TRADING CLASSICAL FOR QUANTUM COMPUTATION USING INDIRECTION

RODNEY VAN METER

Graduate School of Science and Technology, Keio University and CREST-JST
3-14-1 Hiyoushi, Kohoku-ku, Yokohama-shi, Kanagawa 223-8522, Japan
E-mail: rdv@tera.ics.keio.ac.jp

Modular exponentiation is the most expensive portion of Shor's algorithm. We show that it is possible to reduce the number of quantum modular multiplications necessary by a factor of w, at a cost of adding temporary storage space and associated machinery for a table of 2^w entries, and performing 2^w times as many classical modular multiplications. The storage space may be a quantum-addressable classical memory, or pure quantum memory. With classical computation as much as 10^{13} times as fast as quantum computation, values of w from 2 to 30 seem attractive; physically feasible values depend on the implementation of the memory.

1. Introduction

> Any software problem can be solved by adding another layer of indirection.
> *Steven M. Bellovin*

To factor the number N using Shor's algorithm [5], a quantum computing device must evolve to hold the state $\frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}|a^j \bmod N\rangle$ for a randomly chosen, fixed a, and $q = 2^n$, where n is the bit length of N. Let $d_i = a^{2^i}$, and $x_{n-1}x_{n-2}..x_0$ be the binary expansion of x, where $|x\rangle$ holds the superposition of all values $0..q-1$. The value $a^x \bmod N$ can be rewritten [3,6] as $\prod_{j=0}^{n-1} d_j^{x_j} \bmod N$. Dividing $|x\rangle$ up into words of length w, let $|t_k(x)\rangle$, the kth word in $|x\rangle$, be $|t_k(x)\rangle = |x_{w(k+1)-1}x_{w(k+1)-2}..x_{wk+1}x_{wk}\rangle$ for $0 \le k < l, l = \lceil n/w \rceil$. $|t_k(x)\rangle$, as part of $|x\rangle$, will hold a superposition of all values 0 to 2^w.

We can reduce the n quantum multiplications to l by using the superposition $|t_k(x)\rangle$ as a quantum index into a memory array holding the 2^w n-bit entries with values $b_{m,k} = a^{m2^{wk}} \bmod N$, where m is the index into the array and k is the iteration number. The superposition of values retrieved from the memory is multiplied with the current value, giving $\frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|a^x \bmod N\rangle = \frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|\prod_{j=0}^{l-1} b_{t_j(x),j} \bmod N\rangle$. A total of $lw2^w = n2^w$ classical and l quantum modular multiplications must be performed, compared with n classical and n quantum modular multiplications using Vedral's formulation [6].

2. Indirection

In a computer, arguments to an instruction (or function) can be passed *by value* or *by reference*. By value arguments appear directly in the bits of the instruction. When accessing arguments by reference, the address of the argument is held in the instruction; the actual value must be retrieved from memory before the function can be evaluated. The address is called a *pointer* or an *index*. *Indirection* is a generalization of by reference, in which the value retrieved from memory may itself be a pointer which must in turn be dereferenced.

In the straightforward quantum implementation of modular exponentiation, the d_i values are classical values held in a single quantum register, manipulated during the forward and reverse parts of the modular addition. In our proposal, the $b_{m,k}$ values are held in a table, and a portion of the $|x\rangle$ superposition is used as an index into that table. That is, we are accessing the arguments through a single level of indirection.

3. The *b* Array

The b array is our bridge from classical computation to quantum. Each entry is n bits. We must compute 2^w values for the table, requiring w classical modular multiplications each, before each of the l quantum multiplications. The $b_{m,k}$ values must be stored into quantum-accessible memory; this may be done all at once before the quantum computation is begun, if the b array can hold $l2^w$ entries, or right before each separate multiplication, if the b can only hold 2^w entries.

The array can be held using a quantum addressable classical memory (QACM) [4]. In such a device, memory cells (the modular exponentiation values) are classical, but a quantum superposition is used as an address, and the read out value is a superposition of each classical value in proportion to the "amount" of its address present in the address superposition. One such possible device is an optical plate, with photons steered through the various cells according to the value of specific address bits. Figure 1 shows a 3-bit example. At the top, the input (generally $|0\rangle$) is steered left or right according to the high-order bit of the address superposition (carried on a control line not shown in the figure). Subsequent circles steer left or right according to their address bits, to reach the appropriate classical data memory cells. The values retrieved from the memory are combined to give the full output superposition, in weights according to the address superposition.

An equivalent array of qubits can be used in place of the QACM. However, in that case, the cost of filling the table must be accounted for, and our limitation will be the number of available qubits. Figure 2 shows a 3-bit select circuit which will choose from among the 8 possible arguments for the modular multiplier. The desired value $c_k = b_{t_k,k}$ occupies the location as shown on the right of the figure; it is then used as the argument to the modular multiplier. This select circuit can be reversed following the multiplication to restore the original locations of the $b_{j,k}$ values.

4. The Algorithm

In essence, the algorithm involves moving from a bit-oriented breakdown of the multiplications to a word-oriented breakdown.

The algorithm consists of two main parts: classically calculating the b array values, and calculating their products in the quantum domain. We pay the classical cost in step 1b in the algorithm below, and the quantum cost in step 3c.

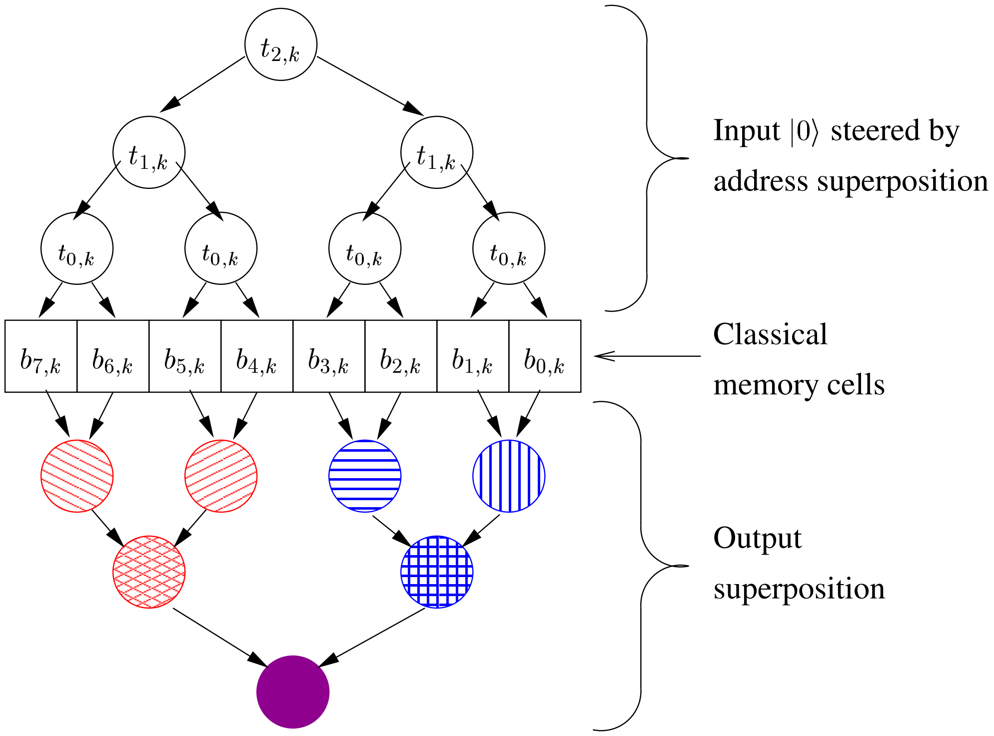

Figure 1. Quantum-Addressable Classical Memory (QACM)

The cost of setting up to use the kth iteration of the b array is technology dependent; only one of steps 1c and 3a is necessary. $O(n2^w)$ gates may be required to set a quantum memory, or only the change of a single pointer or position if a QACM is large enough to hold the entire b array at once.

(1) Calculate the b array elements:
   (a) Classically calculate $b_{j,0} = a^j$ for all $j, 0 \leq j < 2^w$
   (b) For k from 1 to $l-1$, classically square (modulo N) all 2^w elements $b_{j,k-1}$ w times to create $b_{j,k}$
   (c) (Store all $b_{j,k}$ into QACM)
(2) Initialize $|prod\rangle$ to 1
(3) For k from 0 to $l-1$, do
   (a) (Set up to use $b_{j,k}$ values: store into QACM or quantum memory)
   (b) In quantum domain, use $|t_k(x)\rangle$ as index into b, $|c_k\rangle = |b_{t_k(x),k}\rangle$
   (c) $|prod\rangle = |c_k * prod \bmod N\rangle$

Note that the algorithm uses, but does not specify, modular multiplication. We can

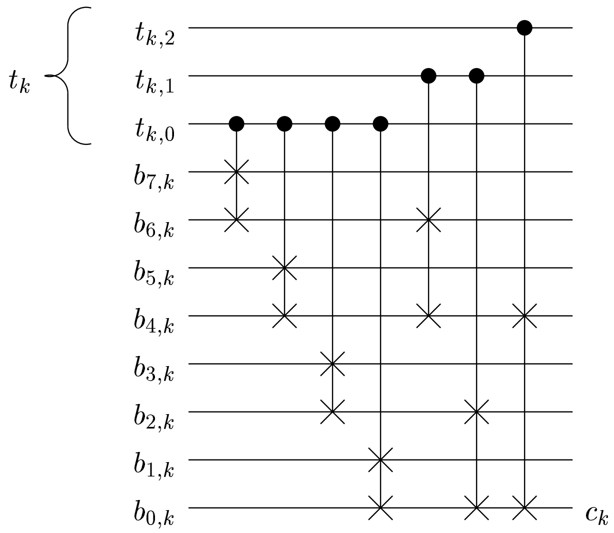

Figure 2. 3-bit Quantum Select Circuit (Q-SEL)

choose to use a variant of Vedral's carry-ripple construction, or Gossett's carry-save [2], which will increase our storage requirements but allow more concurrent execution, reducing wall-clock time (and hence coherence time demands) for the computation.

Figure 3 shows a portion of a modified form of Vedral's circuit using indirection. The dashed box indicates where update of the b array takes place, if necessary; only one-qubit gates are required. Note also that Q-SEL and its reverse are used, but, unlike Vedral's circuit, we do not need the reverse of multiplication to free up our argument. The degenerate case of $w = 1$ is therefore faster than Vedral's circuit.

5. Evaluating Cost and Selecting Word Length

Barenco's bound on the probability of success using the approximate quantum Fourier transform is 10^{-5} or lower for sizes of a few kilobits [1]. While information can be inferred even from failures that may reduce the number of iterations required, a large number will be necessary in the real world. Thus, the cost of the one-time classical computations (which may be stored and reused) is further amortized.

The goal of this work is to minimize the cost of executing Shor's algorithm, for some metric of cost important to the user. In figure 4 we show the total cost of calculating the modular exponentiation, as a function of word length w. "Cost" in this graph is an arbitrary metric; it may be wall clock time, total time on parallel machines, price tag, or some other economic cost of quantum and classical machines. Perhaps the easiest cost to consider is simply time to perform a modular multiplication. The five curves represent total cost at different ratios of quantum:classical cost, ranging from 1:1 to 10^{12}:1. The 'x' marks on each curve are the nearest integer value of w to the minimum. This recommended word length increases by approximately eight bits for each factor of one thousand the relative

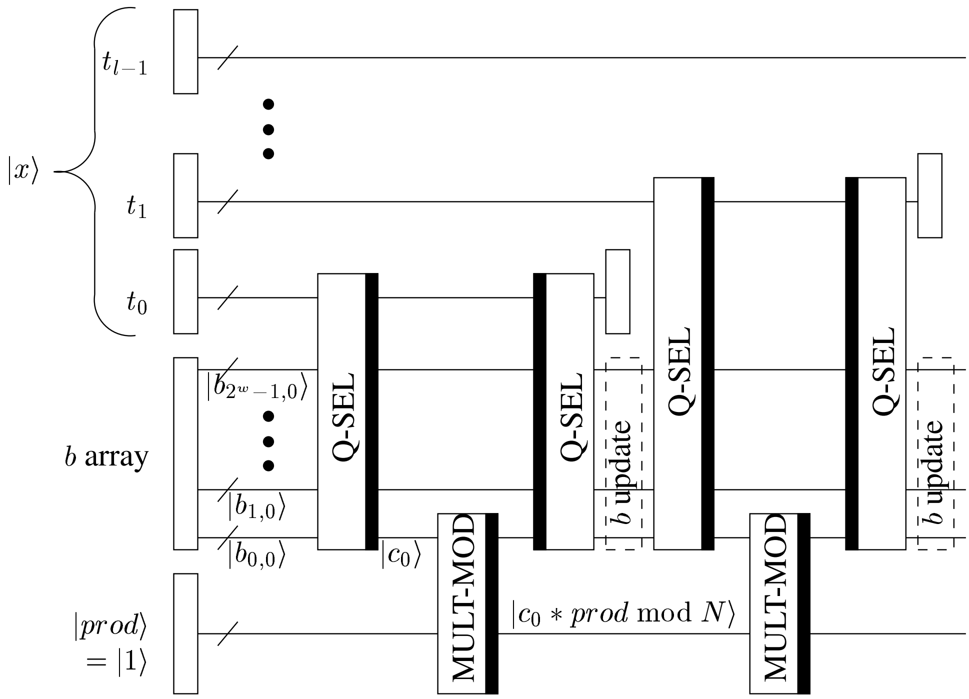

Figure 3. Multiplication Using Indirection, Based on Vedral's Circuit

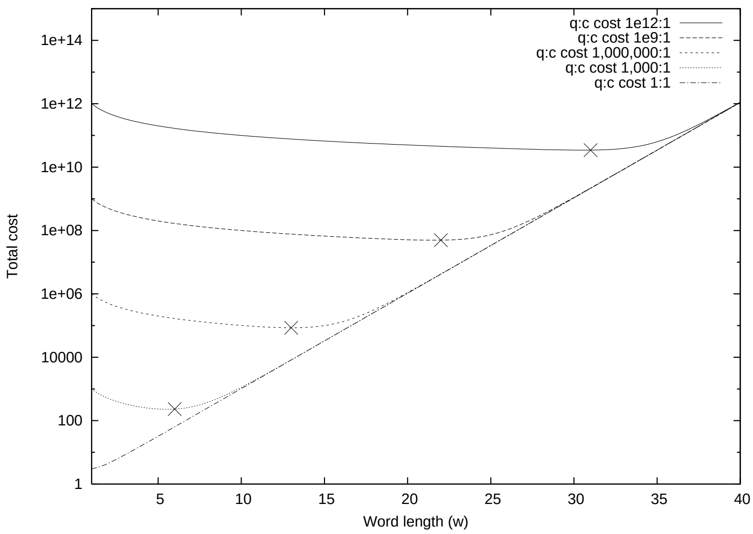

Figure 4. Total Cost

quantum cost increases.

The cost should be normalized to include the differing numbers of times the quantum and classical calculations must be performed. The classical results can be calculated once, and stored and reused. The quantum algorithm must be rerun until success is achieved. Using Barenco's bound, for $m = 14$ and $L = 4096$, we calculate a probability of success using the AQFT of $5.8 * 10^{-6}$. If it takes one hundred thousand runs, on average, to succeed, then the quantum cost used in the graph should be one hundred thousand times the one-time cost.

This graph is somewhat simplified, in that the cost ratio is treated as fixed. In reality, the QACM cost will almost certainly depend on the word length.

Commodity microprocessors may be as much as 10^{13} times as fast as quantum computing devices. Combined with the success probability, it is clear that the limitation on w will be the practical size of the b array rather than computational cost.

6. Conclusions

In this paper, we have shown how the standard computer science technique of indirection can be used in the quantum domain to accelerate the modular exponentiation that is the primary cost of Shor's algorithm. This technique reduces the number of multiplications necessary, and is independent of the multiplication algorithm chosen. The price we pay for this is a large classical/quantum tradeoff; we perform 2^w more multiplications in the classical domain in exchange for reducing the quantum multiplications by a factor of w. We expect that this basic technique will apply to other algorithms, as well.

References

1. A. Barenco, A. Ekert, K.-A. Suominen, and P. Törma. Approximate quantum fourier transform and decoherence. *Physical Review A*, 54:139–146, 1996.
2. P. Gossett. Quantum carry-save arithmetic. http://arXiv.org/quant-ph/9808061, Aug. 1998.
3. N. Kunihiro. Practical running time of factoring by quantum circuits. In *Proc. ERATO Conference on Quantum Information Science (EQIS2003)*, Sept. 2003.
4. M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*, pages 266–268. Cambridge University Press, 2000.
5. P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. 35th Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society Press, 1994.
6. V. Vedral, A. Barenco, and A. Ekert. Quantum networks for elementary arithmetic operations. *Phys. Rev. A*, 54:147–153, 1996. http://arXiv.org/quant-ph/9511018.

Acknowledgments

Thanks to Prof. Kunihiro of the University of Electro-Communications, Drs. Kawano and Takahashi of NTT CRL, Prof. Iwama of Kyoto University, Prof. Yamashita of NAIST, Dr. Yamaguchi of Stanford, and Prof. Itoh and Eisuke Abe of Keio University. This work was supported in part by Ken Adelman and Dave Kashtan under the Network Alchemy basic research funding plan.