doi:10.1038/nature12290

Oversimplifying quantum factoring

John A. Smolin[1], Graeme Smith[1] & Alexander Vargo[1]

Shor's quantum factoring algorithm exponentially outperforms known classical methods. Previous experimental implementations have used simplifications dependent on knowing the factors in advance. However, as we show here, all composite numbers admit simplification of the algorithm to a circuit equivalent to flipping coins. The difficulty of a particular experiment therefore depends on the level of simplification chosen, not the size of the number factored. Valid implementations should not make use of the answer sought.

Building a quantum computer capable of factoring larger numbers than any classical computer can hope to is one of the grand challenges of computing in the twenty-first century. Someday, a quantum computer running Shor's factoring algorithm[1] may factor a number hitherto unthinkably large. Such a device would most probably have to be a fully scalable fault-tolerant[2,3] quantum machine, capable of carrying out any task a quantum computer could be asked to do. Indeed, a large factorization would be convincing proof that a practical quantum computer has been built. Unfortunately, the delicate nature of quantum states—their extreme sensitivity to decoherence due to unwanted interactions with their environment[4]—means that it may be many years before a practical quantum computer is achieved. Until such a time, more modest goals must suffice. There have already been several small-scale demonstrations of Shor's algorithm[5–10], but these experiments have factored numbers no larger than 21.

Given a composite number $N = pq$, Shor's algorithm for factoring on a quantum computer efficiently computes the factors p and q from N. In this setting, 'efficiently' means that the size of the computer and length of the computation required scales polynomially in $\log N$, the number of digits of N. The core of Shor's algorithm is a random choice of a base a with $1 < a < N$, followed by the computation of the period r of an associated function $f_a(x) = a^x \bmod N$. The ability to compute this period allows the factors to be found, and this can be done efficiently on a quantum computer (Box 1). The best known classical algorithm (the number field sieve[11]) scales exponentially worse than Shor's algorithm.

Significant optimization of the basic algorithm has been achieved. As described in Box 1, roughly $3\log N$ qubits are needed. In fact, this can be reduced to exactly $2 + (3/2)\log N$ qubits[12]. A significant part of the reduction is to replace the first 'x' register with a single qubit. This has been shown to be possible[13,14], and uses the fact that the bits of the quantum Fourier transform can be read out one at a time[15]. The use of this semi-classical Fourier transform has become known as qubit recycling. A circuit using qubit recycling is shown in Fig. 1.

Compiling Shor's algorithm

All experimental realizations of Shor's algorithm until now have relied on a further optimization, that of 'compiling' the algorithm. This means using the observation that different bases a in the modular exponentiation lead to different periods of the function $a^x \bmod N$. Some of the periods are both short and lead to a factorization of the composite pq.

In 2001, the composite 15 was factored[5] using two different bases, an 'easy' base ($a = 11$, resulting in a period of 2), and a 'difficult' base ($a = 7$, with a period $r = 4$). Neither is fully general, and this allowed the factorization to take place on a seven-bit quantum computer, when the best known uncompiled algorithm would require 8 bits $(2 + (3/2)\log N$ bits, as per ref. 12). Other factorizations of 15 have since been performed using other architectures[6–8,10]. More recently, 21 has been factored[9] using just one qubit and one qutrit (a three-level system). In this case $a = 4$ is used, resulting in a period $r = 3$. (We note that Shor's algorithm normally fails when r is odd because $a^{r/2}$ is not an integer in general. Here, because $a = 4$ is a perfect square, this problem does not arise.) These results are summarized in Table 1.

BOX 1
Shor's algorithm

Given an integer $N = pq$ with p, q distinct primes, one proceeds as follows:

(1) Choose (at random) an integer $0 < a < N$.

(2) Compute the greatest common divisor (GCD) of a and N. This can be found efficiently using the Euclidean algorithm[18]. If it is not 1, then GCD(a, N) is a non-trivial factor of N. Otherwise go on to the next step.

(3) Choose $S \equiv 2^s$ such that $N^2 \leq S < 2N^2$. Construct the quantum state

$$S^{-1/2} \sum_{x=0}^{S-1} |x\rangle|0\rangle$$

using two quantum registers, the first has s qubits and the second has $\log N$ qubits. Note that in the literature x and a sometimes have their meanings interchanged.

(4) Perform a quantum computation on this state which maps $|x\rangle|0\rangle$ to $|x\rangle|a^x \bmod N\rangle$. This is the slowest step, but can be done in time $O((\log N)^3)$.

(5) Do the quantum Fourier transform on the first register, resulting in the state:

$$S^{-1} \sum_x \sum_y e^{(2\pi i/S)xy} |y\rangle|a^x \bmod N\rangle$$

This step requires $O((\log N)^2)$ time, which is much less than the modular exponentiation of the previous step.

(6) Measure the first register to obtain classical result y. With reasonable probability, the continued fraction approximation of S/y or some S/y' for some y' near y will be an integer multiple of the period r of the function $f_a(x) = a^x \bmod N$. The GCD algorithm can then efficiently find r.

(7) If r is odd, or if $a^{r/2} = -1 \bmod N$, go back to step (1). Otherwise, GCD($a^{r/2} \pm 1$, N) is p or q.

The total resources required scale as $3\log N$ qubits with computation time $O((\log N)^3)$.

[1]IBM T. J. Watson Research Center, Yorktown Heights, New York 10598, USA.

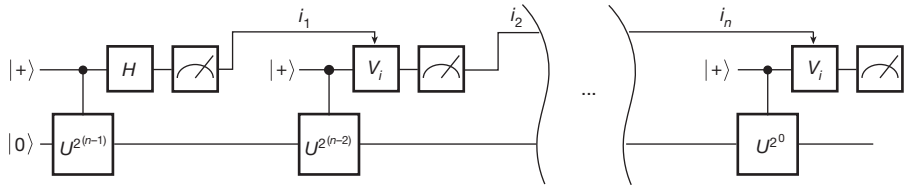

Figure 1 | Circuit for Shor's algorithm using the semi-classical quantum Fourier transform. At each stage a $|+\rangle$ state is prepared. It is used as the control input on a controlled unitary $U^{2^{n-1}}$ for the nth bit of the readout, with $U|y\rangle = |ay \bmod N\rangle$. Next, the gate $V_i = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} H$ is applied and then the qubit is measured. $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is the Hadamard gate, and the phase ϕ is computed as a function of all previous measurement results i (ref. 15). The first time there is no phase so the Hadamard is used. The process is repeated n times to read out n bits of precision of the Fourier transform.

Table 1 | Qubits required for Shor's algorithm and experimental results

N	Qubits needed[12]	Qubits implemented	Qubits compiled
15	8	7 (ref. 5)	2
		4 (refs 6, 7)	
		5 (ref. 8)	
		3 (ref. 10)	
21	10	1 + log3 (ref. 9)	2
RSA-768	1,154	2* (this work)	2
N-20000	30,002	2* (this work)	2

RSA-768 is available in Box 2 and N-20000 is available in Supplementary Information.
* A fully compiled version with one random classical bit has been performed, which can be interpreted as a maximally entangled qubit pair with one qubit held by the environment. See section 'Experiment' in main text.

It was recently shown[16] how to find bases a with small periods r for products of Fermat primes (that is, primes of the form $2^{2^k}+1$; http://en.wikipedia.org/wiki/Fermat_number). Here we go substantially beyond this idea, and show that any composite number pq has compiled versions of Shor's algorithm that can be run on a very small quantum computer. In particular, we show that there always exists a base a such that $r = 2$. Then the second register need only hold two distinct states, and the computation can be performed using only two qubits. In this case, the unitary U needed in the circuit from Fig. 1 reduces to a controlled-NOT gate. Furthermore, only one stage of the circuit is required, because all powers of U^{2^n} are the identity except for $n = 0$. The compiled circuit is shown in Fig. 2.

In order for the second register to need to hold only two distinct states, we must find a base a such that $a^2 = 1 \bmod pq$. The Chinese remainder theorem[17] tells us that

$$a^2 = 1 \bmod pq \quad \text{if and only if} \quad a^2 = 1 \bmod p \ \text{ and } a^2 = 1 \bmod q \qquad (1)$$

for p, q relatively prime. By construction

$$a \equiv \pm p p_q \ \pm q q_p \text{ has } a^2 = 1 \bmod p \ \text{ and } a^2 = 1 \bmod q \qquad (2)$$

where p_q is the multiplicative inverse of $p \pmod q$ and q_p is the inverse of $q \pmod p$. Then equation (1) tells us $a^2 = 1 \bmod pq$. These inverses can be found efficiently using the extended Euclidean algorithm[18]. There are four solutions of equation (2) corresponding to the signs. Two of these will be trivial, ± 1, and the other two will be bases resulting in compiled Shor factorizations where the function $a^x \bmod N$ has period two.

Experiment

Although the circuit shown in Fig. 2 is far simpler than the general Shor's algorithm, it is by no means trivial to implement this two-qubit circuit. Indeed, an intermediate step in the circuit creates a maximally entangled state, a key requirement for quantum computation. We therefore now employ a further optimization not used in previous experiments. Observe that in the circuit in Fig. 2, the second qubit is never measured. In fact, half of the maximally entangled state created by the controlled-NOT is simply discarded. The resulting state of the first qubit is therefore maximally mixed (that is, totally random). Because of the unitary equivalence of purifications, if we create a maximally mixed state in any way at all, it is entangled with some system in the environment. A maximally mixed state is unaffected by the Hadamard gate, so this too is unnecessary. We can therefore produce the appropriate probability distribution at the output by tossing an unbiased coin. Figure 3 shows the data for factoring 15, RSA-768 and N-20000 using this method. RSA-768 is the largest number yet factored by a general-purpose classical algorithm, and is shown in Box 2, whereas N-20000 is a 20,000-bit number of our own creation and is given in Supplementary Information.

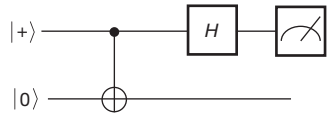

Figure 2 | The circuit for the fully compiled Shor's algorithm. The modular exponentiation is the single controlled-NOT, and the quantum Fourier transform is a Hadamard gate.

Conclusions

Of course this should not be considered a serious demonstration of Shor's algorithm. It does, however, illustrate the danger in 'compiled' demonstrations of Shor's algorithm. To varying degrees, all previous factorization experiments have benefited from this artifice. Although there is no objection to having a classical compiler help design a quantum circuit (indeed, any future quantum computer would probably function in this way), it is not legitimate for a compiler to know the answer to the problem being solved. As the cases of RSA-768 and N-20000 suggest, very large numbers can be trivially factored if the compilation depends on the answer to be found. To call such a procedure compilation is a misuse of language.

The prescription in Box 3 gives a more stringent test of small experimental implementations of Shor's algorithm. It will be a long time before even those experiments passing our test can be said to solve an interesting mathematical question. Current experiments ought to be viewed

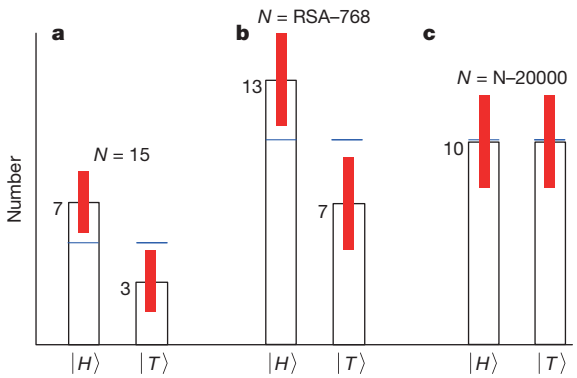

Figure 3 | Experimental data from unbiased coins. **a**, A 1998 US quarter (25 cents) was tossed 10 times to factor 15. **b**, A 1968 US penny (1 cent) was tossed 20 times in order to factor RSA-768. **c**, A 2008 US Oklahoma commemorative quarter was tossed 20 times to factor N-20000. Here $|H\rangle$ and $|T\rangle$ indicate heads and tails, respectively. The numerals at the top of the vertical open bars indicate number of occurrences, red error bars show 1σ, and light blue horizontal bars indicate the average for an unbiased coin.

BOX 2

RSA–768

RSA-768 =123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341 3

= 33478071698956898786044169848212690817704794983713768568912433889828837938780022876147116525317430877378144679994893 × 36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917

The base *a* used was

a = 10290317933024932580034888183769058752645751201785679957159211173833740637809554762657146559655560974877155097084531342124720712415517107376676461250176719955373197497390350453435865275994668289350825576184000476274812558092995299393

BOX 3

Prescription

Ideally, one would fully implement Shor's algorithm, but this has proven to be technologically challenging, because the simplest non-trivial implementation would require exquisite control of at least eight qubits. Previous experiments have instead demonstrated compiled versions of the algorithm, but for these the level of difficulty depends not on the size of the problem solved but on the level of simplification chosen. A more objective intermediate test of the period-finding kernel of Shor's algorithm would be to demonstrate the ability to find all periods from 1 … *N* on the same apparatus. For instance, a good choice would be to do period-finding on cycles of length *m* for all $1 < m \leq N$:

$$g_m(x) = \begin{cases} x+1 \bmod m & : 0 \leq x < m \\ x & : m \leq x < N \end{cases}$$

As quantum computers grow, this test will become impractical because the number of qubits needed for factorization grows with log*N* but our suggested period-finding test requires *N* experiments. Owing to the efficiency of the number-field sieve, there is a wide region where this test will be infeasible but where the factors can be found classically. For example, RSA-768 has been factored classically, but performing 2^{768} period-findings is impracticable. In such cases, the factors could be used to select a base with an easy period. Here, the length of the period found provides a good proxy for experimental difficulty, but one ought to perform Shor's algorithm blindly, using random bases. An open question is whether there is a better measure of legitimacy in this regime.

instead as technology demonstrations, showing that we can manipulate small numbers of qubits. In ref. 9, for instance, it was shown that intentionally added decoherence reduced the contrast in the data, a hallmark of a quantum-coherent process. All the experiments in refs 5–10 are important tiny steps in the direction of building a quantum computer, but actually running algorithms on only a handful of qubits is a somewhat frivolous endeavour.

Received 2 April; accepted 9 May 2013.

1. Shor, P. W. in *Proc. 35th IEEE Symp. on the Foundations of Computer Science (FOCS)* 124–134 (IEEE Computer Society, 1994).
2. Shor, P. W. in *Proc. 37th IEEE Symp. on the Foundations of Computing (FOCS)* 56–65 (IEEE Computer Society, 1996).
3. Aliferis, P., Gottesman, D. & Preskill, J. Accuracy threshold for postselected quantum computation. *Quant. Inform. Comput.* **8,** 181–244 (2008).
4. Zurek, W. H. Pointer basis of quantum apparatus: into what mixture does the wave packet collapse? *Phys. Rev. D* **24,** 1516–1525 (1981).
5. Vandersypen, L. M. K. *et al.* Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature* **414,** 883–887 (2001).
6. Lanyon, B. P. *et al.* Experimental demonstration of a compiled version of Shor's algorithm with quantum entanglement. *Phys. Rev. Lett.* **99,** 250505 (2007).
7. Lu, C.-Y., Browne, D. E., Yang, T. & Pan, J.-W. Demonstration of a compiled version of Shor's quantum factoring algorithm using photonic qubits. *Phys. Rev. Lett.* **99,** 250504 (2007).
8. Politi, A., Matthews, J. C. F. & O'Brien, J. L. Shor's quantum factoring algorithm on a photonic chip. *Science* **325,** 1221 (2009).
9. Martín-López, E., Laing, A., Lawson, T., Zhou, X.-Q. & O'Brien, J. L. Experimental realization of Shor's quantum factoring algorithm using qubit recycling. *Nature Photon.* **6,** 773–776 (2012).
10. Lucero, E. Computing prime factors with a Josephson phase qubit quantum processor. *Nature Phys.* **8,** 719–723 (2012).
11. Lenstra, A. K., Lenstra, H. W. Jr, Manasse, M. S. & Pollard, J. M. in *Proc. 22nd Annual ACM Symp. on Theory of Computing (STOC)* 564–572 (ACM Press, New York, 1990).
12. Zalka, C. Shor's algorithm with fewer (pure) qubits. Preprint at http://arXiv.org/abs/quant-ph/0601097 (2006).
13. Mosca, M. & Ekert, A. in *Quantum Computing and Quantum Communications* (ed. Williams, C. P.) 174–188 (Vol. 1509, Lecture Notes in Computer Science, Springer, 1999).
14. Parker, S. & Plenio, M. B. Efficient factorization with a single pure qubit and log*N* mixed qubits. *Phys. Rev. Lett.* **85,** 3049–3052 (2000).
15. Griffiths, R. B. & Niu, C. S. Semiclassical Fourier transform for quantum computation. *Phys. Rev. Lett.* **76,** 3228–3231 (1996).
16. Zhou, Z. & Geller, M. R. Factoring 51 and 85 with 8 qubits. Preprint at http://arXiv.org/abs/1304.0128 (2013).
17. Zi, S. The mathematical classic of Sun Zi. In Yong, L.-L. & Se, A.-T. *Fleeting Footsteps: Tracing the Conception of Arithmetic and Algebra in Ancient China* (World Scientific, 2004).
18. Heath, T (ed) *The Thirteen Books of Euclid's Elements* (Dover, 1956).

Supplementary Information is available in the online version of the paper.

Acknowledgements We acknowledge support from IARPA (contract no. W911NF-10-1-0324) and from the DARPA QUEST programme (contract no. HR0011-09-C-0047). All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of the US Government.

Author Contributions J.A.S., G.S. and A.V. designed and carried out the research. G.S. performed the experiments, J.A.S. analysed the data, and A.V. carried out the number theory. J.A.S., G.S. and A.V. wrote the paper.

Author Information Reprints and permissions information is available at www.nature.com/reprints. The authors declare no competing financial interests. Readers are welcome to comment on the online version of the paper. Correspondence and requests for materials should be addressed to J.A.S. (smolin@alum.mit.edu).