Quantum parallelism: Period of a sequence:

We now have sufficient ingredients to understand how a quantum computer can perform logical operations and compute just like an ordinary computer. In this section we describe an algorithm which makes use of the quantum parallelism that we have hinted at already: finding the period of a long sequence.

Consider the sequence

$$f(0),\ f(1),\ \ldots,\ f(q-1)\ ,$$

where $q \equiv 2^k$; we shall use quantum parallelism to find its period. We start with a set of initially spin-down particles which we group into two sets (two quantum registers, or quantum variables):

$$|0;0\rangle = |\downarrow,\downarrow,\ \ldots\ ;\downarrow,\downarrow,\ \ldots\rangle\ ,$$

the first set having **k** bits; the next having sufficient for our needs. (In fact other registers are required, but by applying Bennett's solution to space management they may be suppressed in our discussion here.) On each bit of the first register we perform the $U_{-\pi/2}$ one-bit operation, yielding a superposition of every possible bit-string of length **k** in this register:

$$\longrightarrow \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a;\ 0\rangle\ .$$

The next stage is to break down the computation, corresponding to the function $f(a)$, into a set of one-bit and two-bit unitary operations. The sequence of operations is designed to map the state $|a;\ 0\rangle$ to the state $|a;\ f(a)\rangle$ for any input **a**. Now we see that the number of bits required for this second register must be at least sufficient to store the longest

result $f(a)$ for any of these computations. When, however, this sequence of operations is applied to our exponentially large superposition, instead of the single input, we obtain

$$\longrightarrow \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a;\, f(a)\rangle \ .$$

An exponentially large amount of computation has been performed essentially for free.

The final computational step, like the first, is again a purely quantum mechanical one. Consider a discrete ``quantum'' Fourier transform on the first register

$$|a\rangle \longrightarrow \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} e^{2\pi i a c/q} |c\rangle \ .$$

It is easy to see that this is reversible via the inverse transform and indeed it is readily verified to be unitary. Further, an efficient way to compute this transform with one-bit and two-bit gates has been described by Coppersmith (Fig. 10) [23,24,6].

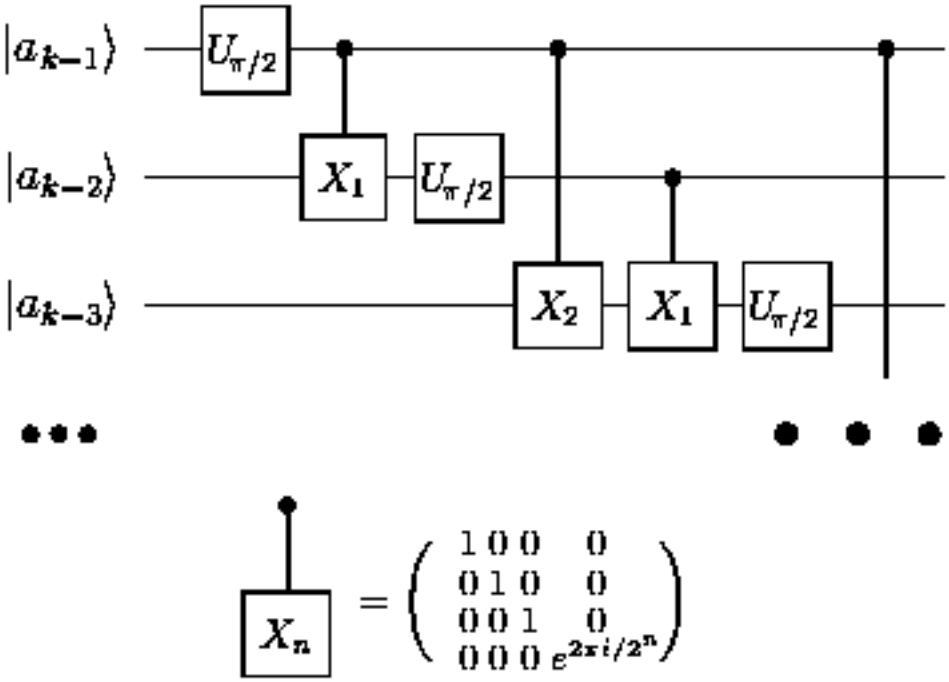

Fig. 10 Circuit for the quantum Fourier transform of the variable $|a_{k-1}\ldots a_1 a_0\rangle$ using Coppersmith's fast Fourier transformation approach [23,24,6]. The two-bit ``X_n'' gate may itself be decomposed into various one-bit and XOR gates [14].

When this quantum Fourier transform is applied to our superposition, we obtain

$$\longrightarrow \frac{1}{q}\sum_{a=0}^{q-1}\sum_{c=0}^{q-1} e^{2\pi i a c/q}|c;\ f(a)\rangle\ .$$

The computation is now complete and we retrieve the output from the quantum computer by measuring the state of all spins in the first register (the first **k** bits). Indeed, once the Fourier transform has been performed the second register may even be discarded [27].

What will the output look like? Suppose $f(a)$ has period **r** so $f(a+r) = f(a)$. The sum over **a** will yield constructive interference from the coefficients $e^{2\pi i a\, c/q}$ only when c/q is a multiple of the reciprocal period $1/r$ [25]. All other values of c/q will produce destructive interference to a greater or lesser extent. Thus, the probability distribution for finding the first register with various values is shown schematically by Fig. 11.

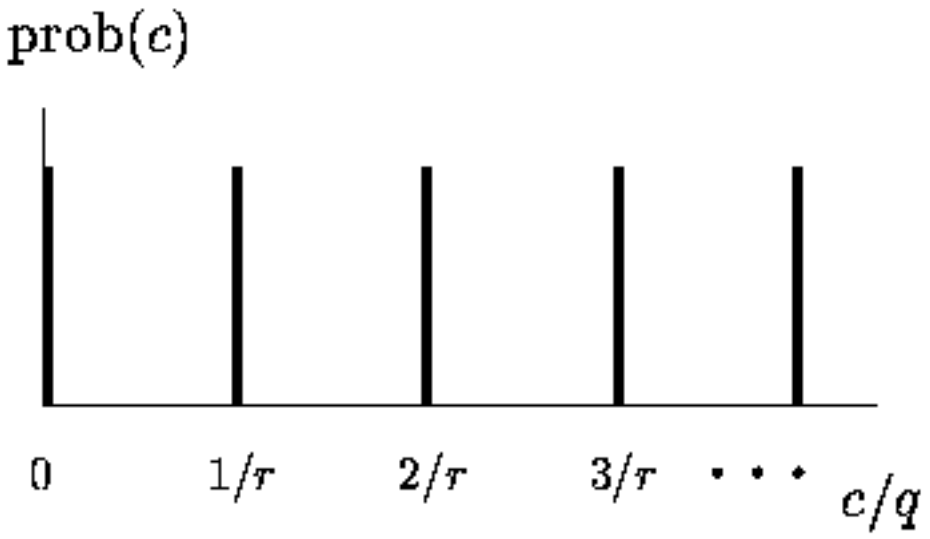

Fig. 11 Plot of the probability of each result $\mathrm{prob}(c)$ versus c/q. Constructive interference produces narrow peaks at multiples of the inverse period of the sequence $1/r$.

One complete run of the quantum computer yields a random value c/q of underneath one of the peaks in the probability of each result $\mathrm{prob}(c)$. That is, we obtain a random multiple of the inverse period. To extract the period itself we need only repeat this quantum computation roughly $\log \log r/k$ times in order to have a high probability for at least one

of the multiples to be relatively prime to the period **r**---uniquely determining it [1]. Thus, this algorithm yields only a probabilistic result. Fortunately, we can make this probability as high as we like.

All the above work may appear a little anti-climactic. We have gone to a lot of trouble to design a quantum computer to find the period of a sequence. The point is, however, that the sequence is calculated in parallel and is exponentially long---even for a small value of say **k=140** bits in the first register the quantum computer has calculated and stored more results than there are particles in the universe. We now describe the simple structure that exists in the mathematical problem of factoring which allows us to apply the above quantum computer algorithm.