慶應義塾大学理工学部　物理情報工学科　Keio University

Fast Quantum Modular Exponentiation

Rodney Van Meter rdv@tera.ics.keio.ac.jp

EQIS 2004

Goal

Establish fast quantum modular exponentiation algorithms to run Shor's algorithm on various quantum computer architectures, using the best available quantum and classical techniques. This will support performance analysis of both architectures and algorithms, benefiting near-term experimentation and long-term research planning.

Basic Idea

Speed is critical, and subject to:

- clock speed
- algorithm
- architecture

In this work, we have concentrated on the latter two elements, understanding the constraints of architecture and searching for the best algorithms to match each architecture.

Architectural Features

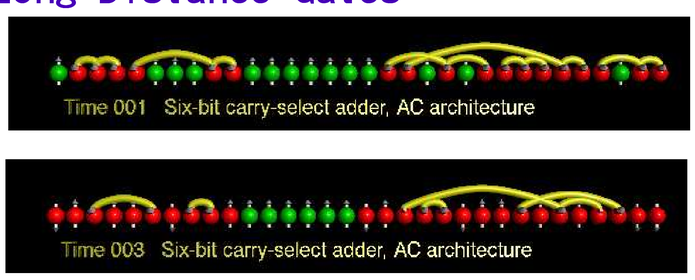
The three most important features of an architecture, in terms of its ability to run algorithms efficiently, are:

- number of qubits
- the ability to run multiple gates concurrently
- the interconnect capability of the system: can gates happen between neighbors only, or any arbitrary pair of qubits?

We use architectural descriptions with important features that help us understand performance. AC supports long-distance gates, while NTC supports only operations on nearest neighbors.

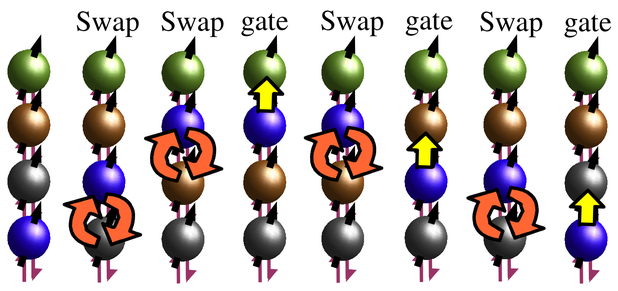
AC: Long-Distance Gates

Two frames from an animation showing long-distance gates on an AC architecture. O(log n) possible, depending on algorithm.

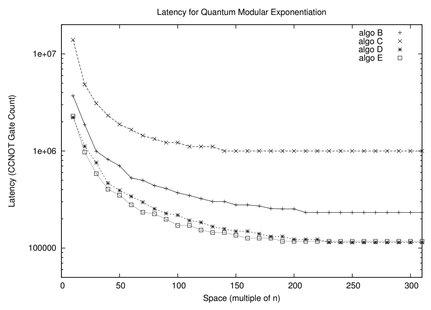


NTC: 1D Layout, Neighbors Only

Position first, then do action gates
Only 1 new neighbor after each swap
What's the performance penalty?
O(n) performance at best.



Taking Advantage of Space



The primary use of large amounts of space is parallel multiplication.
Graph is execution latency (circuit depth) for exponentiating 128 bits on an AC architecture, for several different algorithms.

Modular Exponentiation in Shor's Algorithm

Modular exponentiation is the dominant cost in Shor's algorithm. *2n* classical modular multiplications and *n* quantum modular multiplications are used in the standard method.

Let $d_j = a^{2^j} \bmod N$

d_j values are computed classically[3].

To factor the *n* bit number N, we must evolve to hold: $\frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} |\, a^j \bmod N \,\rangle$

Binary expansion of $x : x_{n-1} x_{n-2} \cdots x_0$

$$a^x \bmod N = \prod_{j=0}^{n-1} d_j^{x_j} \bmod N$$

Faster Algorithms

We use a variety of techniques to accelerate the basic computations in modular exponentiation. Parallel multiplication is critical, and improvements in modulo arithmetic and argument handling help.

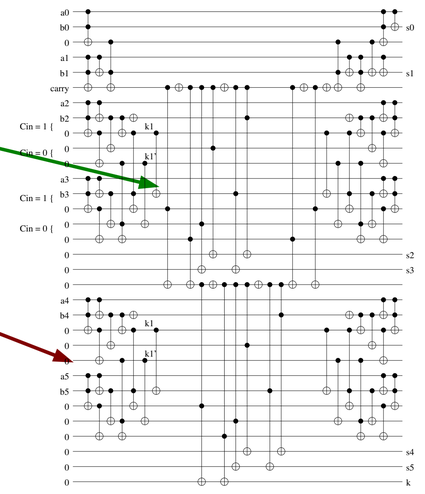
Below we show one of the most important optimizations, a faster adder, and compare it to the standard one.

Conditional-Sum Adder

O(log n) latency when long-distance gates are free.
O(n) when swap required (NTC architecture) -- with a big constant!

Better use of concurrent gates (total still O(n) or larger).

(Carry-save and carry-lookahead are other types that reach O(log n). See quant-ph/9808061, quant-ph/0406142.)



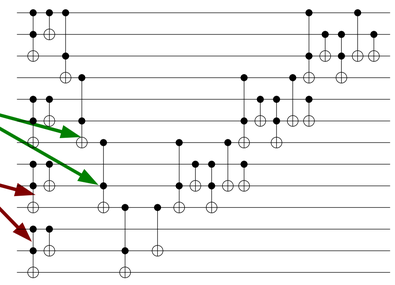
Carry-Ripple Adder

O(n) latency on all architectures

Carry ripples one bit at a time

(Can use concurrent gates here, but limited above)

(Variants used in both major exponentiation algorithms, VBE (quant-ph/9511018) and BCDP (quant-ph/9602016).)



Results

Algorithms that are up to 700 times faster than basic algorithms, when 100n qubits of storage are available, for factoring a 128-bit number. Our algorithms are O(n^2 log n) for nearest-neighbor architectures, and O(n log^2 n) for long-distance-gate architectures.

Latency for Mod Exp (128 bits)

	AC gates	perf	TC gates	perf	NTC gates	perf
VBE	1.25E+08	1	4.99E+08	1	8.32E+08	1
BCDP	4.96E+07	2.5	1.32E+08	3.7	4.64E+08	1.8
VBE (100n)	7.56E+06	16	3.03E+07	16	5.05E+07	17
BCDP (100n)	2.53E+06	49	6.71E+06	74	2.36E+07	35
A	2.65E+07	4.7	1.07E+08	4.7	1.77E+08	4.7
B	3.71E+05	336	1.38E+06	360	1.71E+07	49
C	1.22E+06	102	4.89E+06	102	8.11E+06	103
D	2.19E+05	570	N/A		N/A	
E	1.71E+05	727	N/A		N/A	

700x faster on AC,
100x faster on NTC!

Algorithms **B, C, D, E, VBE(100n), BCDP(100n)** use 100n storage; others use 5n-7n
gates for AC are CCNOT, others are CNOT

Future Work

We are currently examining the full impact of clock speed, and have begun investigations into different physical topologies and performing these algorithms on fault-tolerant, error-corrected qubits.

References

1) P. Shor, "Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM J. Comp.,* 1997.
2) R. Van Meter and K. M. Itoh, "Fast Quantum Modular Exponentiation," quant-ph/0408006.
3) V. Vedral *et al.,* PRA, 1996 (VBE algo.).
4) D. Beckman *et al.*, PRA, 1996 (BCDP algo.).
5) Fowler *et al.*, QIC, 2004, quant-ph/0402196.
6) Draper *et al.*, EQIS 2004, quant-ph/0406142.