $|x>$ and $|0>$ to be 4 bit registers, then the output is:

$$1/4(|0>|1> + |1>|2> + |2>|4> + |3>|8> + |4>|1> + |5>|2> + \ldots + |15>|8>)$$

Note that the all function values now appear and that they've all been computed simultaneously! Also that the number of such simultaneous evaluations goes up exponentially with the size of the register!

Note also that the function values (the function results or outputs) are intertwined or *entangled* with the input values.

The question now is: how do we use all this information to get what we want? - if we look at it won't it all collapse?

5. Shor's algorithm for fast factorisation

As noted, the tantalising thing about the above entangled state, is that it contains, in the second register, the sequence of function values 1, 2, 4, 8, 1, 2, 4, 8, ..., ie all the information we need to find the period r. The trouble is, that if we look at it the superposition collapses and we get only a single value!

But now note that the superposition can be factored (ignoring the 1/4):

$$\begin{array}{ll} & (|0> + |4> + |8> + |12>)\,|1> \\ + & (|1> + |5> + |9> + |13>)\,|2> \\ + & (|2> + |6> + |10> + |14>)\,|4> \\ + & (|3> + |7> + |11> + |15>)\,|8> \end{array}$$

(If we'd used $a = 7$ this would be

$$\begin{array}{ll} & (|0> + |4> + |8> + |12>)\,|1> \\ + & (|1> + |5> + |9> + |13>)\,|7> \\ + & (|2> + |6> + |10> + |14>)\,|4> \\ + & (|3> + |7> + |11> + |15>)\,|13> \end{array}$$

Suppose we first look just at register 2. The state collapses to one of the four products and we measure 1, 2, 4, or 8 in register 2 (or 1, 7, 4, or 13 in the $a = 7$ case). This tells us nothing about the period. But now look at what happens in register 1. This is either case (1): (|0> + |4> + |8> + |12>) or

case (2): (|1> + |5> + |9> + |13>) or

case (3): (|2> + |6> + |10> + |14>) or

case (4): (|3> + |7> + |11> + |15>)

(the same for a = 2 or 7). We are not far from finding the period. If we knew the collapse was to case 1, we can see that each register 1 value is a multiple of the period 4; although, when we look at register 1, we will only observe one of these values - 0, 4, 8, or 12, by repeated looking we could get several multiples and extract the 4. The trouble, however, is that we can't assume that what's in register 1 is case (1) - it could be case (2), or (3), or (4) (depending how the initial collapse went on looking at register 2). If it was case (2) we still have multiples of 4 except for the offset 1 (ie you have to subtract 1 from the number to get the multiple, eg if you observed 9 you would subtract 1 to get 8 = 2*4). In case (3) the offset is 2 etc. Then the trouble will be, in repeatedly looking at register 2 followed by register 1, that the offset will jump about uncontrollably and we cannot conclude that the values we find in register 1 are straight multiples of the period.

Quantum Fourier transform

The offset is clearly the fly in the ointment and if we could remove it all would be well. Shor's brilliant solution to this problem was to apply a discrete Fourier transform to register 1 before looking (those with a mathematical background may not be surprised that Fourier transforms are coming in here, seeing that we have a periodic function). For a 4 bit register this looks like:

$$|\,j > \stackrel{DFT}{\rightarrow} \frac{1}{\sqrt{2^4}} \sum_{k=0}^{15} e^{\frac{2\pi ijk}{16}} \,|\,k >$$

If we regard DFT as a gate (it can be implemented!) and pass in the (superposition) state in register 1, eg |2> + |6> + |10> + |14> in case (3), we find that, magically,

$$|2> + |6> + |10> + |14> \stackrel{DFT}{\rightarrow} |0> - |4> + |8> - |12>$$

(using the above formula with j = 2, 6, 10, 14; see Appendix). The same applies in the other cases: the output state has the offset removed from it.

6. Discussion

There are a number of things we haven't touched upon.

One, is how do you implement the function f_N or the Fourier transform, DFT? It turns out that, similar to the case for conventional computers, there is a set of fundamental or universal quantum gates in terms of which all computations can be expressed. These gates involve at most three qubits with the qubits interacting. An example of a two qubit gate, called controlled NOT, is shown in Figure 12: whenever the *control* bit is on (1), the *target* bit is flipped (as in NOT); the control qubit is unaffected. The figure also shows a realisation of this gate using photon direction for the control qubit and photon polarisation for the target qubit.

Another issue is the fragility of quantum computations: how is it possible to control the behaviour of elementary particles (photons, electrons, atoms, etc) without disturbing them? Amazingly, as in conventional computers, it turns out that one can use additional qubits to correct for noise (in the spirit of parity bits?).

References

Milburn, G. (1998), "The Feynaman Processor", Allen & Unwin.

Nielsen, M. (2000), "Quantum Computation and Quantum Information", Cambridge University Press.

Pitinger, A. (2000), "An Introduction to Quantum Computing Algorithms", Birkhauser.

Rieffel, E. (2000), "An Introduction to Quantum Computing for non-Physicists", arXiv:quant-ph/9809016v2