AFRL-IF-RS-TR-2007-12
Final Technical Report
January 2007

ARCHITECTURES AND APPLICATIONS FOR SCALABLE QUANTUM INFORMATION SYSTEMS

Massachusetts Institute of Technology

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. L484

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

AFRL-IF-RS-TR-2007-12 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
STEVEN L. DRAGER Work Unit Manager	JAMES A. COLLINS, Deputy Chief Advanced Computing Division Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* JAN 2007	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* Jun 01 – Jun 06

4. TITLE AND SUBTITLE

ARCHITECTURES AND APPLICATIONS FOR SCALABLE QUANTUM INFORMATION SYSTEMS

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8750-01-2-0521

5c. PROGRAM ELEMENT NUMBER
62716E

6. AUTHOR(S)

Isaac Chuang

5d. PROJECT NUMBER
L484

5e. TASK NUMBER
AA

5f. WORK UNIT NUMBER
SQ

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Massachusetts Institute of Technology
77 Massachusetts Ave
Cambridge MA 02139

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 N. Fairfax Dr.
Arlington VA 22203

AFRL/IFTC
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2007-12

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 07- **018**

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The goal of this project was to understand what key interchangeable elements form a scalable, fault-tolerant quantum information systems architecture. The effort was a collaboration between computer science and physical sciences involving four groups: MIT, providing experimental quantum technology parameters and fundamental expertise in quantum information theory; UC Davis, devising fault-tolerant architecture designs and implementing numerical simulations; UC Berkeley, creating quantum cryptosystems and providing distributed applications; and U Washington, focusing on languages for quantum computation and an architectural simulator. Accomplishments of project include: design of several complete quantum architectures for large-scale, reliable quantum computers; implementation of a predictive design-tool to analyze system reliability given technology parameters and constraints; evaluation of requirements and performance of Shor's factoring algorithm on a complete benchmark quantum architecture design; and design of experimental realizations of experiments to identify crucial parameters for fault-tolerant quantum architectures.

15. SUBJECT TERMS
Quantum Computing, Scalable Fault tolerant Quantum Information Processing, Quantum Computer Systems, Architecture, Quantum Computer Aided-Design Tools

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Steven L. Drager
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UL	117	**19b. TELEPHONE NUMBER (*Include area code)***

Contents

List of Figures

1 Introduction

This project initiated 06/29/2001 with the goals of designing and testing architectures and applications for a scalable, fault-tolerant, quantum information processing system. Our project was a collaborative computer science and physical sciences effort involving four groups working in tight coordination, with MIT providing experimental quantum technology parameters and fundamental expertise in quantum information theory, U.C. Davis devising fault-tolerant architecture designs and implementing numerical simulations, U.C. Berkeley creating quantum cryptosystems and providing distributed applications pull, and U. Washington focusing on new languages for quantum computation, and an architectural simulator.

Our principle aim was to: design a complete *system architecture* for a realistic *programmable, arbitrary-scale* quantum computer, focusing on maximum reduction of overhead and reaching towards two targets: a solid-state spin-based quantum computer, and application of quantum information to the real-world problems in secure distributed information storage.

We report the following major accomplishments over the span of this project timeline, from 06/29/01 to 03/31/06:

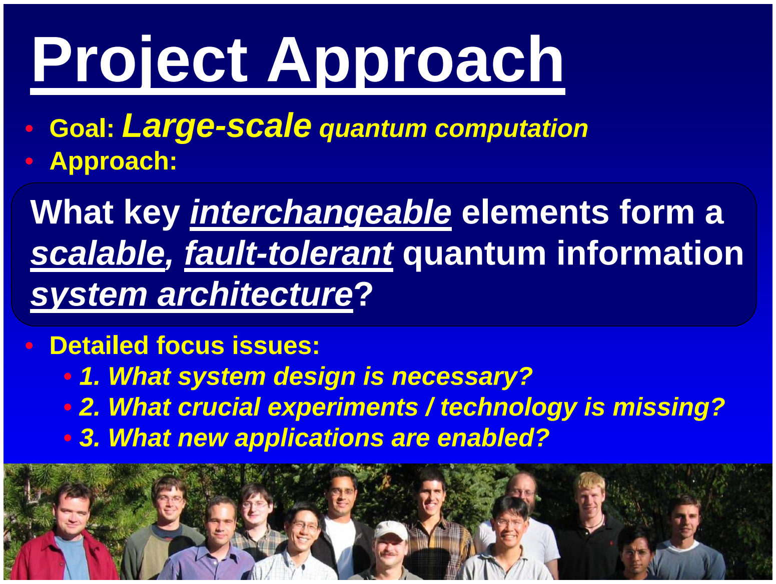
- Detailed architecture design for large-scale, fault-tolerant quantum computers:
    1. Identification of quantum wires and quantum communication requirements as a key bottleneck in quantum processor design (ISCA'03 paper)
    2. Design of quantum FPGA and micro-architecture processor layout for optimized performance and reduced overhead (ISCA'05 and ISCA'06 papers)
- Software tool-chain for quantum computer architecture design and analysis
    1. Development of systematic scalability criteria for reliable large-scale quantum computers (IEEE Computer'02)
    2. Implementation of predictive quantum CAD tools for quantum architecture design and evaluation (IEEE Computer '06)
- Enabling experimental results for quantum computation

1. Realization of quantum optimization algorithm (Phys. Rev. Lett '03)
2. Implementation of first standard quantum algorithm with a trapped ion (Nature'04)

Major publications detailing these results, specifically including those cited here, appear in the appendix of this report (cf VIII Article 28 C.2 of the project agreement).

2 Project motivation, approach, and timeline

The main goal of this project was a quest to understand what key *interchangeable* elements form a *scalable, fault-tolerant* quantum information *system architecture* (Fig. 1). Our focus was squarely on large-scale quantum computation, meaning a quantum computer capable of solving problems such as factorization of 1024 bit numbers, which can reasonably be expected to require $\mathcal{O}(10^6)$ qubits even when using perfect hardware. With imperfect hardware, quantum error correction is required, which requires an overhead of perhaps 10^3 to 10^6 times more qubits, depending on the base error rate.



Project Approach

- Goal: *Large-scale quantum computation*
- Approach:

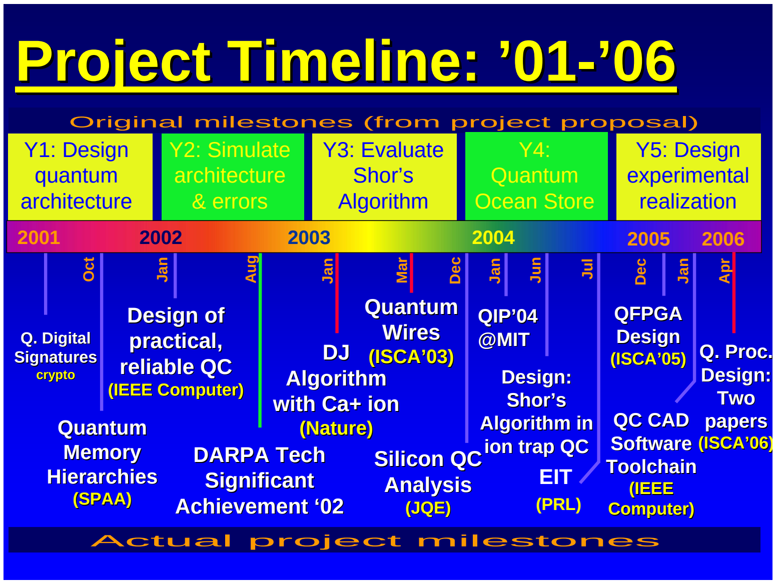
What key ***interchangeable*** elements form a ***scalable***, ***fault-tolerant*** quantum information ***system architecture***?

- Detailed focus issues:
  - *1. What system design is necessary?*
  - *2. What crucial experiments / technology is missing?*
  - *3. What new applications are enabled?*

Figure 1: Summary of main project problem and approach.

Our approach to addressing this problem was to focus on three main issues: (1) the system design elements necessary for large-scale, reliable quantum computation, (2) the missing experiments or technology necessary to realize complete quantum computing systems, and (3) new applications which might be enabled by large-scale quantum computation.

Crucially, we realized early in the project that we would have to focus on designing reliable quantum computers from unreliable parts, because of the inevitability of a high rate of errors in quantum systems due to the intrinsic nature of decoherence. Quantum states do not readily remain in superposition, and usually require active correction to maintain useful stability.



Project Timeline: '01-'06

Original milestones (from project proposal)

Y1: Design quantum architecture

Y2: Simulate architecture & errors

Y3: Evaluate Shor's Algorithm

Y4: Quantum Ocean Store

Y5: Design experimental realization

Q. Digital Signatures crypto

Design of practical, reliable QC (IEEE Computer)

Quantum Memory Hierarchies (SPAA)

DARPA Tech Significant Achievement '02

DJ Algorithm with Ca+ ion (Nature)

Quantum Wires (ISCA'03)

Silicon QC Analysis (JQE)

QIP'04 @MIT

Design: Shor's Algorithm in ion trap QC

EIT (PRL)

QFPGA Design (ISCA'05)

QC CAD Software Toolchain (IEEE Computer)

Q. Proc. Design: Two papers (ISCA'06)

Actual project milestones

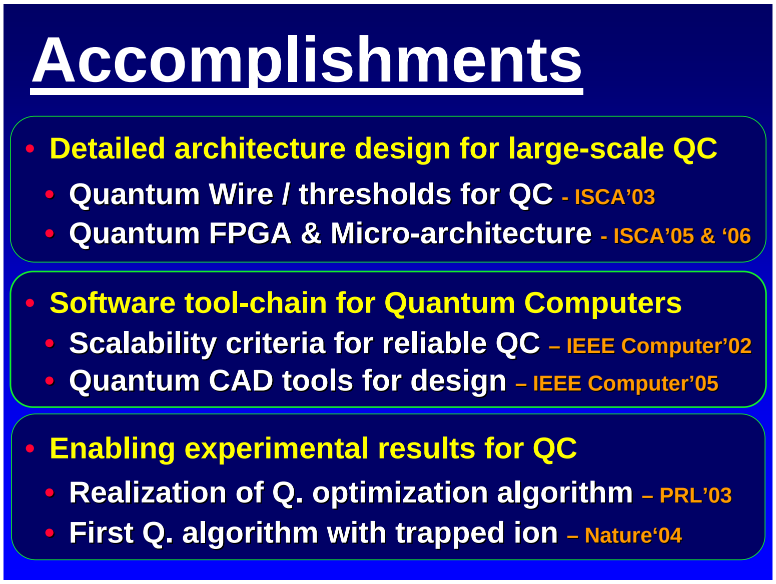
Figure 2: Timeline of major project results, in comparison with original goals.

The project started with five main goals (Fig. 2): design of a quantum architecture, simulation of architectures constructed from faulty components, evaluation of a benchmark application (Shor's algorithm), application to a large-scale distributed classical computing system (Ocean Store), and design and evaluation of a possible experimental realization of a full large-scale, reliable quantum computing system.

3 Project accomplishments and self-evaluation

The actual goals which we successfully accomplished during this project matched four of the five original goals quite well, and there were unanticipated surprises as well (Fig. 3).

The four originally anticipated goals which we accomplished were:



Accomplishments

- **Detailed architecture design for large-scale QC**
  - **Quantum Wire / thresholds for QC** - ISCA'03
  - **Quantum FPGA & Micro-architecture** - ISCA'05 & '06
- **Software tool-chain for Quantum Computers**
  - **Scalability criteria for reliable QC** – IEEE Computer'02
  - **Quantum CAD tools for design** – IEEE Computer'05
- **Enabling experimental results for QC**
  - **Realization of Q. optimization algorithm** – PRL'03
  - **First Q. algorithm with trapped ion** – Nature'04

Figure 3: Summary of major project accomplishments.

- The design of a variety of complete quantum architectures for large-scale, reliable quantum computers, including identification of quantum wires, memories, control systems, and spatial layout as key design elements (IEEE Computer'02, SPAA'03, and ISCA'03 & '05 papers)

- Simulation of architectures of reliable quantum computers, and the implementation of a predictive design-tool to analyze system reliability given technology parameters and constraints (IEEE Computer '02 and '06 papers)

- Evaluation of the requirements and possible performance of Shor's factoring algorithm on a complete benchmark quantum architecture design (ISCA'05 and '06 papers)

- Design of potential experimental realizations and implementation of actual experiments to identify crucial parameters for fault-tolerant quantum architectures (Nature'04, JQE'03, PRL'04, and ISCA'06 papers)

The single goal which we did not make much progress on was our hope to identify new applications for large-scale quantum information processors in secure, distributed classical computing applications such as Ocean Store. Actually, early in the project, we proved that one of our original ideas, to enable secure remote computation using quantum protocols, was not possible. However, near the start of the project, in 2001, we successfully completed a study on quantum digital signatures, which can be used for authentication and transferable authentication in multiparty systems.

Two of the most interesting lessons we learned in this project were perhaps that (1) fault-tolerance is a crucial concept which is widely invoked but not well understood, particularly with respect to resource requirements, and (2) surprisingly, solid state (silicon based) quantum computing was initially very promising, but ultimately unrealistic due to quantum communication and wiring needs.

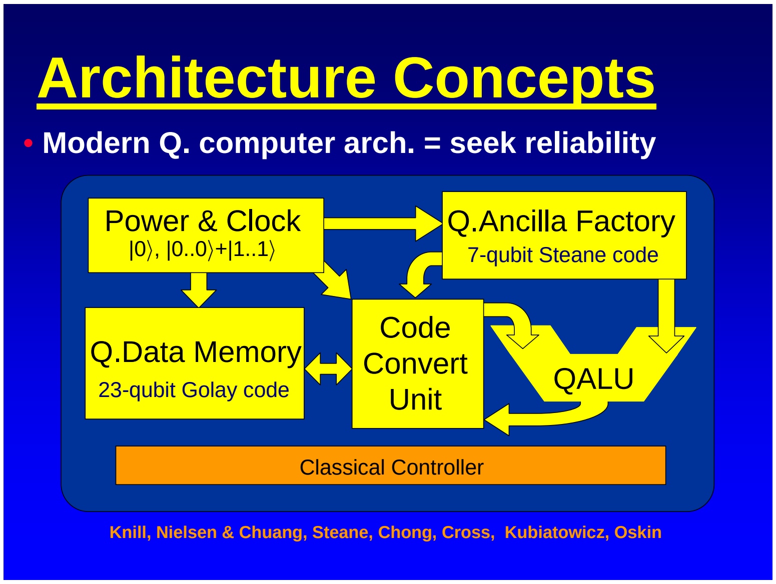
And perhaps the most interesting unanticipated success we had was with another technology: trapped ion quantum computation. Originally, we did not have great hopes for that technology, but upon detailed study and modeling, it turned out to be extremely promising, much more so than other currently available quantum computing technologies. We identified this promise in 2003, just as experimental successes at NIST and the University of Innsbruck were coming about, demonstrating basic quantum protocols such as quantum teleportation, quantum error correction, and simple quantum algorithms, such as the Deutsch-Jozsa algorithm which we played a role in making possible.

4 Summary of Main Scientific Results

Three main accomplishments of this project deserve special identification: our results in architecture design, scalability criteria, and software design tools for fault-tolerant quantum architectures.

4.1 Quantum architecture concepts

Modern computer architecture is about the optimization of one central goal: *parallelism.* Modern CPUs employ concepts such as functional unit specialization, speculative execution, H-tree clock distribution, and subsystem power control, to maximize performance and minimize energy cost. Quantum architecture focuses on a different central goal: *reliability*, because quantum noise is an unavoidable fact in any realistic quantum computer implementation, and must be managed carefully from a systems approach.



Architecture Concepts

- Modern Q. computer arch. = seek reliability

Power & Clock
$|0\rangle$, $|0..0\rangle+|1..1\rangle$

Q.Ancilla Factory
7-qubit Steane code

Q.Data Memory
23-qubit Golay code

Code Convert Unit

QALU

Classical Controller

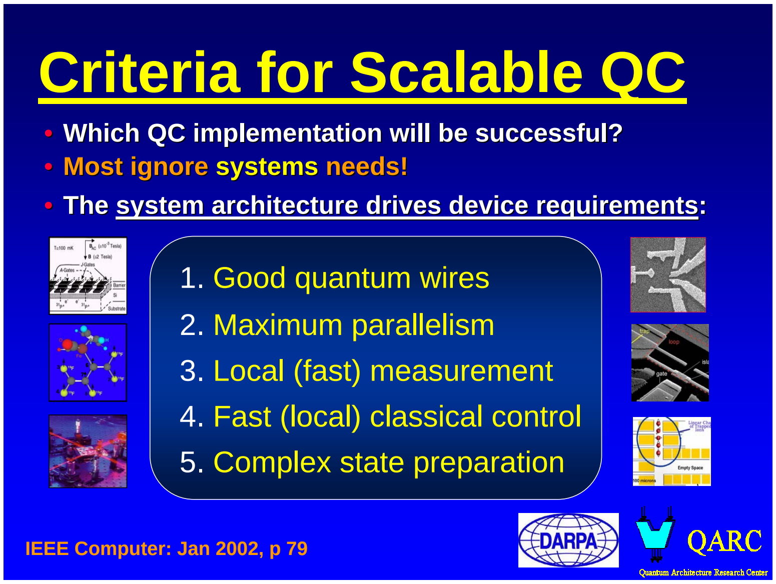
Knill, Nielsen & Chuang, Steane, Chong, Cross, Kubiatowicz, Oskin

Figure 4: Illustrative major result: concepts for quantum processor design.

One major result of this project was our introduction of new concepts for quantum architecture (Fig. 4), including entropy exchange units, code conversion teleportation, quantum wires, entanglement based clocks, and entanglement "power sources." From this work, we developed a considerable understanding of the overall cost of fault tolerance in quantum computation, and how this can be reduced through design improvements in balancing memory, computation, and communication (see, in particular, our ISCA'02, ISCA'05, and ISCA'06 papers). Our results lead to building-block based designs that are conceptually clean, buildable, and debuggable.

4.2 Scalability criteria

The overall system cost of fault-tolerance in quantum architectures is very high, but many elements of this cost have largely been neglected in the community until our work. For example, the number of wires required grows exponentially with the number of levels of concatenation used in fault tolerance constructions, and moreover, these wires must generally connect gates spatially separated by the entire size of the code block! Naturally, providing such communication capability cannot come for free, but this cost was disregarded in all early feasibility claims for quantum computing proposals. In particular, results early in our project (ISCA'03 and related papers) identified the cost of such quantum communication needs as the primary pitfall in constructing realistic quantum computing systems based on solid state devices with fixed qubits, such as Kane's original impurity-based qubit scheme. Another widely neglected issue is the cost of the classical control system needed to schedule, perform, and stabilize fault-tolerant quantum gates. This control system must be much more reliable, faster, and more parallel than the quantum system it controls, so for example, if qubits are running with a nanosecond timescale clock, the classical control system should run with a subnanosecond timescale clock. For many quantum computing technologies, this would require a control system that is far too power-hungry to be realistic, due to cryogenic cooling requirements for the qubit implementation.



Criteria for Scalable QC

- Which QC implementation will be successful?
- Most ignore systems needs!
- The system architecture drives device requirements:

1. Good quantum wires
2. Maximum parallelism
3. Local (fast) measurement
4. Fast (local) classical control
5. Complex state preparation

IEEE Computer: Jan 2002, p 79

DARPA QARC Quantum Architecture Research Center

Figure 5: Illustrative major result: criteria for scalable fault-tolerant quantum computation.

More broadly, through our project work over four years, we have identified a set of five criteria (Fig. 5), initially sketched in our IEEE Computer'02 article, which must be satisfied for a fault-tolerant quantum architecture to obtain realistic performance. These criteria stipulate that a good quantum computing system much satisfy not just the normal DiVincenzo criteria, but also must have:

1. Good quantum wires: the ability to move quantum information between nearly any two

points in a quantum processor at a reliability high compared with gate error probabilities.

2. Maximum parallelism: the ability to perform multiple quantum gates in a single timestep.

3. Local (fast) measurement: the ability to measure nearly any qubit in the system, without requiring more than a constant amount of movement, and in a time comparable to a single gate.

4. Fast (local) classical control: facilities for gates to be applied to qubits at the proper time and place, measurement results to be used in feedback to correct errors, code syndromes to be extracted and voted upon, and qubits to be scheduled and moved for inter-gate communication.

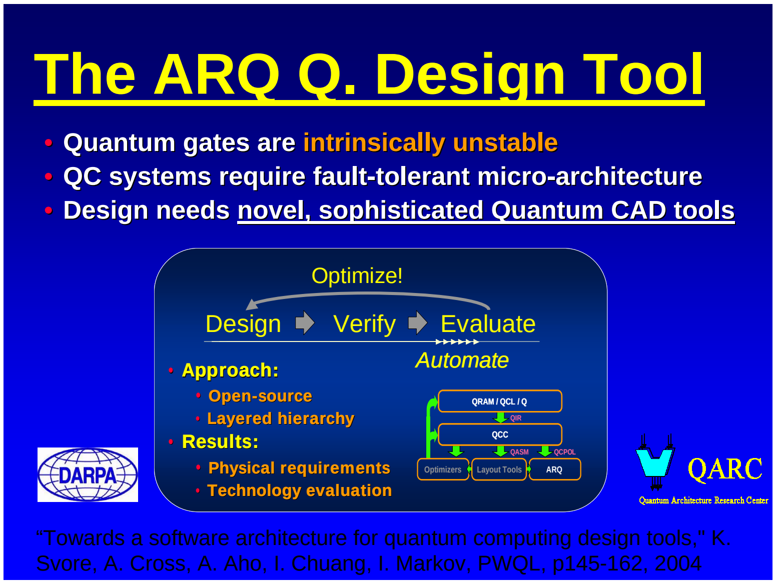
5. Complex state preparation: the ability to prepare not just $|0\rangle$ states, but also complex states such as logically encoded $|0_L\rangle$ and $|\pm_L\rangle$ states, Bell, and cat states, $|00\cdots0\rangle + |11\cdots1\rangle$.

The lack of any of these elements will likely result in a significant worsening of the fault tolerance threshold for the system, compared to the ideal threshold determined by code properties. We have studied and quantified such costs in many of our publications; see, for example, the S.M. thesis of Andrew Cross (available at the MIT DSpace archive permanent URL `http://hdl.handle.net/1721.1/30175`).

4.3 Software tool-chain for quantum CAD

Modern computers are designed first, then built afterwards. This is made possible by the use of predictive software tools, which allow computer aided design of models which accurately reflect the performance of actual chip implementations. One major result of this project was the development of several suites of new tools for predictive analysis and simulation of quantum architectures (Fig. 6).

Unlike classical CAD tools, which focus on just performance optimization and aiding in the design of complex systems, these quantum CAD tools focus on achieving and evaluating reliability. We began with an initial tool developed to study solid-state implementations, then turned



The ARQ Q. Design Tool

- Quantum gates are intrinsically unstable
- QC systems require fault-tolerant micro-architecture
- Design needs novel, sophisticated Quantum CAD tools

Optimize!

Design ➡ Verify ➡ Evaluate

Automate

- Approach:
  - Open-source
  - Layered hierarchy
- Results:
  - Physical requirements
  - Technology evaluation

QRAM / QCL / Q

QIR

QCC

QASM

QCPOL

Optimizers

Layout Tools

ARQ

DARPA

QARC

Quantum Architecture Research Center

"Towards a software architecture for quantum computing design tools," K. Svore, A. Cross, A. Aho, I. Chuang, I. Markov, PWQL, p145-162, 2004

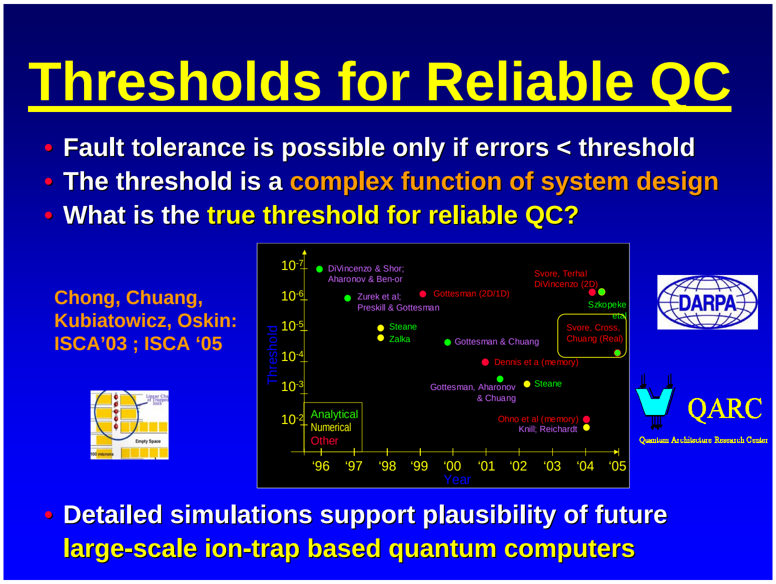
Figure 6: Illustrative major result: modular architectural design tools for large-scale fault-tolerant quantum computation.

this into a full-blown analysis tool for evaluating trapped ion quantum computer architectures. By simulating just quantum error correction circuits, a class which require only "Clifford group" gates that are easily classically simulated, we were able to simulate quantum circuits with $\mathcal{O}(1000)$ qubits efficiently.

We deployed this on a Beowulf cluster to compute fault tolerance thresholds for trapped ion quantum computers, analyzing the impact of specific technology parameters such as ion movement, memory, waiting, one- and two-qubit gates, measurement, and preparation. Using this tool, we computed a first set of new thresholds for fault-tolerant quantum computation in the presence of realistic resource assumptions (Fig. 7).

This software tool has since evolved in several directions, including a separate branch developed at U. Washington, and one at Columbia University in collaboration with Al Aho's group there. His group introduced the idea of using feedback in the simulation to allow optimization of thresholds. The tool has been used in teaching of students in compiler optimization techniques, and is now also a basis for development of a hardware "physical operations" language for basic quantum computer operations.

Based on these results, we believe similar tools can (and should!) be developed to accurately



Thresholds for Reliable QC

- Fault tolerance is possible only if errors < threshold
- The threshold is a complex function of system design
- What is the true threshold for reliable QC?

Chong, Chuang, Kubiatowicz, Oskin: ISCA'03 ; ISCA '05

- Detailed simulations support plausibility of future large-scale ion-trap based quantum computers

Figure 7: Illustrative major result: thresholds for fault-tolerant quantum computation

predict fault tolerance thresholds for quantum computers implemented with other technologies, such as solid state systems, and combinations of technologies. Just as for classical computers, these tools will allow the performance of realistic quantum computing systems to be evaluated and predicted, in advance of actual fabrication, and perhaps even in advance of technology developments, as a strategic tool in directing investments in technology sectors.

5 List of manuscripts submitted/appearing in print

- *Papers in Refereed Journals and Conferences:*

1. L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. Sherwood, and I. L. Chuang. *Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance.* Nature, 414 pp. 883, 2001.

2. M. Oskin, F. Chong, and I. Chuang, "A practical architecture for reliable quantum computers," *IEEE Computer*, vol. 35, p. 79, 2002.

3. Dean Copsey, Mark Oskin, Frederic T. Chong, Isaac Chuang, and Khaled Abdel-Ghaffar. "Memory Hierarchies for Quantum Data" Workhop on Non-Silicon Computing, held

in Conjunction with the International Symposium on High Performance Computer Architecture, Feb. 2002.

4. Tzvetan Metodiev, Dean Copsey, Frederic T. Chong, Isaac Chuang, Mark Oskin, and John Kubiatowicz. ”A Brief Comparison: Ion-Trap and Silicon-Based Implementations of Quantum Computation, ” In the 2nd workshop on Non-Silicon Computing (NSC-2), June 2003.

5. Dean Copsey, Mark Oskin, Andrew Cross, Tzvetan Metodiev, Frederic T. Chong, Isaac Chuang, and John Kubiatowicz. ”A Design Overview for a Simulation Infrastructure for Exploring Quantum Architectures, ” In the 2nd workshop on Non-Silicon Computing (NSC-2), June 2003.

6. Andrew Petersen and Mark Oskin. A New Algebraic Foundation for Quantum Programming Languages In the 2nd workshop on Non-Silicon Computing (NSC) held in conjunction with the 30th Annual International Symposium on Computer Architecture (ISCA), June 2003.

7. D. Copsey, M. Oskin, T. Metodiev, F.T. Chong, and I. Chuang. “The Effect of Communication Costs in Solid-state Quantum Architectures,” in the Symposium on Parallel Architectures and Applications (SPAA), p. 65-64, 2003.

8. M. Oskin, F. Chong, I. Chuang, and J. Kubiatowicz, “Building quantum wires: the long and the short of it,” *International Symposium on Computer Architecture*, (ISCA) p. 374-385, 2003.

9. Steffen, van Dam, Hogg, Breyta, Chuang, “Experimental implementation of an adiabatic quantum optimization algorithm,” Phys. Rev. Lett., vol 90, no 067903, Feb 2003.

10. Guide et al, *Implementation of the Deutsch-Jozsa algorithm on an ion-trap quantum computer*, *Nature*, vol 421, p 48, 2003.

11. D. Copsey, M. Oskin, F. Impens, T. Metodiev, A. Cross, F. Chong, I. Chuang, and J. Kubiatowicz, “Toward a Scalable, Silicon-Based Quantum Computing Architecture,” *Journal of Selected Topics in Quantum Electronics*, Vol. 9, No. 6, pp 1552-1569, 2003.

12. "Can we build Classical Control Circuits for Silicon Quantum Computers?," Mark Whitney, Yatish Patel, Nemanja Isailovic, and John Kubiatowicz. Appears in Proceedings of the Second Workshop in Non-Silicon Computing (NSC2), June 2003.

13. N. Isailovic, M. Whitney, Y. Patel, J. Kubiatowicz, D. Copsey, F. Chong, I. Chuang, and M. Oskin. "Datapath and Control for Quantum Wires," *Transactions on Architecture and Code Optimization*, (TACO), vol. 1, no. 1, p. 34-61, 2004.

14. "Preliminary Results On Simulating a Scalable Fault Tolerant Ion-Trap System for Quantum Computation," Tzvetan Metodiev, Andrew Cross, Darshan Thaker, Kenneth Brown, Dean Copsey, Frederic T. Chong, and Isaac L. Chuang. In the Proceedings of the Third Workshop on Non-Silicon Computing held with the 2004 International Symposium on Computer Architecture.

15. L. Vandersypen and I. Chuang. "NMR Techniques for Quantum Control and Computation," *Rev. Mod. Phys.*, vol. 76, p. 1037, 2004.

16. "Towards a software architecture for quantum computing design tools," K. Svore, A. Cross, A. Aho, I. Chuang, I. Markov, Proceedings of the workshop on Quantum Programming Languages, p145-162, 2004.

17. K. Murali, H.B. Son, M. Steffen, P. Judeinstein, and I. Chuang. "Test by NMR of the phase coherence of electromagnetically induced transparency" *Phys. Rev. Lett.*, vol. 93, p. 033601, 2004.

18. "Efficient Quantum Circuits for Schur and Clebsch-Gordon Transforms," D. Bacon, A. Harrow, and I. Chuang, quant-ph/0407082.

19. Dean Copsey, Mark Oskin and Frederic T. Chong, "Ions, Atoms, and Bits: An Architectural Approach to Quantum Computing" book Chapter in Advances in Computers, Volume 61 2004.

20. T. Yu, K. Brown, and I. Chuang, "Bounds on the entanglability of thermal states in liquid-state nuclear magnetic resonance," *Phys. Rev. A*, vol. 71, p.32341, 2005.

21. T.S. Metodi, D. Thaker, A.W. Cross, F.T. Chong, and I. Chuang, "A general purpose

architectural layout for arbitrary quantum computations," *Proc. SPIE*, vol. 5815, p. 91-102, 2005.

22. D. Thaker, R. Amirtharajah, F. Impens, I. Chuang, and F.T. Chong, "Recursive TMR: scaling fault tolerance in the nanoscale era," *IEEE Design & Test of Computers*, vol. 22, p. 298-305, 2005.

23. Tzvetan S. Metodi, Darshan D. Thaker, Andrew W. Cross, Frederic T. Chong, and Isaac L. Chuang. "A Quantum Logic Array Microarchitecture: Scalable Quantum Data Movement and Computation, " The 2005 International Symposium on Microarchitecture (MICRO-38), Barcelona, Spain.

24. Tzvetan Metodi, Darshan Thaker, Andrew Cross, Frederic T. Chong, and Isaac L. Chuang. "A General Purpose Architectural Layout for Arbitrary Quantum Computations, " In Proceedings for the SPIE Defense & Security symposium of 2005, Orlando, FL.

25. Steven Balensiefer, Lucas Kregor-Stickles, Mark Oskin. An Evaluation Framework and Instruction Set Architecture for Ion-Trap based Quantum Micro-architectures, The International Symposium on Computer Architecture (ISCA) 2005.

26. Steven Balensiefer, Lucas Kregor-Stickles, Mark Oskin. QUALE: Quantum Architecture Layout Evaluator, SPIE 2005.

27. C.E. Pearson, D.R. Leibrandt, W.S. Bakr, W.J. Mallard, K.R. Brown, and I.L. Chuang, "Experimental investigation of planar ion traps," *Phys. Rev. A*, to appear, 2006.

28. K. Svore, A. Aho, A. Cross, I. Chuang, and I. Markov, "A layered software architecture for quantum computing design tools", *IEEE Computer*, vol. 39, p. 74, 2006.

29. Darshan D. Thaker, Tzvetan S. Metodi, Andrew Cross, Isaac L. Chuang and Frederic T. Chong "Quantum Memory Hierarchies: Efficient Designs to Match Available Parallelism in Quantum Computing, " International Symposium on Computer Architecture (ISCA-33), Boston, MA, 2006.

30. Tzvetan Metodi, Darshan Thaker, Andrew Cross, Frederic T. Chong, and Isaac L. Chuang. "Scheduling Physical Operations in a Quantum Information Processor, " In Proceed-

ings for the SPIE Defense & Security symposium, Orlando, FL. April, 2006.

31. Interconnection Networks for Scalable Quantum Computers, Nemanja Isailovic, Yatish Patel, Mark Whitney, and John Kubiatowicz. Appears in Proceedings of the 33rd International Symposium on Computer Architecture (ISCA-33), Boston, MA 2006.

32. K.R. Brown, R.J. Clark, and I.L. Chuang, "Limitations of quantum simulation examined by simulating a pairing Hamiltonian using Nuclear Magnetic Resonance," Phys. Rev. Lett, vol. 97, p. 050504, 2006.

- *Ph.D. and Master's Theses:*

33. "A Prototype Quantum Computer Using Nuclear Spins in Liquid Solution," Matthias Steffen, Ph.D. Thesis, Stanford University, June 2003.

34. "A Parallel Environment for Simulating Quantum Computation," Geva Patz, S.M. Thesis, MIT, June 2003.

35. "Synthesis and evaluation of fault-tolerant quantum computer architectures" Andrew Cross, S.M. Thesis, MIT, February 2005.

6 Scientific personnel supported and honors/awards/degrees

John Kubiatowicz (Prof., U. C. Berkeley)

Fred Chong (Prof., U.C. Davis)

Isaac Chuang (Prof., MIT)

Mark H. Oskin (Prof., U. Washington)

Dean Copsey (Ph.D. 2004, U.C. Davis)

Andrew Houck (Ph.D. awarded June'06, MIT)

Andrew Cross (Ph.D. student, fellowship, MIT)

Francois Impens (M.S., degree awarded June'04, MIT)

Nemanja Isailovic (Ph.D. student, U.C. Berkeley)

John McCanne (Ph.D. student, fellowship, U.C. Davis)

Yatish Patel (Ph.D. student, U.C. Berkeley)

Geva Patz (M.S., degree awarded June'03, MIT)

Chris Pearson (M.S. student, degree anticipated June'06, MIT)

Andrew Petersen (Ph.D. student, U. Washington)

Ravishankar Rao (Ph.D. student, U.C. Davis)

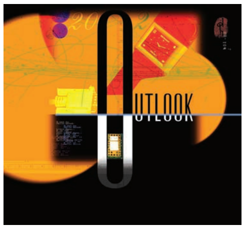
Matthias Steffen (Ph.D., degree awarded June'03 Stanford/MIT)

Mark Whitney (Ph.D. student, U.C. Berkeley)

A Reference papers

A.1 Computer'02 Paper

A Practical Architecture for Reliable Quantum Computers



Quantum computation has advanced to the point where system-level solutions can help close the gap between emerging quantum technologies and real-world computing requirements.

Mark Oskin
University of Washington

Frederic T. Chong
University of California, Davis

Isaac L. Chuang
Massachusetts Institute of Technology

Quantum computers offer the prospect of computation that scales exponentially with data size. Unfortunately, a single bit error can corrupt an exponential amount of data. Quantum mechanics can seem more suited to science fiction than system engineering, yet small quantum devices of 5 to 7 bits have nevertheless been built in the laboratory,[1,2] 100-bit devices are on the drawing table now, and emerging quantum technologies promise even greater scalability.[3,4]

More importantly, improvements in quantum error-correction codes have established a threshold theorem,[5] according to which scalable quantum computers can be built from faulty components as long as the error probability for each quantum operation is less than some constant (estimated to be as high as 10^{-4}). The overhead for quantum error correction remains daunting: Current well-known codes require tens of thousands of elementary operations to provide a single fault-tolerant logical operation. But proof of the threshold theorem fundamentally alters the prospects for quantum computers. No principle of physics prevents their realization—it is an engineering problem.

Empirical studies of practical quantum architectures are just beginning to appear in the literature.[6] Elementary architectural concepts are still lacking: How do we provide quantum storage, data paths, classical control circuits, parallelism, and *system* integration? And, crucially, how can we design architectures to reduce error-correction overhead?

QUANTUM COMPUTATION

Quantum information systems can be a mathematically intense subject. We can understand a great deal, however, by using a simple model of abstract building blocks: quantum bits, gates, and algorithms, and the available implementation technologies—in all their imperfections.[7] The basic building block is a quantum bit, or *qubit*, represented by nanoscale physical properties such as nuclear spin. In contrast to classical computation, in which a bit represents either 0 or 1, a qubit represents both states simultaneously. More precisely, a qubit's state is described by *probability amplitudes*, which can destructively interfere with each other and only turn into probabilities upon external observation.

Quantum computers manipulate these amplitudes directly to perform a computation. Because n qubits represent 2^n states, a two-qubit vector simultaneously represents the states 00, 01, 10, and 11—each with some probability when measured. Each additional qubit doubles the number of amplitudes represented—thus, the potential to scale exponentially with data size.

A fundamental problem, however, is that we generally cannot look at the results of a quantum com-

Quantum Algorithms

Recent interest in quantum computers has focused on Peter Shor's algorithm for prime factorization of large numbers.[1] Shor showed that a quantum computer could, in theory, factor an n-bit integer in $O(n^3)$ time.

Shor's discovery drew a lot of attention. The security of many modern cryptosystems relies on the seeming intractability of factoring the product of two large primes, given that the best-known factoring algorithms for a classical computer run in exponential time. To put this in perspective, researchers using the number field sieve have successfully factored a 512-bit product of two primes, but it took 8,400 MIPS years.[2] A 1,024-bit product would take approximately 1.6 billion times longer. That seems intractable.

With Shor's algorithm, you could factor a 512-bit product in about 3.5 hours, assuming the quantum architecture and error-correction schemes described in this article, and a 1-GHz clock rate. Under the same assumptions, the algorithm could factor a 1,024-bit number in less than 31 hours.

Another key algorithm is Lov Grover's for searching an unordered list of n elements in $\sqrt{n}$ queries.[3] Quantum algorithms have also been devised for cryptographic key distribution[4] and clock synchronization.

It is expected, however, that a major application area for quantum computers will be the simulation of quantum mechanical systems that are too complex to be simulated on classical computers.[5] This prospect opens possibilities impossible to imagine in the classical world of our intuition and current computers.

References

1. P. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," *Proc. 35th Ann. Symp. Foundations of Computer Science*, IEEE CS Press, Los Alamitos, Calif., 1994, p. 124.
2. B. Preneel, ed., *Factorization of a 512-Bit RSA Modulus*, vol. 1807, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2000.
3. L. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," *Proc. 28th Ann. ACM Symp. Theory of Computation*, ACM Press, New York, 1996, pp. 212-219.
4. C.H. Bennet, G. Brassard, and A.K. Ekert, "Quantum Cryptography," *Scientific Am.*, Oct. 1992, pp. 50-57.
5. M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2000.

putation until it ends, at which point we get only a random value from the vector. More precisely, measuring a qubit vector collapses it into a probabilistic classical bit vector, yielding a single state randomly selected from the exponential set of possible states. Perhaps for this reason, quantum computers are best at "promise" problems—applications that use some hidden structure in a problem to find an answer that can be easily verified. Such is the case for the application domains of the two most famous quantum algorithms, Shor's for prime factorization of an n-bit integer in $O(n^3)$ time[8] and Grover's for searching an unordered n-element list in $\sqrt{n}$ queries.[9] The "Quantum Algorithms" sidebar provides additional information about applications for these algorithms. Obviously, designers of quantum algorithms must be very clever about how to get useful answers from their computations.

Another problem is that qubits lose their quantum properties exponentially quickly in the presence of a constant amount of noise per qubit. This sensitivity is referred to as *decoherence*, and it is widely believed to be the reason why the world around us is so predominantly classical. Nevertheless, quantum computation can tolerate a finite amount of decoherence, so the engineering problem is to contain it to a sufficiently small amount. The relevant measure is the amount of decoherence per operation, p, which has been estimated for a wide range of physical systems. Specifically, it can range from 10^{-3} for electron charge states in GaAs semiconductors, to 10^{-9} for photons, 10^{-13} for trapped ions, and 10^{-14} for nuclear spins.[7]

How realistic is quantum computation as a technology? We cannot achieve these physical limits with current technologies, but researchers have proposed concepts for realizing scalable quantum computers, and initial experiments are promising. Nuclear spins manipulated by nuclear magnetic resonance (NMR) techniques have demonstrated Shor's algorithm with seven qubits.[10] In these systems, single-qubit operations take place at about 1 MHz, and two-qubit gates at about 1 kHz, with an error probability $p \approx 10^{-3}$. It is believed that $p \approx 10^{-6}$ will ultimately be possible for this kind of device.

Lower error rates are expected to apply for NMR systems that use other techniques, such as artificial molecules synthesized from solid-state quantum dots[11] or carefully placed phosphorus impurities in silicon.[3] Faster clock speeds of around 1 GHz should also be possible. For scalability and to take advantage of a tremendous historical investment in silicon fabrication, our architecture assumes a solid-state technology such as quantum dots or phosphorus atoms. We want to use these technologies to provide the building blocks for reliable quantum computation, much as von Neumann did for classical computation.[12]

We're a long way from system-scale maturity in today's quantum logic gates, but it was also a long way from the initial silicon transistors to modern VLSI. We propose stepping in that direction.

PROGRAMMING MODEL

Given that a technology solution is possible, how would we implement a quantum algorithm?

Although some early work was done on quantum Turing machines,[13] the quantum computation community has focused almost entirely on a circuit model[14] in which algorithms and architecture are tightly integrated—similar to a classical application-specific integrated circuit, or ASIC. In contrast, our goal is to design a *general-purpose* piece of hardware that we can program to perform arbitrary quantum computations.

We can express quantum algorithms through a model that performs quantum operations on quantum data under the control of a classical computer. Accordingly, quantum programs would combine quantum unitary transforms (quantum gates), quantum measurements, classical computation, and classical control-flow decisions into a single instruction stream. A compiler (such as QCL[15]) then reads a mixed quantum/classical language and breaks down complex quantum operations into a small set of universal operators. The compiler encodes these operators into a classical bit instruction stream that also includes conventional processor instructions.

We anticipate that this compiler will have two main parts: a static precompiler and a dynamic compiler. Both parts are cross-compilers, running on a conventional microprocessor and producing code for our quantum architecture.

The precompiler would generate code that produces a computation with a targeted end-to-end error probability on an ideal quantum computer. This end-to-end error means that the generated code must check the answer and restart if it is wrong. Similar to conventional VLSI synthesis tools, the compiler employs a technology model, but only to the extent that it specifies a universal set of primitive operations. The compiler does not need any knowledge of error models.

The dynamic compiler accepts the precompiled binary code and produces an instruction stream to implement a fault-tolerant computation, using the minimal quantum error correction necessary to meet the end-to-end error rate. This compiler is also given the technology model and, importantly, a bound on program execution time. Errors occur so infrequently in classical architectures that program run length is rarely an issue. In quantum architectures, however, errors are frequent, and correction incurs a polylogarithmic cost in run length. Our work on this architecture indicates that exploiting program run length is key to performance.

The bound on program run length can originate in either a user *hint* or dynamic profiling. The hint expresses the algorithm's running time given some input data size. To date, such information is available for all known quantum algorithms. If the hint is not available, the compiler uses an adjustable policy to optimize programs adaptively. An aggressive policy would start with minimal error correction and increase reliability until the program produces the right answer; a conservative policy would start with extremely reliable correction and decrease reliability for future runs.

In quantum architectures, errors are frequent, and correction incurs a polylogarithmic cost in run length.

QUANTUM ERROR CORRECTION

The nonlocalized properties of quantum states means that localized errors on a few qubits can have a global impact on the exponentially large state space of many qubits. This makes quantum error correction perhaps the single most important concept in devising a quantum architecture. Unlike classical systems, which can perform brute-force, signal-level restoration error correction in every transistor, quantum state error correction requires a subtle, complex strategy.

Quantum difficulties

The difficulty of error-correcting quantum states has two sources.

First, errors in quantum computations are distinctly different from errors in classical computing. Despite the digital abstraction of qubits as two-level quantum systems, qubit state probability amplitudes are parameterized by continuous degrees of freedom that the abstraction does not automatically protect. Thus, errors can be continuous in nature, and minor shifts in the superposition of a qubit cannot be discriminated from the desired computation. In contrast, classical bits suffer only digital errors. Likewise, where classical bits suffer only bit-flip errors, qubits suffer both bit-flip and phase-flip errors, since their amplitude signs can be either negative or positive.

The second source of difficulty is that we must correct quantum states without measuring them because measurement collapses the very superpositions we want to preserve.

Error-correction code

Quantum error-correction codes successfully address these problems by using two classical codes simultaneously to protect against both bit and phase errors, while allowing measurements to determine only information about the error that occurred and nothing about the encoded data. An $[n, k]$ code uses n qubits to encode k qubits of data. The encoding

Table 1. Recursive error-correction overhead for a single-qubit operation using [7,1] Steane correction code.

Recursion level (k)	Storage overhead 7^k	Operation overhead 153^k	Minimum time overhead 5^k
0	1	1	1
1	7	153	5
2	49	23,409	25
3	343	3,581,577	125
4	2,401	547,981,281	625
5	16,807	83,841,135,993	3,125

circuit takes the k data qubits as input, together with $n - k$ *ancilla* qubits. Ancilla bits are extra “scratch” qubits that quantum operations often use; a specialized, entropy exchange unit produces the ancilla bits and “cools” them to an initial state $|0\rangle$. The decoder takes in an encoded n-qubit state and outputs k (possibly erroneous) qubits together with $n - k$ qubits that, with high probability, specify which error occurred. A recovery circuit then performs one of 2^{n-k} operations to correct the error on the data.

This model assumes that qubit errors are independent and identically distributed. Classical error correction makes the same assumption, and we can adapt classical strategies for handling deviations to the quantum model.

Quantum error correction has a powerful and subtle effect. Without it, the “correctness”—technically, the *fidelity*—of a physical qubit decays exponentially and continuously with time. With it, the exponential error model becomes linear: A logical qubit encoded in a quantum error-correcting code and undergoing periodic error measurement suffers only linear discrete amounts of error, to first order.

Not all available codes are suitable for fault-tolerant computation, but the largest class—the *stabilizer codes*—support computation *without* decoding the data and thus propagating more errors in the process. We chose the [7,1] Steane stabilizer code for our architecture. It uses seven physical qubits to encode one logical qubit and is nearly optimal (the smallest perfect quantum code is [5,1][16]). The code can perform an important set of single-qubit operations as well as the two-qubit controlled-NOT operator (used in the architecture’s quantum ALU) on the encoded qubit simply by applying the operations to each individual physical qubit.

Error-correction costs

The cost of error correction is the overhead needed to compute encoded states and to perform periodic error-correction steps. Each such step is a *fault-tolerant operation*. The Steane code requires approximately 153 physical gates to construct a fault-tolerant single-qubit operation.

Despite this substantial cost, the 7-qubit error-correcting code dramatically improves the quantum computing situation. The probability of a logical qubit error occurring during a single operation changes from p to cp^2, where c is a constant determined by the number of places two or more failures can occur and propagate to the next logical qubit, and we want $cp^2 < p$.

For a single logical gate application, c is about 17,446. For a physical qubit transform failure rate of $p = 10^{-6}$, this means the 7-qubit Steane code has a probable logical qubit transform failure rate of 1.6×10^{-7} when a maximally parallelized operation uses an optimized error measurement procedure.[16] Producing systems with a lower c and more reasonable overheads requires a failure rate that is closer to 10^{-9}.

Recursive error correction

The most important application of quantum codes to computation is a recursive construction,[5] which exponentially decreases error probabilities with only polynomial effort. This is crucial because even an error probability of cp^2 is too high for most quantum applications.

The following example helps to understand the construction: The Steane code transforms the physical qubit error rate p to a logical qubit error rate cp^2 but requires some number of physical qubit gates per logical qubit gate operation. Suppose, however, that a logical gate on a 7-qubit code again implemented each of those physical gates. Each gate would have a logical gate accuracy of cp^2, and the overall logical gate error rate would become $c(cp^2)^2$. For a technology with $p = 10^{-6}$, the error rate for each upper level gate would be roughly 4.3×10^{-10}. The key observation is that as long as $cp^2 < p$, error probabilities decrease exponentially with only a polynomial increase in overhead.

Table 1 summarizes the costs of recursive error

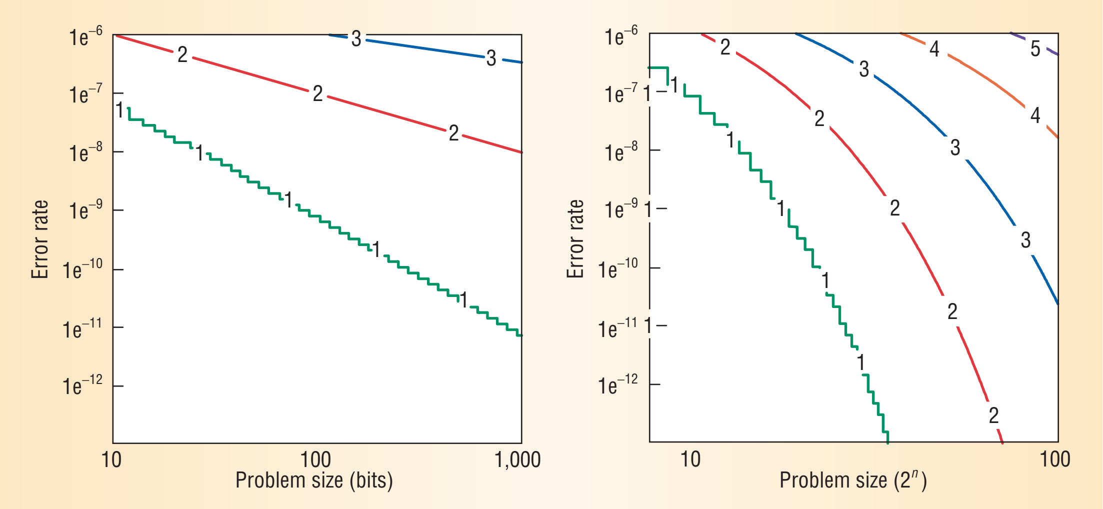


Figure 1. Recursion level k with a varied problem size and underlying qubit error probability for Shor's quantum factorization algorithm (left) and Grover's quantum search algorithm (right).

correction up to five levels for storage, operation, and minimum time overheads. Clearly, the high cost of recursive error correction means that a quantum computer architecture should choose the minimum recursion level for a given algorithm and data size.

Figure 1 depicts recursion level *k* with a varied problem size and underlying qubit error probability for both Shor's and Grover's algorithms. Increases in problem size or error probability require stronger error correction through additional levels of recursion.

QUANTUM COMPUTER ARCHITECTURE

Building upon the theory of fault-tolerant quantum computation, we define the building blocks for a general architecture that can dynamically minimize error-correction overhead. In contrast to the circuit model used in much of the quantum computing literature, our architecture can efficiently support different algorithms and data sizes. The key mechanisms enabling this generalization are reliable data paths and efficient quantum memory.

In many respects, quantum computation is similar to classical computation. For example, quantum algorithms have a well-defined control flow that manipulates individual data items throughout the execution. The physical restrictions on quantum technologies also resemble the classical domain. Even though two qubits can interact at a distance, the strongest—and least error-prone—interaction is between near neighbors. Furthermore, controlled interaction requires classical support circuitry, which must be routed appropriately throughout the device.

Although our quantum computer architecture is similar to a classical architecture, certain aspects of the computation are unique to the quantum domain. As Figure 2 shows, the overall architecture has three major components: the quantum arithmetic logic unit (ALU), quantum memory, and a dynamic scheduler. In addition, the architecture uses a novel quantum wiring technique that exploits quantum teleportation.[17]

Quantum ALU

At the core of our architecture is the quantum ALU, which performs quantum operations for both computation and error correction. To efficiently perform any specified quantum gates on the quantum data, the ALU applies a sequence of basic quantum transforms under classical control.[7] The transforms include

- the Hadamard (a radix-2, 1-qubit Fourier transform),
- identity (I, a quantum NOP),
- bit flip (X, a quantum NOT),
- phase flip (Z, which changes the signs of amplitudes),
- bit and phase flip (Y),
- rotation by π/4 (S),
- rotation by π/8 (T), and
- controlled NOT (CNOT).

These gates form one of the smallest possible universal sets for quantum computation. The underlying physical quantum technology can implement these gates efficiently on encoded data. All except CNOT operate on only a single qubit; the CNOT gate operates on two qubits.

To perform the high-level task of error correction, the ALU applies a sequence of elementary

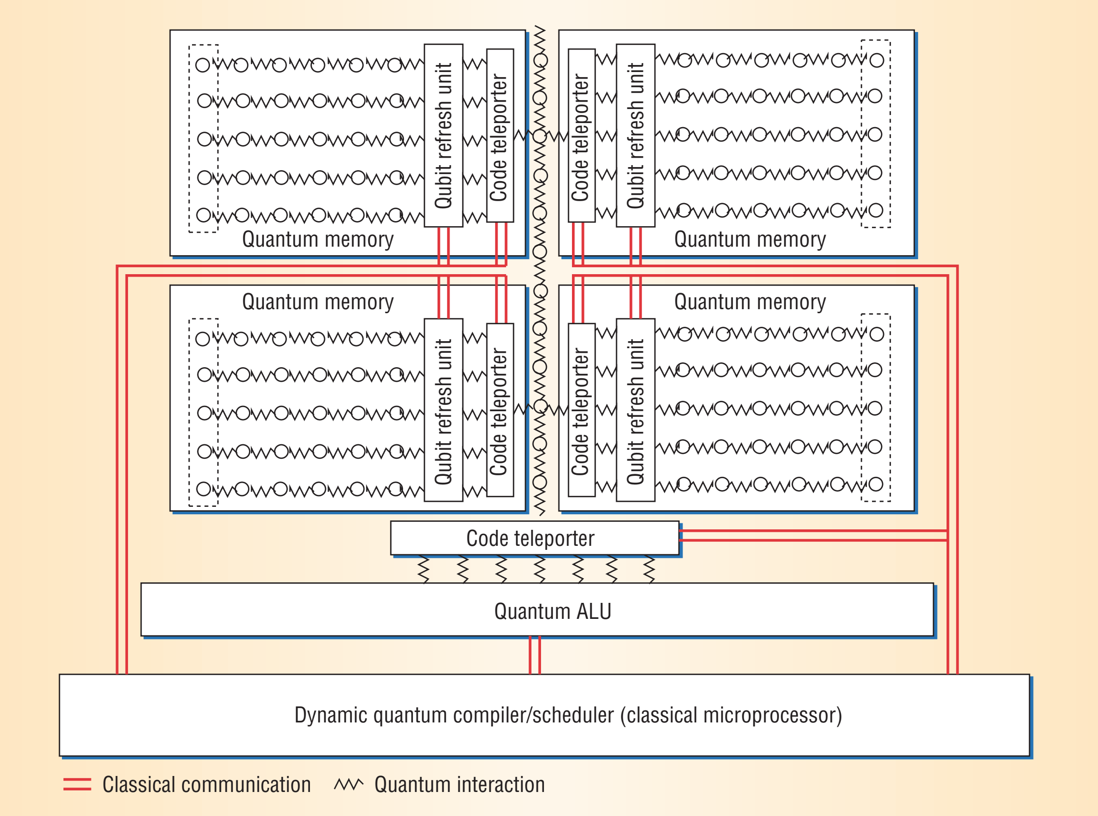


Figure 2. Fault-tolerant quantum computer architecture. The quantum arithmetic logic unit (ALU) performs all quantum operations, quantum memory banks support efficient code conversion, teleportation transmits quantum states without sending quantum data, and the dynamic scheduler controls all processes.

operations. Because this task is requisite to fault-tolerant quantum computing, the ALU performs it on encoded data after most logical operations. This procedure consumes ancilla states, which help in the computation of parity checks. Specialized hardware provides elementary standard states that the ALU uses to manufacture requisite ancilla.

Quantum memory

The architecture's generality relies on an efficient quantum memory. The key is building quantum memory banks that are more reliable than quantum computation devices. We can also use specialized "refresh" units that are much less complex than our general ALU.

The storage of qubits not undergoing computation is very similar to the storage of conventional dynamic RAM. Just as individual capacitors used for DRAM leak into the surrounding substrate over time, qubits couple to the surrounding environment and decohere over time. This requires periodically refreshing individual logical qubits. As Figure 2 shows, each qubit memory bank has a dedicated refresh unit that periodically performs error detection and recovery on the logical qubits. From a technological standpoint, decoherence-free subsystems,[18] which naturally provide lower decoherence rates for static qubits, could implement such quantum memories.

The architecture uses multiple quantum memory banks. This is not for improving logical qubit access times. In fact, the underlying error rate of the qubit's physical storage mechanism, the algorithm's complexity and input data size, the quantum ALU's operation time and parallelism, and the error-correction code that stores the logical qubits limit the bank size. For example, if we run Shor's algorithm on a 1,024-bit number using a memory technology with an error rate of $p = 10^{-9}$, we estimate that it would use 28,000 physical qubits to represent about 1,000 physical bits using two levels of recursion in a 5-qubit error-correction code. On the other hand, if the error rate increases to $p = 10^{-6}$, error correction would require four levels of recursion to refresh a bank size of just 1,000 physical qubits that would store only two logical qubits.

Quantum wires

Moving information around in a quantum computer is a challenge. Quantum operations must be reversible, and we cannot perfectly clone qubits—that is, we cannot copy their value. We cannot simply place a qubit on a wire and expect it to transmit the qubit's state accordingly. Instead, our architecture will use a purely quantum concept to implement quantum wires: teleportation.[17] This procedure, which has been experimentally demonstrated,[19] transmits a quantum state between two points without actually sending any quantum data. Instead, with the aid of a certain standard preshared state, teleportation sends two classical bits of data for each qubit.

Teleportation is superior to other means of delivering quantum states. Recall that a solid-state technology implements qubits with atoms implanted in silicon.[3,11] The physical qubits cannot move, but we can apply a swap operation to progressive pairs of atoms to move the qubit values along a line of atoms. While we could use a series of quantum swap gates to implement quantum wires, each swap gate is composed of three CNOT gates, which introduces errors in the physical qubits—errors that generate additional overhead in the correction procedures.

Teleportation instead uses quantum swap gates that are not error-corrected to distribute qubits in a *cat* state to the source and destination of the wire. A cat state (named after Schrödinger's cat) is a qubit vector with probabilities equally distributed between all bits set to 1 and all bits set to 0. The qubits in a cat state are entangled, and measuring one of the qubits uniquely determines the state of all qubits in the qubit vector. Teleportation uses a two-qubit cat state.

This cat state can be checked for errors easily and independently of the physical qubit being transmitted. If errors have overwhelmed the cat state, it can be discarded with little harm to the transmission process. Once a correct cat state exists at both ends, the cat state's qubits teleport the physical qubit across the required distance.

Code teleportation

Teleportation can also provide a general mechanism for simultaneously performing quantum operations while transporting quantum data. Precomputing the desired operation on the cat states forms a kind of "quantum software" that automatically performs its operation on the teleported data.[20] We can use this mechanism to perform an optimization by converting between different error-correction codes during teleportation. Specifically, we chose the Steane error-correction code for its computational ease, not its compactness. The quantum memories, however, perform only error measurement and recovery, not computation. Hence, they can use a more compact code that sacrifices some ease of computation.

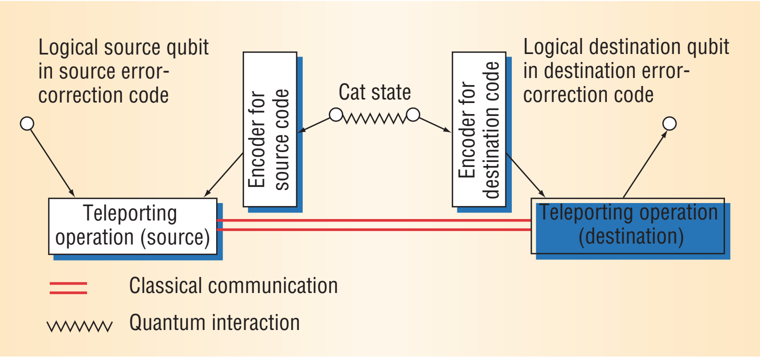

Logical source qubit in source error-correction code
Encoder for source code
Cat state
Encoder for destination code
Logical destination qubit in destination error-correction code
Teleporting operation (source)
Teleporting operation (destination)
Classical communication
Quantum interaction

Figure 3. Code teleportation. In a slight modification of standard quantum teleportation, an encoder at the destination recreates quantum data encoded in another form at the sender.

Converting between codes is usually an error-prone process, but teleportation performs code conversion without a single physical qubit error compromising a complete logical qubit state.[20] Thus, our architecture can store the logical qubits efficiently in a dense error-correcting code if it uses teleportation during transmission to the quantum ALU for conversion to a less compact, but more easily computable, error-correction code.

From a conceptual standpoint, this process is only a slight modification of standard quantum teleportation. As Figure 3 shows, specialized hardware generates a cat state, sends one qubit through the encoding mechanism for the source error-correction code, and sends the other qubit through the encoder for the destination error-correction code. The sender and receiver then perform the logical qubit equivalents of the teleportation operation on each end of the entangled pair.

To implement a more robust form of this process, the underlying architecture could use stabilizer measurements to generate the appropriately encoded cat states prior to teleportation.

Dynamic scheduler

The architecture uses a complete high-performance classical processor for control. This processor runs a dynamic scheduling algorithm that takes in logical quantum operations, interleaved with classical control-flow constructs, and dynamically translates them into physical individual qubit operations. The algorithm uses knowledge about the overall input data size and physical qubit error rates to construct a dynamic schedule to control the quantum ALU, code teleportation, and qubit RAM refresh units. This is a lot of work for a single classical processor. We expect significantly faster processor clock speeds to be available, but it may be necessary to run multiple classical processors in parallel.

The classical processor is critical to making a

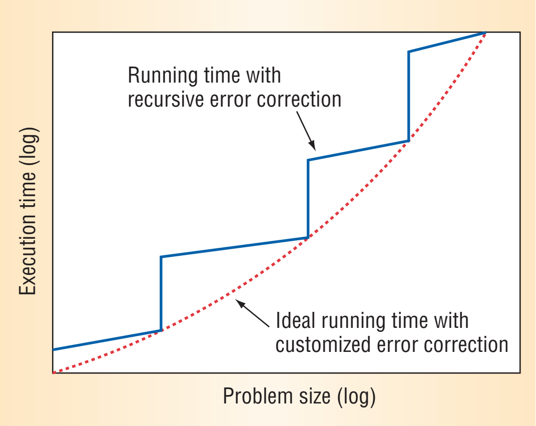


Figure 4. Quantum computing performance with recursive error correction. The recursive approach to stronger error corrections results in a stairstep curve.

quantum architecture efficient. We could execute all quantum algorithms with the maximum available error correction, but doing so would be incredibly inefficient. Moreover, using dynamic compilation and knowledge of an algorithm's execution time make several performance optimizations available to the computation, including application-specific clustering prior to error measurement.

APPLICATION-SPECIFIC ERROR OPTIMIZATION

While theoretically possible, quantum error correction introduces overheads yet unheard of in the classical domain. A single level of error correction incurs an overhead of at least 153 quantum gates per logical operation in our architecture; a k level recursive scheme has a factor of 153^k overhead.

The scheduling unit ultimately implements mechanisms to control this overhead dynamically at execution time. This unit compiles the quantum software instructions (that operate on logical qubits) into the specific quantum operations required for execution on the physical qubits of the error-correction codes used throughout the architecture. Furthermore, the unit dynamically schedules the quantum operations to intermix classical control-flow constructs with the quantum operations, while fully utilizing the available quantum ALU functional units.

Figure 4 abstractly depicts the effects of recursive error correction on execution time. As application data size increases, so must the recursive structure, but the recursion increases occur at integral steps. Using the classical processor for just-in-time quantum software compilation, we customize the error correction to the algorithm and data size. This customization aggregates the cost of error-correction processes over several operations, thereby making the integral cost more continuous.

Our architecture achieves system-level efficiencies through code teleportation, quantum memory refresh units, dynamic compilation of quantum programs, and scalable error correction. Our work indicates that reliability of the underlying technology is crucial; practical architectures will require quantum technologies with error rates between 10^{-6} and 10^{-9}.

In addition to the underlying technology, the significant overhead of quantum error correction remains the most pressing quantum computing architectural issue. The clustering solution we propose can regain some of the performance lost from recursive error correction, but the gains are limited to the cost of only a single recursion layer. Further reductions will require other new techniques. Quantum theorists are working on new correction codes with attractive properties. Some can correct for more than a single error or condense more than one logical qubit together to increase density.

The key to exploiting these algorithmic developments in a quantum architecture is to identify the basic building blocks from which a design methodology can grow. We hope to lay the foundation for a science of quantum CAD for the reliable quantum computers of the future. ■

Acknowledgments

We conducted this work as part of the Quantum Architecture Research Center (http://feynman.media.mit.edu/quanta/qarc/), funded by the DARPA Quantum Information Science and Technology program. John Kubiatowicz contributed to the initial discussions of this work. John Black provided comments on the state of the art in breaking cryptosystems. Thanks also to Matt Farrens and Gary Tyson.

References

1. L.M. Vandersypen et al., "Experimental Realization of Order-Finding with a Quantum Computer," *Physical Rev. Letters*, vol. 15, 15 Dec. 2000, pp. 5452-5455.
2. E. Knill et al., "An Algorithmic Benchmark for Quantum Information Processing," *Nature*, vol. 404, 2000, pp. 368-370.
3. B. Kane, "A Silicon-Based Nuclear Spin Quantum Computer," *Nature*, vol. 393, 1998, pp. 133-137.
4. J.E. Mooij et al., "Josephson Persistent-Current Qubit," *Science*, vol. 285, 1999, pp. 1036-1039.
5. D. Aharonov and M. Ben-Or, "Fault-Tolerant Computation with Constant Error," *Proc. 29th Ann. ACM Symp. Theory of Computing*, ACM Press, New York, 1997, pp. 176-188.

6. K.M. Obenland, "Using Simulation to Assess the Feasibility of Quantum Computing," doctoral dissertation, Univ. of Southern California, Los Angeles, 1999.
7. M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2000.
8. P. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," *Proc. 35th Ann. Symp. Foundations of Computer Science*, IEEE CS Press, Los Alamitos, Calif., 1994, p. 124.
9. L. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," *Proc. 28th Ann. ACM Symp. Theory of Computation*, ACM Press, New York, 1996, pp. 212-219.
10. L.M. Vandersypen et al., "Experimental Realization of Shor's Quantum Factoring Algorithm Using Nuclear Magnetic Resonance," *Nature*, vol. 414, 2001, p. 883.
11. D.P. DiVincenzo and D. Loss, "Quantum Information Is Physical," *Superlattices and Microstructures*, vol. 23, 1998, p. 419.
12. J. von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," in *Automata Studies*, Princeton University Press, Princeton, N.J., 1956, pp. 329-378.
13. A.C. Yao, "Quantum Circuit Complexity," *Proc. 34th Ann. IEEE Symp. Foundations of Computer Science*, IEEE CS Press, Los Alamitos, Calif., 1993, pp. 352-361.
14. A. Barenco et al., "Elementary Gates for Quantum Computation," *Physical Rev. A*, vol. 52, 1995, pp. 3457-3467.
15. B. Ömer, "Quantum Programming in QCL," master's thesis, Technical University of Vienna, 2000.
16. A. Steane, "Active Stabilisation, Quantum Computation and Quantum State Synthesis," *Physical Rev. Letters*, vol. 78, 1997, p. 2252.
17. C.H. Bennett et al., "Teleporting an Unknown Quantum State via Dual Classical and EPR Channels," *Physical Rev. Letters*, vol. 70, 1993, pp. 1895-1899.
18. D.A. Lidar, I.L. Chuang, and K.B. Whaley, "Decoherence-free Subspaces for Quantum Computation," *Physical Rev. Letters*, vol. 81, no. 12, 1998, pp. 2594-2597.
19. D. Bouwmeester et al., "Experimental Quantum Teleportation," *Nature*, vol. 390, 1997, pp. 575-579.
20. D. Gottesman and I.L. Chuang, "Quantum Teleportation Is a Universal Computational Primitive," *Nature*, vol. 402, 1999, pp. 390-392.

***Mark Oskin** is an assistant professor in the Department of Computer Science and Engineering at the University of Washington. He received a PhD in computer science from the University of California, Davis. His research focuses on reconfigurable systems, computational substrates, and automated design tools for quantum computing architectures. Contact him at oskin@cs.washington.edu.*

***Frederic T. Chong** is an associate professor in the Department of Computer Science at the University of California, Davis. He received a PhD in electrical engineering and computer science from the Massachusetts Institute of Technology. His research focuses on computer architectures for novel technologies. Contact him at chong@cs.ucdavis.edu.*

***Isaac L. Chuang** is an associate professor at the Massachusetts Institute of Technology, where he leads the quanta research group at the MIT Media Laboratory. He received a PhD in electrical engineering from Stanford University, where he was a Hertz Foundation Fellow. His research focuses on quantum information science, the physics of computation, information theory, and implementations of quantum computers and cryptosystems. Contact him at ike@media.mit.edu.*

A.2 Computer’06 Paper

COVER FEATURE

A Layered Software Architecture for Quantum Computing Design Tools

Krysta M. Svore, Alfred V. Aho
Columbia University

Andrew W. Cross, Isaac Chuang
Massachusetts Institute of Technology

Igor L. Markov
University of Michigan

Compilers and computer-aided design tools are essential for fine-grained control of nanoscale quantum-mechanical systems. A proposed four-phase design flow assists with computations by transforming a quantum algorithm from a high-level language program into precisely scheduled physical actions.

Quantum computers have the potential to solve certain computational problems—for example, factoring composite numbers or comparing an unknown image against a large database—more efficiently than modern computers. They are also useful in controlling quantum-mechanical systems in emergent nanotechnology applications, such as secure optical communication, in which modern computers cannot natively operate on quantum data.

Despite convincing laboratory demonstrations of quantum information processing, as the "Ongoing Research in Quantum Computing" sidebar describes, it remains difficult to scale because it relies on inherently noisy components. Adequate use of quantum error correction and fault tolerance theoretically should enable much better scaling, but the sheer complexity of the techniques involved limits what is doable today. Large quantum computations must also achieve a high degree of parallelism to complete before quantum states decohere.

As candidate quantum technologies mature, the feasibility of quantum computation will increasingly depend on software tools, especially compilers, that translate quantum algorithms into low-level, technology-specific instructions and circuits with added fault tolerance and sufficient parallelism.

We propose a layered software architecture consisting of a four-phase computer-aided design flow that assists with such computations by mapping a high-level language source program representing a quantum algorithm onto a quantum device. By weighing different optimization and error-correction procedures at appropriate phases of the design flow, researchers, algorithm designers, and tool builders can trade off performance and accuracy.

QUANTUM COMPUTATION

The *quantum circuit*,[1] a commonly used computation model similar to a modern digital circuit, provides a representation of a quantum algorithm. Digital circuits capture both mathematical algorithms, such as for sorting and searching, and methods for real-world control and measurement, as in cellular phones and automobiles. Quantum circuits likewise describe methods for control of quantum systems, such as atomic clocks and optical communication links, that cannot be fully controlled with conventional binary digital circuits alone.

A quantum circuit consists of quantum bits (qubits), quantum gates, quantum wires, and qubit measurements. A *qubit* is analogous to a classical bit but can be in a wave-like *superposition* of the symbolic bit values 0 and 1, written $a|0\rangle + b|1\rangle$, where a and b are complex numbers. Mathematically, a qubit can be written as a vector of complex numbers. When measured, a qubit

Published by the IEEE Computer Society

0018-9162/06/$20.00 © 2006 IEEE

Ongoing Research in Quantum Computing

Researchers in industry and government labs are exploring various aspects of quantum design and automation with a wide range of applications. In addition to the examples described below, universities in the US, Canada, Europe, Japan, and China are carrying out much broader efforts.

BBN Technologies

Based in Cambridge, Massachusetts, BBN Technologies (www.bbn.com) developed the world's first quantum key distribution (QKD) network with funding from the US Defense Advanced Research Projects Agency. The fiber-optical DARPA Quantum Network offers 24x7 quantum cryptography to secure standard Internet traffic such as Web browsing, e-commerce, and streaming video.

D-Wave Systems

Located in Vancouver, British Columbia, Canada, D-Wave Systems (www.dwavesys.com) builds superconductor-based software-programmable custom integrated circuits for quantum optimization algorithms and quantum-physical simulations. These ICs form the heart of a quantum computing system designed to deliver massively more powerful and faster performance for cryptanalysis, logistics, bioinformatics, and other applications.

Hewlett-Packard

The Quantum Science Research Group at HP Labs in Palo Alto, California, is exploring nanoscale quantum optics for information-processing applications (www.hpl.hp.com/research/qsr). In addition, the Quantum Information Processing Group at the company's research facility in Bristol, UK, is studying quantum computation, cryptography, and teleportation and communication (www.hpl.hp.com/research/qip).

Hypres

Located in Elmsford, New York, Hypres Inc. (www.hypres.com) is the leading developer of superconducting digital circuits for wireless and optical communication. Based on rapid single-flux quantum logic, these circuits have achieved gate speeds up to 770 GHz in the laboratory.

IBM Research

Scientists at IBM's Almaden Research Center in California and the T.J. Watson Research Center's Yorktown office in New York developed a nuclear magnetic resonance (NMR) quantum computer that factored 15 into 3 ×5 (http://archives.cnn.com/2000/TECH/computing/08/15/quantum.reut). Researchers at the Watson facility and the Zurich Research Lab are also developing Josephson junction quantum devices (www.research.ibm.com/ss_computing) as well as studying quantum information theory (www.research.ibm.com/quantuminfo).

Id Quantique

Based in Geneva, Switzerland, id Quantique (www.idquantique.com) is a leading provider of quantum cryptography solutions, including wire-speed link encryptors, QKD appliances, a turnkey service for securing communication transfers, and quantum random number generators. The company's optical instrumentation product portfolio includes single-photon counters and short-pulse laser sources.

Los Alamos National Lab

The Los Alamos National Lab (http://qso.lanl.gov/qc) in New Mexico is studying quantum-optical long-distance secure communications and QKD for satellite communications. It has also conducted groundbreaking work on quantum error correction, decoherence, quantum teleportation, and the adaptation of NMR technology to quantum information processing.

MagiQ Technologies

MagiQ Technologies (www.magiqtech.com), headquartered in New York City, launched the world's first commercial quantum cryptography device in 2003. MagiQ Quantum Private Network systems incorporate QKD over metro-area fiber-optic links to protect against both cryptographic deciphering and industrial espionage.

NEC Labs

Scientists at NEC's Fundamental and Environmental Research Laboratories in Japan, in collaboration with the Riken Institute of Physical and Chemical Research, have demonstrated a basic quantum circuit in a solid-state quantum device (www.labs.nec.co.jp/Eng/innovative/E3/top.html). Recently, NEC researchers have also been involved in realizing the fastest fortnight-long, continuous quantum cryptography final-key generation.

NIST

The Quantum Information Program at the US National Institute of Standards and Technology (http://qubit.nist.gov) is building a prototype 10-qubit quantum processor as a proof-in-principle of quantum information processing. Potential applications include ultraprecise measurement (atomic clocks, optical metrology, and so on), control of dynamic processes, and nanotechnology. Researchers at the program's facilities in Boulder, Colorado, and Gaithersburg, Maryland, are also optimizing the speed of free-space quantum cryptography systems.

NTT Basic Research Labs

NTT's Superconducting Quantum Physics Research Group in Japan focuses on the development of quantum cryptography protocols (www.brl.ntt.co.jp/group/shitsuryo-g/qc). In particular, they have exhibited quantum cryptography using a single photon realized in a photonic network of optical fibers.

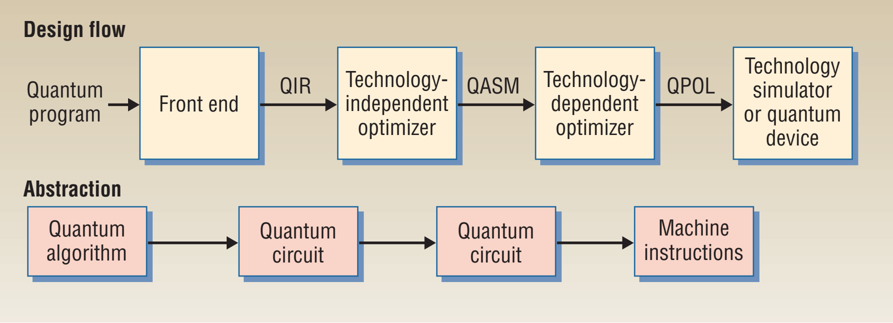


Figure 1. Proposed design flow. The first three phases are part of the quantum computer compiler, while the last phase implements the quantum algorithm on a quantum device or simulator.

assumes either the value 0 or the value 1, with probability $|a|^2$ and $|b|^2$, respectively.

An n-qubit quantum state is written as a vector representing a superposition of 2^n different bit strings. The state remains in a superposition for the computation's duration, and the final sequence of measurements collapses the state onto the bit string that gives the result of the computation. This result will not be affected if all bit strings in a given state are multiplied by a constant, called a *global phase*, before measurement. However, the ratios of coefficients of different bit strings are significant and determine *relative phases*.

A *quantum gate* is a reversible transformation of a quantum state that preserves total probability—for example, for a single qubit $|a|^2 + |b|^2 = 1$. Quantum gates are represented by unitary matrices that act on quantum state vectors by left multiplication. Gates are connected by *quantum wires* that transport qubits forward in time or space. Quantum wires cannot fan out—that is, qubits with unknown state cannot be duplicated. Matrix multiplication models composition of gates in series; the Kronecker, or tensor, product models composition of gates in parallel.

Inaccurate gates and uncontrolled environmental couplings introduce data errors. Uncontrolled coupling results in *decoherence,* which causes qubits to collapse to states that behave probabilistically, like (possibly biased) classical coins. Such states have no phase information and cannot perform quantum computation. These effects complicate quantum information processing, but researchers can address them using tools that perform optimizations and automatically add error correction.

FOUR-PHASE DESIGN FLOW

We envision a hierarchy of design tools with simple interfaces between layers that include programming languages, compilers, optimizers, simulators, and layout tools. Such an architecture appears necessary because no single entity can afford the huge investments required to develop all necessary tools. To this end, open source software encourages wider community participation.

A sufficiently transparent architecture facilitates tool interoperability, focused point-tool development, and incremental improvements. Quantum algorithm designers and those developing quantum circuit optimizations can explore new algorithms and error-correction procedures in more realistic settings involving actual noise and physical resource constraints. Researchers can also simulate important quantum algorithms on proposed new technologies before doing expensive lab experiments.

Our four-phase design flow, shown in Figure 1, maps a high-level program representing a quantum algorithm into a low-level set of machine instructions to be implemented on a physical device. The high-level quantum programming language encapsulates the mathematical abstractions of quantum mechanics and linear algebra.[1] The design flow's first three phases are part of the quantum computer compiler (QCC). The last phase implements the algorithm on a quantum device or simulator.

In addition to providing support for the abstractions used to specify quantum algorithms, the programming languages and compilers at the top level of our tool suite accommodate optimization improvements as our understanding of new quantum technologies matures. The simulation and layout tools at the bottom level incorporate details of the emerging quantum technologies that would ultimately implement the algorithms described in the high-level language. The tools balance tradeoffs involving performance, qubit minimization, and fault-tolerant implementations.

The representations of the quantum algorithm between the phases are the key to an interoperable tools hierarchy. In the first phase, the compiler front end maps a high-level specification of a quantum algorithm into a *quantum intermediate representation* (QIR)—a quantum circuit with gates drawn from some universal set. Compared to traditional logic circuits, quantum circuits are more structured and typically have intrinsic sequential semantics, wherein gates modify globally maintained state qubits in parallel.

In the second phase, a technology-independent optimizer maps the QIR into an equivalent lower-level circuit representation of single-qubit and controlled-NOT (CNOT) gates. The compiler optimizes this *Quantum Assembly Language* (QASM) according to a cost function such as circuit size, circuit depth, or accuracy. Since limiting quantum computing to a fixed set of registers and fixed word size would significantly restrict its power, QASM does not have such limitations, unlike traditional assembly languages. Therefore, parallelism has a greater impact and must be extracted by the compiler.

The third phase consists of optimizations suited to the quantum computing technology and outputs *Quantum Physical Operations Language* (QPOL), a physical-language representation with technology-specific parameters. QPOL includes two subphases: The first maps the representation of single-qubit and CNOT gates into a QASM representation using a fault-tolerant discrete universal set of gates; the second maps these gates into a QPOL representation containing the physical instructions for the fault-tolerant operations scheduled in parallel, including the required movements of physical particles. Knowledge of the physical layout and architectural limitations enters no later than at this step.

The quantum programming environment should allow easy separation of classical and quantum computations.

The final phase utilizes technology-dependent tools such as layout modules, circuit and physical simulators, or interfaces to actual quantum devices. If at this point certain technology constraints or objectives have not been met, algorithm and device designers can repeat some earlier phases. In addition, it is possible to add fault tolerance and error correction at multiple phases of the design process.

The "Sample Design Flow: EPR Pair Creation" sidebar provides a concrete example of how our proposed design flow automates the process of transforming mathematical models into software for controlling a live quantum-mechanical system.

PROGRAMMING ENVIRONMENT AND LANGUAGE

Designing a quantum programming environment is difficult given the currently limited repertoire of quantum algorithms. However, this situation is likely to improve as the demand for nanoscale control increases. The programming model is also uncertain because researchers can design a quantum computer as either an application-specific integrated circuit or a general-purpose processor. However, it is safe to assume that classical computers will monitor quantum devices through a bidirectional communication link.[2]

A quantum programming environment should possess several key characteristics.[2] First, it needs a high-level quantum programming language that offers the necessary abstractions to perform useful quantum operations. It should support complex numbers, quantum unitary transforms (quantum gates), and measurements as well as classical pre- and postprocessing. Support for reusable subroutines and gate libraries is also required. However, the exact modularization of a quantum programming environment remains an open question.

In addition, the environment as well as the programming language should be based on familiar concepts and constructs. This would make learning how to write, debug, and run a quantum program easier than using a totally new environment.

The quantum programming environment also should allow easy separation of classical and quantum computations. Because a quantum computer has noise and limited coherence time, this separation can limit computation time on the quantum device. The compiler for a quantum programming language should be able to translate a source program into an efficient and robust quantum circuit or physical implementation; it should be easy to translate into different gate sets or optimize with respect to a desired cost function.

Further, the high-level programming language should be hardware-independent and compile onto different quantum technologies. However, the language and environment should allow the inclusion of technology-specific modules.

A language that supports high-level abstractions would facilitate development of new quantum algorithms and applications. Researchers have proposed many quantum programming languages based on the quantum circuit model,[2,3] but a language that provides further insights on quantum information processing is needed. We also seek a language that simplifies creation of robust, optimized target programs.

QUANTUM COMPUTER COMPILER

A generic compiler for a classical language on a classical machine consists of a sequence of phases that transform the source program from one representation into another.[4] This partitioning of the compilation process has led to the development of efficient algorithms and tools for each phase. Because the front-end processes for QCCs are similar to those of classical compilers, researchers can use the algorithms and tools to build lexical, syntactic, and semantic analyzers for QCCs. However, the intermediate representations, the optimization phase, and the code-generation phase of QCCs differ greatly from classical compilers and require novel approaches, such as a way to insert error-correction operations into the target language program.

Quantum intermediate representation

Other popular quantum computation models, such as adiabatic quantum computing, can be converted to quantum circuits. Therefore, in our design flow's first phase, the QCC's front end maps a high-level specification of a quantum algorithm into a QIR based on the quantum circuit model.[1]

Provisions must be made in the QIR for classical and quantum control flows as well as data flows. In particular, quantum-to-classical conversions are accomplished via quantum measurements, while quantum conditionals and entangled switch statements are implemented using quantum multiplexer gates.[5] High-level optimizations may involve simultaneous changes to quantum and classical control flows and to data flows. We also consider fault-tolerant constructions at various phases in the design flow and incorporate circuit synthesis and optimization techniques in both the technology-independent and technology-dependent phases.

Circuit synthesis and optimization

During the second and third phases, the QCC synthesizes and optimizes a QASM representation of a quantum circuit using procedures similar to those currently used for digital circuits. Algorithms for classical logic circuit synthesis map a Boolean function into a circuit using gates from a given gate library. Similarly, quantum circuit synthesis creates a circuit that performs a given unitary transform up to an irrelevant global phase or a prescribed quantum measurement.

A digital logic designer can immediately construct a two-level circuit of a Boolean function, linear in the size of the function's truth table, and then use various techniques to optimize it. In contrast, finding a good quantum circuit to implement a $2^n \times 2^n$ unitary matrix is difficult. Only very recently have constructive algorithms become available that yield an asymptotically optimal circuit with $O(4^n)$ gates. Because CNOT gates are typically most expensive, their counts have been pushed down to only a factor of two away from lower bounds.[5] Remaining gates operate on single qubits at a time, but unlike CNOT gates their functionality can be tuned using continuous parameters.

When developing reusable software for automating quantum circuit design, reducing technological dependence is desirable. Today, the NAND gate is easier to implement than the AND gate in CMOS-based integrated circuits. Commercial circuit synthesis tools address this by decoupling libraryless logic synthesis from technology mapping. The former step uses an abstract gate library, such as AND-OR-NOT, and emphasizes the scalability of synthesis algorithms that capture the given

Sample Design Flow: EPR Pair Creation

Accurately capturing quantum-mechanical systems using traditional 0s and 1s is inherently difficult. Quantum information must therefore be processed directly—without converting it to bits—during state transformation and teleportation, communication, measurements, and other common tasks.

Figure A illustrates how our proposed four-phase design flow automates the transformation of mathematical models into software for controlling a live physical system.

An algorithm designer, researcher, or engineer initially expresses a mathematical specification of a quantum algorithm in a high-level quantum programming language, automatically creating a quantum circuit that encapsulates the mathematical abstractions of quantum mechanics and linear algebra.

In the design flow's first phase, the quantum computer compiler abstracts the quantum circuit as a quantum intermediate representation (QIR). Next, the QCC translates the circuit into Quantum Assembly Language (QASM) that captures a universal set of quantum gates. In the third phase, the QCC translates QASM instructions into Quantum Physical Operations Language using software tools. QPOL has knowledge of particulars of the quantum device, including layout and a technology-specific gate library. Finally, technology-dependent software tools translate QPOL into machine instructions.

In this example, we demonstrate how to produce Einstein-Podolsky-Rosen (EPR) pairs[1] for implementation on a trapped-ion computer. Trapped-ion systems have shown considerable potential as a future quantum computing technology.[2] These computers use charged, electromagnetically trapped atoms as qubit carriers and the internal state of single ionized atoms as qubits. Ions can be shuttled in and out of ion traps to increase the quantum computer's effective size.

An important physical resource for quantum computing and communication, EPR pairs are entangled quantum states that cannot be decomposed into (tensor products of) single-qubit states. They represent *quantum nonlocality* and have applications in quantum state teleportation, ultraprecise measurement, and lithography as well as in a number of quantum computing algorithms.

For EPR pair creation, we abstract the mathematical representation in a quantum circuit composed of a Hadamard (H) and CNOT gate. The figure shows sample QASM and QPOL representations. Determining the phase in which to insert fault tolerance and error correction is an open research question; here we show how to replace a CNOT gate with a circuit for a fault-tolerant encoded CNOT operation limited to local interactions.

QPOL instructions for creating an EPR pair can be translated into a sequence of laser pulses—in this case, for performing a CNOT gate on an ion-trap device. The machine instructions are as follows:

1. Alternately raise and lower the potentials of electrodes A, 1, 2, and 3 to move ions from trap A to trap B.
2. Apply a laser to the "green" ion to cool the ion chain that may have heated during movement.
3. Apply π pulse on the first red sideband of the *x* ion.
4. Apply pulse on carrier of the *y* ion.
5. Apply π pulse on the first red sideband of the *x* ion.
6. Split the "green" ion and the *x* ion away from the *y* ion and move them back to trap A.

The six-step process could take around 10-100 μ computation's global structure. The latter step converts all gates of a logic circuit to gates from a technology-specific gate library, often supplied by a chip manufacturer, and is based on local optimizations.

We expect the distinction between technology-independent circuit synthesis and technology mapping to carry over to quantum circuits.[6] This is precisely why the QCC maps the quantum algorithm into a QASM representation consisting of single-qubit and CNOT gates in the second phase of our design flow.

In addition, temporary decompositions into elementary gates could help optimize pulse sequences and reduce systematic inaccuracies in physical implementations. For example, a CNOT gate can be mapped onto a specific technology by appropriately timing pulses that couple two qubits, with pre- and postprocessing by less sophisticated pulses that affect single qubits.[6]

Technology-mapped circuits could potentially be optimized further via automatic instantiation of error correction, efficient handling of universal gate libraries without tunable gates, and identification of reusable quantum logic blocks and their efficient implementation.

Quantum Assembly Language

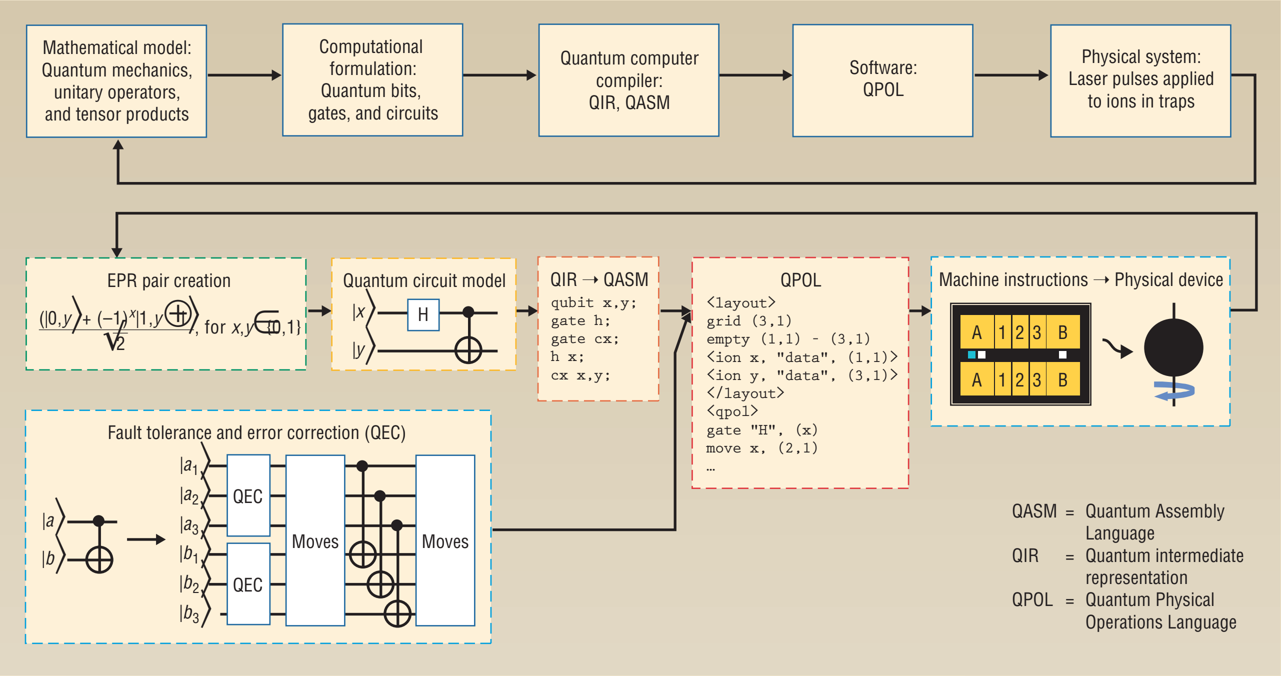
During the technology-independent phase of our design flow, the QCC maps a representation of the quantum algorithm into an equivalent set of Quantum Assembly Language instructions. QASM is a classical reduced-instruction-set computing assembly language extended by a set of quantum instructions based on the quantum circuit model. It uses qubits and registers of classical bits (cbits) as static units of information that must be declared at the program's beginning. Quantum instructions in QASM consist solely of single-qubit unitary gates, CNOT gates, and measurements. Any quantum circuit can be constructed using these instructions.

Quantum Physical Operations Language

QPOL precisely describes the execution of a given quantum algorithm expressed as a QASM program on a particular technology, like trapped-ion systems. QPOL includes physical operations as well as technology-

References

1. A. Einstein, B. Podolsky, and N. Rosen, "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?" *Physical Rev.,* vol. 47, no. 10, 1935, pp. 777-780.
2. D.J. Wineland et al., "Experimental Issues in Coherent Quantum-State Manipulation of Trapped Atomic Ions," *J. Research of NIST,* vol. 103, no. 3, 1998, pp. 259-328.



Mathematical model: Quantum mechanics, unitary operators, and tensor products
Computational formulation: Quantum bits, gates, and circuits
Quantum computer compiler: QIR, QASM
Software: QPOL
Physical system: Laser pulses applied to ions in traps
EPR pair creation
$\frac{(|0,y\rangle + (-1)^x|1,y \oplus 1\rangle)}{\sqrt{2}}$, for $x,y \in \{0,1\}$
Quantum circuit model
$|x\rangle$
$|y\rangle$
H
QIR → QASM
QPOL
Machine instructions → Physical device
A 1 2 3 B
A 1 2 3 B
Fault tolerance and error correction (QEC)
$|a\rangle$
$|b\rangle$
$|a_1\rangle$
$|a_2\rangle$
$|a_3\rangle$
$|b_1\rangle$
$|b_2\rangle$
$|b_3\rangle$
QEC
QEC
Moves
Moves

```
qubit x,y;
gate h;
gate cx;
h x;
cx x,y;
```

```
<layout>
grid (3,1)
empty (1,1) - (3,1)
<ion x, "data", (1,1)>
<ion y, "data", (3,1)>
</layout>
<qpol>
gate "H", (x)
move x, (2,1)
…
```

QASM	=	Quantum Assembly Language
QIR	=	Quantum intermediate representation
QPOL	=	Quantum Physical Operations Language

***Figure A.** Using quantum information processing to control live physical systems. Proposed four-phase design flow, detailed for EPR pair creation on a trapped-ion computer with machine instructions translated into a sequence of laser pulses that perform a CNOT gate. A feedback loop allows for repetition of earlier phases.*

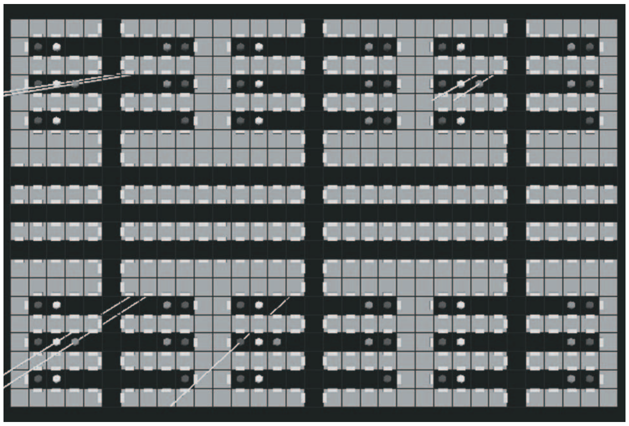


Figure 2. Trapped-ion simulator. Graphical display shows an H-tree layout. Qubits are electronic states of ions, represented by spheres, and gates are laser pulses, represented by lines. The qubits can move within the black regions but not into the substrate, drawn using light squares.

specific modules. In particular, it organizes physical operations into five instruction types:

- *Initialization* instructions specify how to prepare the initial system state. This can include loading qubits into the quantum computer, initializing auxiliary physical states used in computations, and setting qubits to $|0\rangle$.
- *Computation* instructions include quantum gates and measurements.
- *Movement* instructions control the relative distance between qubits to bring them together to undergo simultaneous operations or move them apart.
- *Classical control* instructions provide simple logic operations and allow quantum gates to be applied based on classical bit values stored in classical memory.
- *System-specific* instructions control physical parameters of the system that do not explicitly fall into the other categories.

The final QPOL distributes these instructions to the available instruction processing units—highly parallel quantum computers will have many—and by inserting appropriate waiting times.

In the case of trapped-ion computers, initialization has three stages: loading of multiple ions into a loading region, laser cooling to reduce ion temperatures, and optical pumping to put all qubits into a known state.

Computation is naturally described in terms of single-qubit rotations and a controlled-phase gate between ions in the same trap, both achieved using a laser pulse sequence. Measurement uses another laser pulse that causes ions in the $|0\rangle$ state to fluoresce. Electrostatic fields can move ions between multiplexed traps, and they can move multiple ions in and out of the same trap.

An external classical processor controls the execution of QPOL instructions, stores measurement results, and performs conditional instructions based on stored cbits.

System-specific instructions recool ions when they heat due to movement operations. Certain laser pulses also accomplish recooling, but the lasers are applied differently for cooling than for gates, requiring different programming and pulse-sequence optimization.

HIGH-PERFORMANCE SIMULATION OF QUANTUM CIRCUITS

Quantum-mechanical effects are useful for accelerating certain classical computations, as Lov Grover[7] and Peter Shor[8] have shown; however, numerical simulation of quantum computers on classical computers remains important for engineering reasons.

In classical electronic design automation, chip designers always test independent modules and complete systems by simulating them on test vectors before costly manufacturing. Numerical simulations can also help to evaluate quantum heuristics that defy formal worst-case analysis or only work well for a fraction of inputs.

For the numerical simulation phase of our design flow, we again use the quantum circuit formalism. Because mathematical models of quantum states, quantum gates, and measurement involve linear algebra, a key aspect of efficient simulation is exploiting the structure in the matrices and vectors derived from quantum circuits. To this end, researchers have proposed polynomial-time simulation techniques for circuits arising in error correction[9] and for "slightly entangled" quantum computation.

QuIDDPro: A generic graph-based simulator

George Viamontes and colleagues[10] have proposed a generic simulation technique based on data compression using the *quantum information decision diagram* (QuIDD) data structure. Its worst-case performance is no better than what can be achieved with basic linear algebra, but it can dramatically compress structured vectors and matrices, including all basis states, small gates, and some tensor products.

A QuIDD is a directed acyclic graph with one source and multiple sinks, each labeled with a complex number. The graph models matrix and vector elements as directed paths; any given vector or matrix can be encoded as a QuIDD and vice versa. Graph algorithms working on QuIDDs, supplied as a software library, implement all linear-algebraic operations in terms of compressed data representations.

Time and memory used by these algorithms to simulate a useful class of quantum circuits scale polynomially with the number of qubits. All components of Grover's algorithm, except for some application-dependent oracles, fall into this class. QuIDD-based simulation of the algorithm requires time and memory resources that are polynomial in the oracle function's size. If a compact

QuIDD can represent a particular oracle function for some search problem, then classical simulation of the algorithm runs nearly as fast as an ideal quantum circuit.

QuIDDs can also simulate density matrices by implementing several additional operations, such as traceovers, in terms of graph traversals.[10] Straightforward modeling of any 16-qubit density matrix would require 64 Tbytes of memory. In contrast, for a reversible 16-qubit adder circuit using CNOT and Toffoli gates, the QuIDDPro package (http://vlsicad.eecs.umich.edu/quantum/qp) requires less than 5 Mbytes.

Trapped-ion simulator

Numerical simulations of quantum systems are also useful when studying the feasibility or performance of specific physical implementations.[9] We have carried out such a simulation for trapped-ion systems with up to 1,000 qubits; this applies to quantum *stabilizer circuits*, which are central to quantum error correction.

The keys to such realistic simulations are the layout of qubits in physical space and the scheduling of operations. Our layout tool maps circuits onto an *H-tree,* a recursively constructed fractal layout. This reduces movement operations required per gate by keeping qubits in inner codes near one another within concatenated quantum codes, which also have a self-similar structure. Our scheduler tool uses implicitly specified paths to optimize for minimal distances, expanding QASM instructions to include movements.

The simulator output includes the final quantum state (for circuit verification), measurement and failure histories, total execution time, and, in the case of a fault-tolerant circuit, validity of the final output. As Figure 2 shows, output also is a graphical display of QPOL instructions as they are simulated.

DESIGN FLOW FOR FAULT-TOLERANT ARCHITECTURES

The inherently noisy nature of quantum computers requires inserting error-correction routines and replacing gates with their fault-tolerant implementations to achieve scalability. A system architect can apply this process manually, synthesizing and laying out each fault-tolerant gate (*architecture-driven design*), or a compiler can apply it algorithmically (*software-driven design*).

We are currently considering both processes for trapped-ion computing systems, but the principles extend to other physical systems. The central goal of both designs is to guarantee that the final sequence of physical operations will execute fault-tolerantly on the target system—if failures occur infrequently enough, then the resulting errors cannot cause the system to fail.

Fault-tolerant classical components

In special applications of modern digital computers, the canonical method for fault-tolerant computation is triple modular redundancy.[11] TMR involves feeding gate inputs copied three times into three gates that fail with probability $O(p)$. The output lines of these faulty gates fan out into three majority voting gates. The majority gates essentially amplify the correct value of the computation so that the fault-tolerant gate fails only if two or more failures occur. Mathematically, the fault-tolerant gate fails with probability $O(p^2)$.

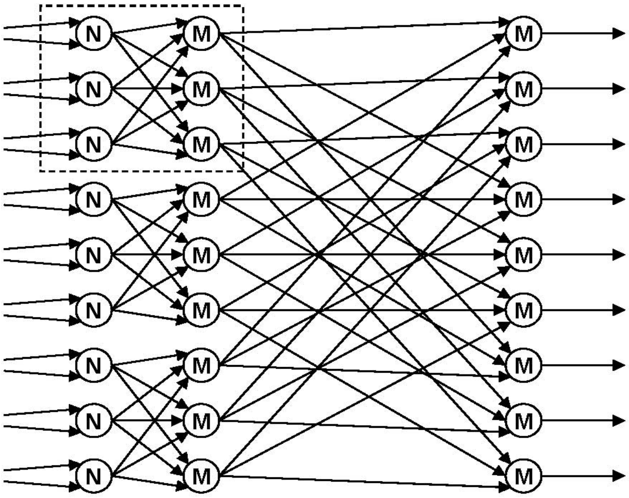


Figure 3. TMR fault-tolerant NAND gate at the second level of recursion, constructed from three fault-tolerant NAND (N) gates and three majority (M) gates.

Figure 3 shows a TMR fault-tolerant NAND gate at the second level of recursion, constructed from three fault-tolerant NAND gates and three majority gates. All gates are assumed to fail with probability p, such that the highlighted TMR NAND gate fails with probability $< 6p^2$, ignoring input errors. The entire circuit shown fails with probability $< 6^3p^4$. If $p < 1/6$, then this circuit is more reliable than a basic gate.

Applying TMR recursively k times, as illustrated in Figure 3 for $k = 2$, fault-tolerant components can be made to fail with probability bounded above by $p_f(k) = (cp)^{2k}/c$. The constant c is determined by the maximum number of fault paths through the highlighted circuit that lead the circuit to fail. In this case, $c = 6$ because at least two gates or two majority voters must fail. If each basic gate fails with probability $p < 1/c$, then $p_f(k) \rightarrow 0$ as $k \rightarrow \infty$. This construction exhibits a *fault-tolerance threshold* $p_{th} = 1/c$.

Fault-tolerant quantum components

We construct fault-tolerant quantum components using procedures similar to classical fault-tolerance techniques. They can encode quantum information using *quantum computation codes*[12] that allow fault-tolerant computation via a discrete universal set of gates. Calderbank-Shor-Steane codes are one family of quantum codes that allow a transversal implementation of an encoded CNOT gate. Transversal gates are always fault tolerant because they

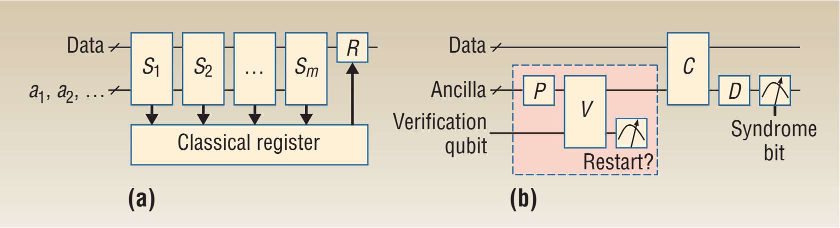


Figure 4. Fault-tolerant quantum computation. (a) Recovery operation. (b) Single syndrome bit extraction.

are implemented in a bitwise fashion—a gate between a pair of encoded qubits is implemented by applying the gate from bit 1 of the first encoded qubit to bit 1 of the second encoded qubit, and so on.

However, there are no known computation codes for which a universal set of encoded operations can be implemented transversally. In practice, performing quantum gates requires fault-tolerant preparation of several kinds of *ancillas,* or scratch qubits. After each gate, we insert on each qubit a recovery operation that consumes a *syndrome extraction ancilla* to acquire syndrome bits. Syndrome extraction ancillas must be available in great supply and may need to be checked for critical errors using *verification ancillas*. All of these operations must remain fault tolerant when qubits can only interact locally.[13]

Figure 4 illustrates key aspects of this process. A recovery operation, shown in Figure 4a, interacts fault-tolerantly with the data via syndrome bit extraction networks S_1,S_2, … S_m. This involves using a syndrome extraction ancilla (a_1,a_2, …) to measure each syndrome bit, possibly several times, and storing the results to a classical register. A classical computer processes the register and applies the appropriate error correction R to the data. Recovery operations follow every fault-tolerant gate to correct errors potentially introduced by that gate.

As Figure 4b shows, extracting a single syndrome bit fault-tolerantly first requires an ancilla state. The highlighted network prepares (P)and verifies (V) the ancilla; a verification qubit indicates if the ancilla failed the verification network V. Upon successful preparation of an ancilla, the C network interacts with the data fault-tolerantly to collect a syndrome bit. The quantum network D then decodes and measures the bit. Some classical postprocessing may take the place of D.

Fault-tolerant architectures

A quantum computation code conceptually separates the logical and physical machine. Both architecture-driven and software-driven designs exploit this fact to yield two different processes within the framework of our design flow.

An architecture-driven design process inserts fault-tolerant gates from a predesigned library during technology-dependent code generation. A design team creates the library of universal, fault-tolerant, technology-specific components using a combination of replacement rules, heuristic methods, and device models, then publishes the library together with design rules for connecting the composite components.

A software-driven design process inserts fault-tolerant gates during technology-independent code generation using replacement rules based on quantum circuits. Sophisticated schedulers and layout tools insert QPOL instructions to preserve fault tolerance. Algorithmic optimizations make fine-grained replacements, and compilers can use feedback from simulators to focus the optimizers on the circuit's critical regions. Our software architecture allows such insertion and testing of error-correction and fault-tolerance techniques at multiple stages in the design flow.

Our work has thus far focused on the languages, transformations, and fault-tolerance procedures needed along the design flow to produce robust implementations. However, many important challenges remain to be solved before researchers can build or even realistically design a scalable quantum computer.

To effectively use available quantum resources, we must be able to schedule and synchronize parallel quantum computations. We also need efficient technology-independent optimization algorithms for realistic classes of quantum circuits as well as strategies for adapting generic circuits to specific architectural constraints and implementation technologies.

Identifying and evaluating meaningful architectural design blocks will necessitate further development of simulation techniques for quantum circuits and high-level programs.

Achieving robust, scalable quantum computation will require both fault-tolerant architectural strategies compatible with emerging quantum device technologies and optimization algorithms that minimize the number of fault paths, code size, or number of gates in fault-tolerant circuits.

It will also be necessary to match tools to experimental implementations as well as develop methodologies for design verification and test such as quantum state tomography, circuit-equivalence checking, and test-vector generation.

The grandest challenge of all is to design a high-level programming language that encapsulates the principles of quantum mechanics in a natural way so that physicists and programmers can develop and evaluate more quantum algorithms.

Design and verification tools for robust quantum circuits are vital to the future of quantum information processing systems, and their development will be a natural evolutionary step as such machines graduate from the laboratory to engineering design. ■

Acknowledgments

We are grateful to Stephen Edwards for many helpful comments on computer-aided design flows. Krysta Svore acknowledges support from an NPSC fellowship, Andrew Cross acknowledges support from an NDSEG fellowship, and Igor Markov acknowledges funding from the DARPA QuIST program and the NSF.

References

1. M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information,* Cambridge Univ. Press, 2000.
2. S. Bettelli, T. Calarco, and L. Serafini, "Toward an Architecture for Quantum Programming," *The European Physics J. D,* vol. 25, no. 2, 2003, pp. 181-200.
3. B. Ömer, "A Procedural Formalism for Quantum Computing," doctoral dissertation, Dept. Theoretical Physics, Technical Univ. of Vienna, 1998.
4. A.V. Aho, R. Sethi, and J.D. Ullman, *Compilers: Principles, Techniques, and Tools,* Addison-Wesley, 1986.
5. V.V. Shende, S.S. Bullock, and I.L. Markov, "Synthesis of Quantum Logic Circuits," to appear in *IEEE Trans. Computer-Aided Design of Integrated Circuits,* 2006; http://arxiv.org/abs/quant-ph/0406176.
6. V.V. Shende, I.L. Markov, and S.S. Bullock, "Finding Small Two-Qubit Circuits," *Proc. SPIE,* vol. 5436, Apr. 2004, pp. 348-359.
7. L.K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," *Proc. 28th Ann. ACM Symp. Theory of Computing,* ACM Press, 1996, pp. 212-219.
8. P.W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM J. Computing,* vol. 26, no. 5, 1997, pp. 1484-1509.
9. S. Aaronson and D. Gottesman, "Improved Simulation of Stabilizer Circuits," *Physical Rev. A,* vol. 70, no. 5, 2004; www.scottaaronson.com/papers/chp5.pdf.
10. G.F. Viamontes, I.L. Markov, and J.P. Hayes, "Graph-Based Simulation of Quantum Computation in the State-Vector and Density-Matrix Representation," *Quantum Information and Computation*, vol. 5, no. 2, 2005, pp. 113-130.
11. J. von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," *Automata Studies,* C.E. Shannon and J. McCarthy, eds., Princeton Univ. Press, 1956, pp. 329-378.
12. D. Aharonov and M. Ben-Or, "Fault-Tolerant Computation with Constant Error," *Proc. 29th Ann. ACM Symp. Theory of Computing,* ACM Press, 1997, pp. 176-188.
13. K.M. Svore, B.M. Terhal, and D.P. DiVincenzo, "Local Fault-Tolerant Quantum Computation," *Physical Rev. A*, vol. 72, no. 5, 2005; http://arxiv.org/abs/quant-ph/0410047.

Krysta M. Svore *is a PhD student in the Department of Computer Science at Columbia University. Her research interests include quantum computation, particularly quantum fault tolerance and error correction, as well as data mining and intrusion detection. Svore received an MS in computer science from Columbia University. Contact her at kmsvore@cs.columbia.edu.*

Alfred V. Aho *is Lawrence Gussman Professor of Computer Science and vice chair for undergraduate education in the Department of Computer Science at Columbia University. His research interests include quantum computing, programming languages, compilers, and algorithms. Aho received a PhD in electrical engineering and computer science from Princeton University. He is a Fellow of the American Association for the Advancement of Science, the ACM, and the IEEE. Contact him at aho@cs.columbia.edu.*

Andrew W. Cross *is a PhD candidate in the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology, and a research assistant in the Quanta Group at MIT Media Lab's Center for Bits and Atoms. His research focuses on fault-tolerant quantum computing. Cross received an MS in electrical engineering and computer science from MIT. Contact him at awcross@mit.edu.*

Isaac Chuang *is an associate professor with joint appointments in the Department of Electrical Engineering and Computer Science and the Department of Physics at MIT, where he also leads the Quanta Group at MIT Media Lab's Center for Bits and Atoms. His research interests include quantum information science, AMO implementations of quantum computers, quantum algorithms, and architectures for quantum information systems. Chuang received a PhD in electrical engineering from Stanford University. Contact him at ichuang@mit.edu.*

Igor L. Markov *is an assistant professor in the Department of Electrical Engineering and Computer Science at the University of Michigan. His research interests include physical design and physical synthesis for VLSI, synthesis and simulation of quantum circuits, and artificial intelligence. Markov received a PhD in computer science from the University of California, Los Angeles. He is a member of the ACM and the IEEE Computer Society, and a senior member of the IEEE. Contact him at imarkov@eecs.umich.edu.*

A.3 ISCA'03 Paper

Appears in *Proceedings of the 30th International Symposium on Computer Architecture* (ISCA 2003)

Building Quantum Wires: The Long and the Short of it

Mark Oskin[†], Frederic T. Chong[‡], Isaac L. Chuang[*], John Kubiatowicz[○]
[†]University of Washington, [‡]University of California, Davis
[*]Massachusetts Institute of Technology, [○]University of California, Berkeley

Abstract

As quantum computing moves closer to reality the need for basic architectural studies becomes more pressing. Quantum wires, which transport quantum data, will be a fundamental component in all anticipated silicon quantum architectures. In this paper, we introduce a quantum wire architecture based upon quantum teleportation. We compare this teleportation channel with the traditional approach to transporting quantum data, which we refer to as the swapping channel. We characterize the latency and bandwidth of these two alternatives in a device-independent way and describe how the advanced architecture of the teleportation channel overcomes a basic limit to the maximum communication distance of the swapping channel. In addition, we discover a fundamental tension between the scale of quantum effects and the scale of the classical logic needed to control them. This "pitch-matching" problem imposes constraints on minimum wire lengths and wire intersections, which in turn imply a sparsely connected architecture of coarse-grained quantum computational elements. This is in direct contrast to the "sea of gates" architectures presently assumed by most quantum computing studies.

1 Introduction

Many important problems seem to require exponential resources on a classical computer. Quantum computers can solve some of these problems with polynomial resources, leading a great number of researchers to explore quantum information processing technologies [28, 31, 13, 15, 41, 9, 17, 42]. Early-stage quantum computers have involved a small number of components (less than 10) and have utilized molecules in solution and trapped ions [47, 25, 35]. To exploit our tremendous historical investment in silicon, however, solid-state silicon quantum computers are desirable. Promising proposals along these lines have begun to appear [22, 50]; these even include ideas that merge atomic physics and silicon micromachining[24]. However, as the number of components grows, quantum computing systems will begin to require the same level of engineering as current computing systems. The same process we as computer architects do for classical silicon-based systems, of building abstractions and optimizing structure, needs to be applied to quantum technologies.

Even at this early stage, a general architectural study of quantum computation is important. By investigating the potential costs and fundamental challenges of quantum devices, we can help illuminate previously unforeseen obstacles of constructing a scalable quantum processor. We may also anticipate and specify important subsystems and techniques common to all implementations. Identifying these practical challenges early will help focus the ongoing development of fabrication and device technology. Developing abstractions for quantum technology and basic architectural concepts for it has proven to be quite fascinating.

This paper is about a seemingly mundane subject: a wire. To be clear, we define a wire in the quantum world as a mechanism for moving quantum data from one spatial location to another. Any optimistic view of the future of quantum computing includes enough interacting devices to introduce a spatial extent to the layout of those devices. This spatial dimension, in turn, introduces a need for wires. As we will show, a quantum wire is a very different creature from a classical one. One of the most important distinctions between quantum and classical wires arises from the fact that quantum information (composed of quantum bits or qubits) *cannot be copied* [31]. Instead, it must be *transported* from source to destination – destroying the information at the source and re-creating it at the destination. This fact changes our normal intuitions about the use of buffers to drive wires, repeaters to amplify signals, and fanout to distribute information. In particular, all wires must be *point-to-point* and can only *protect* information rather that amplifying it.

Quantum information can be encoded in a number of ways, such as the spin component of basic particles like protons or electrons, or in the polarization of photons. Thus, there are several ways in which we might transfer information. First, we might physically transport particles from one point to another. In a large solid-state system, the logical candidate for information carriers would be electrons, since they are highly mobile. Unfortunately, electrons are also highly interactive with the environment and hence subject to corruption of their quantum state, a process known as *decoherence*. Second, we might consider passing information along a line of quantum devices. This *swapping channel* is, in fact, a viable option for short distances (as discussed in Section 4), but tends to accumulate errors over long distances. In some ways, this solution resembles a quantum-cellular automata (QCA) [32] wire, except without duplication of data capabilities.

Over longer distances, we need something fundamentally different. We propose to use a technique called *teleportation* [7] and to call the resulting long-distance quantum wire a *teleportation channel* to distinguish from a swapping channel. Teleportation uses an unusual quantum property called *entanglement*, which allows quantum bits to interact instantaneously at a distance[1]. To understand the mathematical details and practical implications of teleportation, we will need to cover some background and prior art before returning to the subject in Section 2.3.

In the remainder of this paper, we will quantify the advantages and disadvantages of swapping channels versus teleportation channels. Realistic concerns such as quantum error-correction [43] for protecting information data errors and entropy exchange [37] for generating zeros and entangled pairs, greatly complicate things. Also important is an often-neglected facet of quantum computing systems — the fact that they depend upon classical signals for control of quantum operations. We will explore the fundamental tension between the scale at which quantum effects occur and the scale at which classical signals can be reliably routed. The architectural implications of this tension manifest themselves as a pitch-matching problem.

Overall, the contributions of this research are:

- We define the basic building blocks required to construct long and short quantum wires.
- We discover that the interface between classical control and quantum devices requires minimum wire lengths between fanout sites. We generalize these limitations in terms of the ratio of quantum and classical devices in a given technology and discuss the architectural implications of these limitations.
- We find that the latency and bandwidth of swapping channels are extremely sensitive to the length of the channel, but that teleportation channels do not exhibit the same sensitivity.

The remainder of this paper continues with a brief introduction to quantum computing in Sections 2 and 3. Section 4 introduces the swapping channels that can be constructed from solid-state technologies and presents an analysis of the scalability problems with these channels. Section 5 presents teleportation channels, our architectural solution to scalable quantum data transport. Section 6 discusses our future work in system bandwidth issues and in Section 7 we conclude.

2 Quantum Computing

We begin with a brief overview of the basic terminology and constructs of quantum computation. Our purpose is to introduce the language necessary for subsequent sections; in-depth treatments of these subjects are available in the literature [31].

2.1 Quantum states: qubits

The state of a classical digital system $\mathcal{X}$ can be specified by a binary string $\mathbf{x}$ composed of a number of bits x_i, each of which uniquely characterizes one elementary piece of the system. For n bits, there are 2^n unique possible states. The state of an analogous quantum system ψ is described by a complex-valued vector $|\psi\rangle = \sum_x c_x|\mathbf{x}\rangle$, a weighted combination (a "superposition") of the basis vectors $|\mathbf{x}\rangle$, where the *probability amplitudes* c_x are complex numbers whose modulus squared sums to one, i.e. $\sum_x |c_x|^2 = 1$.

A single quantum bit is commonly referred to as a *qubit* and is described by the equation $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$. Such a qubit might be represented, for example, by the nuclear spin of an atom. Legal qubit states include pure states, such as $|0\rangle$ and $|1\rangle$, and states in superposition, such as $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. Also valid are $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$, which are other equal superpositions, but with different *relative phases* between the basis states.

Larger quantum systems can be composed from multiple qubits. For example, $|00\rangle$ is a valid two-qubit state, and so is $\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle - \frac{1}{\sqrt{2}}|11\rangle$. An n-qubit state is described by 2^n basis vectors, each with its own complex probability amplitude, so an n-qubit system can exist in an arbitrary superposition of the possible 2^n classical states of the system. To compose multiple independent quantum systems together, the tensor product operator $\otimes$ is used, e.g., $a \otimes b$.

Unlike the classical case, however, where the total can be completely characterized by its parts, the state of larger quantum systems cannot be described simply by giving the individual states of its component qubits. This property, known as *entanglement*, is best illustrated with an example: there exist no single qubit states $|\psi_A\rangle$ and $|\psi_B\rangle$ such that the two-qubit state $|\Psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ can be expressed as the composite state $|\psi_A\rangle \otimes |\psi_B\rangle$. Entanglement does not exist classically, and the unique properties of entangled states are widely believed to be at the heart of what gives quantum computers their computational powers.

Another non-intuitive property of quantum states is their behavior when measured. Upon observation, a quantum state collapses into one of a number of possible classical states, the set of possibilities being determined by the measurement apparatus. Specifically, it is conventional (in the quantum computation and quantum information community) to adopt the *computational basis* states $|0\ldots00\rangle$, $|0\ldots01\rangle$, $|0\ldots10\rangle$, …, $|1\ldots11\rangle$, and choose measurements to collapse states into this basis. The probability that a particular basis state $\mathbf{x}$ results is $|c_x|^2$, the modulus square of the probability amplitude for the basis vector $\mathbf{x}$. For example, when $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ is measured, the outcome is $|0\rangle$ or $|1\rangle$ with equal probability. Similarly, when the state $|\Psi\rangle$,

[1] Although this property sounds suspiciously like "faster-than-light" communication, we shall see that the interaction is ambiguous without the additional transmission of two bits of classical information, which must travel at a subluminal velocity.

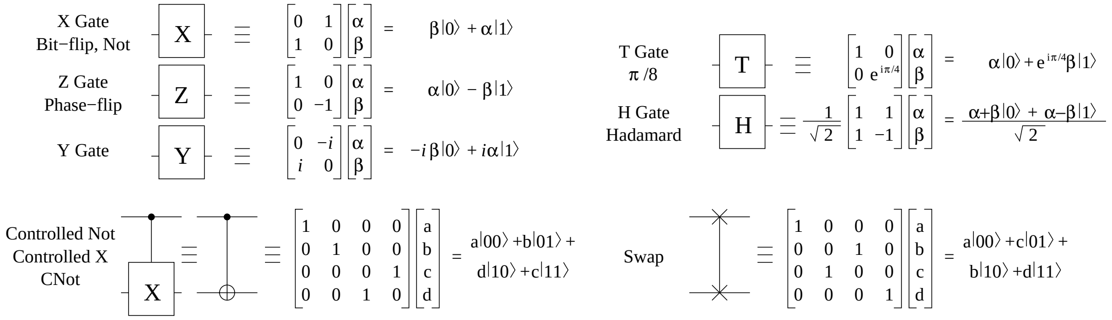


Figure 1. Basic Quantum Gates and their matrix representations.

above, is measured, the result is either $|00\rangle$ or $|11\rangle$, with equal probability; the outcomes $|01\rangle$ or $|10\rangle$ never occur.

Due to the probabilistic nature of measurement, designers of quantum algorithms must be very clever about how to get useful answers out of their computations. One method is to iteratively skew probability amplitudes in a qubit vector until the desired value is near $|1\rangle$ and the other values are close to $|0\rangle$. This technique is used in Grover's algorithm for searching an unordered list of n elements [18]. The algorithm goes through $\sqrt{n}$ iterations, at which point a qubit vector representing the keys can be measured. The desired element is found with high probability.

Another option in a quantum algorithm is to arrange the computation such that it does not matter which of many random results is measured from a qubit vector. This method is used in Shor's algorithm for factoring the product of two large primes [40], which is built upon the quantum Fourier transform, an exponentially fast version of the classical discrete Fourier transform. Essentially, the factorization is encoded within the period of a set of highly probable values, from which the desired result can be obtained no matter what value is measured. Since the tractability of factoring the product of two large primes is the basis of nearly all public-key cryptographic security systems, Shor's algorithm has received much attention.

For the interested reader, quantum algorithms for a variety of problems other than search and factoring have been developed: adiabatic solution of optimization problems (the quantum analogue of simulated annealing) [11], precise clock synchronization (using EPR pairs to synchronize GPS satellites) [21, 12], quantum key distribution (provably secure distribution of classical cryptographic keys) [6], and very recently, Gauss sums [46], testing of matrix multiplication (in $O(n^{1.75})$ steps versus the $O(n^2)$ required classically) [20], and Pell's equation [19].

2.2 Quantum gates and circuits

Just as bits can be flipped using a NOT gate, and interact with each other via multi-bit logic gates such as the XOR, qubits can be operated on by gates such as those shown in Figure 1. In the quantum realm, the role of the classical truth table is played by a unitary operator U. The output state vector is the operator applied to the input vector; that is, $|\psi_{\text{out}}\rangle = U|\psi_{\text{in}}\rangle$. The X gate is analogous to the classical NOT gate: it flips $|0\rangle$ and $|1\rangle$. The Z gate is something new to the quantum realm: it flips the phase of the $|1\rangle$ state, thus exchanging $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The Hadamard gate H is another unusual single-qubit gate: it turns $|0\rangle$ into $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle$ into $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$; it can be thought of as performing a radix-2 Fourier transform. Another important single-qubit gate, T, leaves $|0\rangle$ unchanged but multiplies $|1\rangle$ by $\sqrt{i}$. And analogous to the classical XOR gate is the quantum controlled-NOT (or CNOT) gate.

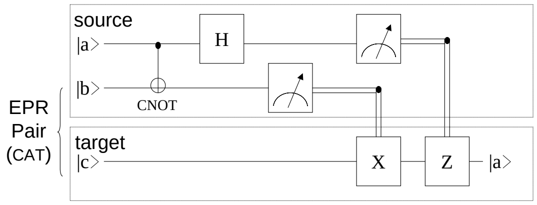


Figure 2. **Quantum Teleportation of state $|a\rangle$ over distance. First, *entangled* qubits $|b\rangle$ and $|c\rangle$ are exchanged. Then, $|a\rangle$ is combined with $|b\rangle$ after which measurements produce two *classical* bits of information (double lines). After transport, these bits are used to manipulate $|c\rangle$ to regenerate state $|a\rangle$ at destination.**

Together, these gates form a *universal set*: just as any Boolean circuit can be composed from AND and NOT gates, any polynomially describable multi-qubit quantum transform U can be efficiently approximated by composing these quantum gates into a circuit. In addition to these universal gates, one more important operator is the SWAP gate. SWAP can be implemented as three CNOTs. However, SWAP is often available as a basic gate for a given technology, which is a valuable thing, given its importance to quantum communication.

In quantum circuits, time goes from left to right, where single lines represent qubits, and double lines represent classical bits. A meter is used to represent measurement. By convention, black dots represent control terminals for quantum-controlled gates. The $\oplus$ symbol is shorthand for the target qubit of the CNOT gate (Figure 2).

2.3 Quantum teleportation

Quantum teleportation is the re-creation of a quantum state at a distance. Contrary to its science fiction counterpart, quantum teleportation is not instantaneous transmission of information. Rather, it uses an entangled *EPR pair*, $|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ [4].

Figure 2 gives an overview of the teleportation process. We start by generating an EPR pair. We separate the pair, keeping one qubit, $|b\rangle$, at the source and transporting the other, $|c\rangle$, to the destination. When we want to send a qubit, $|a\rangle$, we first interact $|a\rangle$ with $|b\rangle$ using a CNOT gate. We then measure $|a\rangle$ and $|b\rangle$ in the computational basis, and send the two one-bit classical results to the destination, and use those results to re-create the correct phase and amplitude in $|c\rangle$ such that it takes on the original state of $|a\rangle$. The re-creation of phase and amplitude is done with X and Z gates, whose application is contingent on the outcome of the measurements of $|a\rangle$ and $|b\rangle$. Intuitively, since $|c\rangle$ has a special relationship with $|b\rangle$, interacting $|a\rangle$ with $|b\rangle$ makes $|c\rangle$ resemble $|a\rangle$, modulo a phase and/or amplitude error. The two measurements allow us to correct these errors and re-create $|a\rangle$ at the destination. Note that the original state of $|a\rangle$ is destroyed when we take our two measurements. This is consistent with the "no-cloning" theorem, which states that a quantum state cannot be copied.

Why bother with teleportation when we end up transporting $|c\rangle$ anyway? Why not just transport $|a\rangle$ directly? First, we can pre-communicate EPR pairs with extensive pipelining without stalling computations. Second, it is easier to transport EPR pairs than real data. Since $|b\rangle$ and $|c\rangle$ have known properties, we can employ a specialized procedure known as *purification* to turn a collection of pairs partially damaged from transport into a smaller collection of asymptotically perfect pairs. Third, transmitting the two classical bits resulting from the measurements is more reliable than transmitting quantum data.

3 Solid-State Technologies

With some basics of quantum operations in mind, we turn our attention to the technologies available to implement these operations. Experimentalists have examined several technologies for quantum computation, including Josephson junctions [30, 50], trapped ions [29], photons [45], bulk spin NMR [48], and phosphorus impurities in silicon [22]. Of these proposals, only those building on a solid-state platform are expected to provide the scalability required to achieve a useful computational substrate. The Kane [22, 42] schemes of phosphorus in silicon builds upon modern semiconductor fabrication and transistor design, drawing upon understood physical properties. To focus the presentation in this paper we begin our calculations with the Kane proposal, and then generalize to consider limits imposed by any solid-state technology. This quantum analysis proceeds in precisely the same manner that it would in the classical domain—by characterizing device technologies with a few underlying parameters.

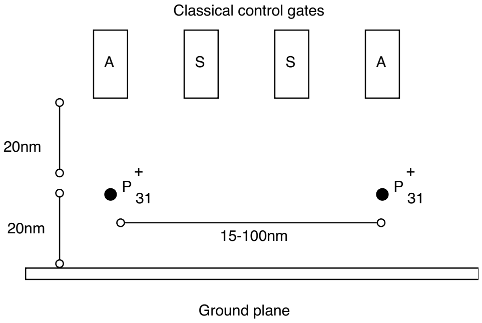


Figure 3. The basic quantum bit technology proposed by Kane [42]. Qubits are embodied by the nuclear spin of a phosphorus atom coupled with an electron embedded in silicon under high magnetic field at low temperature.

Kane proposes that the nuclear spin of a phosphorus atom coupled with an electron embedded in silicon under a high magnetic field and low temperature can be used as a quantum bit, much as nuclear spins in molecules have been shown to be good quantum bits for quantum computation with nuclear magnetic resonance [15]. This quantum bit is classically controlled by a local electric field. The process is illustrated in Figure 3. Shown are two phosphorus atoms spaced 15-100 nm apart. This inter-qubit spacing is currently a topic of debate within the physics community, with conservative estimates of 15nm, and more aggressive estimations of 100nm. What is being traded off is noise immunity versus difficulty of manufacturing. For our study, we will use a figure (60nm) that lies between these two. We parameterize our work, however, to generalize for changes in the underlying technology.

Twenty nanometers above the phosphorus atoms lie three classical wires that are spaced 20 nm apart. By applying precisely timed pulses to these electrodes Kane describes how arbitrary one- and two-qubit quantum gates can be realized. Four different sets of pulse signals must be routed to each electrode to implement a universal set of quantum operations. The details of the pulses and quantum mechanics of this technique are beyond the scope of this paper and are described in [42].

The Kane proposal, like all quantum computing proposals, uses classical signals to control the timing and sequence of operations. All known quantum algorithms, including basic error correction for quantum data, require the determinism and reliability of classical control. Without efficient classical control, fundamental results demonstrating the feasibility of quantum computation do not apply (such as the Threshold Theorem used in Section 4.2.3).

Quantum computing systems display a characteristic tension between computation and communication. Fundamentally, technologies that transport data well do so be-

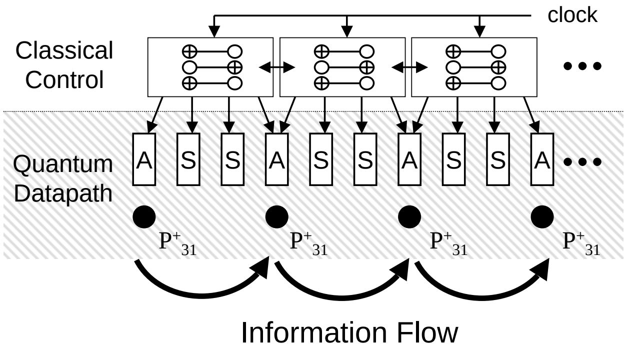


Figure 4. Short wires are constructed from successive qubits (phosphorus atoms). Information in the quantum data path is swapped from atom to atom by classical control. This localized control produces swapping behavior through a repeated series of three back-to-back CNOT **operations.**

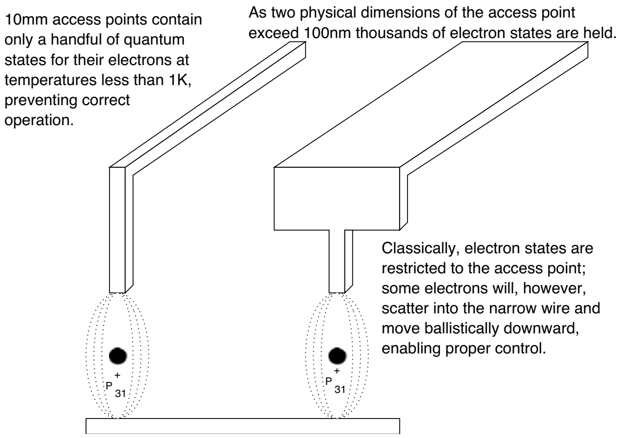

10mm access points contain only a handful of quantum states for their electrons at temperatures less than 1K, preventing correct operation.
As two physical dimensions of the access point exceed 100nm thousands of electron states are held.
Classically, electron states are restricted to the access point; some electrons will, however, scatter into the narrow wire and move ballistically downward, enabling proper control.
P+31
P+31

Figure 5. Quantization of electron states overcome by increasing the physical dimension of the control lines beyond 100 nm. The states propagate quantum-mechanically downward through access vias to control the magnetic field around the phosphorus atoms.

cause they are resistant to interaction with the environment or other quantum bits; on the other hand technologies that compute well do so precisely because they *do* interact. Thus, computation and communication are somewhat at odds.

In particular, atomic-based solid-state technologies are good at providing scalable computation but complicate communication, because their information carriers have nonzero mass. The Kane proposal, for example, represents a quantum bit with the nuclear spin of a phosphorus atom implanted in silicon. The phosphorus atom does not move, thus transporting this state to another part of the chip is laborious and requires carefully controlled swapping of the states of neighboring atoms. In contrast, photon-based proposals that use polarization to represent quantum states can easily transport data over long distances through fiber. It is very difficult, however, to get photons to interact and achieve any useful computation. Further, transferring quantum states between atomic and photon-based technologies is extremely difficult.

Optimizing these tensions, between communication and computation, between classical control and quantum effects, imply a structure to quantum systems. Rather than cover the gamut of quantum architecture we instead will focus on a very crucial architectural concept: a wire. Specifically, we begin by examining a short wire.

4 Short Wires

We begin by examining a "short" quantum wire. Section 4.2 shows that the basic short wire does not scale well, hence a more scalable approach appears Section 5.

In solid-state technologies, a line of qubits is one plausible approach to transporting quantum data. Figure 4 provides a schematic of a *swapping channel* in which information is progressively swapped between pairs of qubits in the *quantum datapath*—somewhat like a bubble sort[2]. Swapping channels require active control from classical logic, illustrated by the *classical control* plane of Figure 4.

4.1 Technical Challenges

As simple as it might appear, a quantum swapping channel presents significant technical challenges. The first hurdle is the placement of the phosphorus atoms themselves. The leading work in this area has involved precise ion implantation through masks, and manipulation of single atoms on the surface of silicon [23]. For applications where substantial monetary investment is not an issue, slowly placing a few hundred thousand phosphorus atoms with a probe device [16] may be possible. For bulk manufacturing the advancement of DNA or other chemical self-assembly techniques [1] may need to be developed. Note, while new technologies may be developed to enable precise placement, the key for our work is only the spacing (60 nm) of the phosphorus atoms themselves, and the number of control lines (3) per qubit. The relative scale of quantum interaction and the classical control of these interactions is what will lead our analysis to the fundamental constraints on quantum computing architectures.

A second challenge is the scale of classical control. Each control line into the quantum datapath is roughly 10 nm in width. While such wires are difficult to fabricate, we expect that either electron beam lithography [3], or phase-shifted masks [36] will make such scales possible.

[2] For technologies that do not have an intrinsic swap operation, one can be implemented by three controlled-not gates performed in succession. This is a widely known result in the quantum computing field and we refer the interested reader to [31].

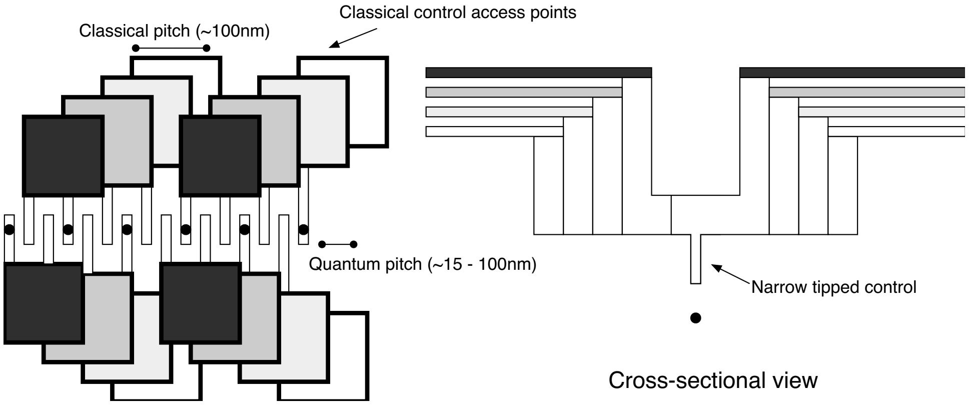


Figure 6. A linear row of quantum bits: In this figure (not drawn to scale) we depict access control for a line of quantum bits. On the left, we depict a "top down" view. On the right is a vertical cross-section which more clearly depicts the narrow-tipped control lines that quickly expand to classical dimensions.

A remaining challenge is the temperature of the device. In order for the quantum bits to remain stable for a reasonable period of time the device must be cooled to less than one degree Kelvin. The cooling itself is straightforward, but the effect of the cooling on the classical logic is a problem. Two issues arise: first conventional transistors stop working as the electrons become trapped near their dopant atoms, which fail to ionize. Second, the 10 nm classical control lines begin to exhibit quantum-mechanical behavior such as conductance quantization and interference from ballistic transport [14].

Fortunately, many researchers are already working on low-temperature transistors. For instance, single-electron transistors (SET's) [27] are the focus of intense research due to their high density and low power properties. SET's, however, have been problematic for conventional computing because they are sensitive to noise and operate best at low temperatures. For quantum computing, this predilection for low temperatures is exactly what is needed! Tucker and Shen describe this complementary relationship and propose several fabrication methods in [44].

On the other hand, the quantum-mechanical behavior of the control lines presents a subtle challenge that has been mostly ignored to-date. At low temperatures, and in narrow wires, the quantum nature of electrons begins to dominate over normal classical behavior. For example, in 100 nm wide polysilicon wires at 100 millikelvin, electrons propagate ballistically like waves, through only one conductance channel, which has an impedance given by the quantum of resistance, $h/e^2 \approx 25$ kΩ. Impedance mismatches to these and similar metallic wires make it impossible to properly drive the AC current necessary to perform qubit operations.

Avoiding such limitations mandates a geometric design constraint: narrow wires must be short and locally driven by nearby wide wires. Using 100 nm as a rule of thumb[3] for a minimum metallic wire width sufficient to avoid undesired quantum behavior at these low temperatures, we obtain a control gate structure such as that depicted in Figure 5. Here, wide wires terminate in 10 nm vias that act as local gates above individual phosphorus atoms.

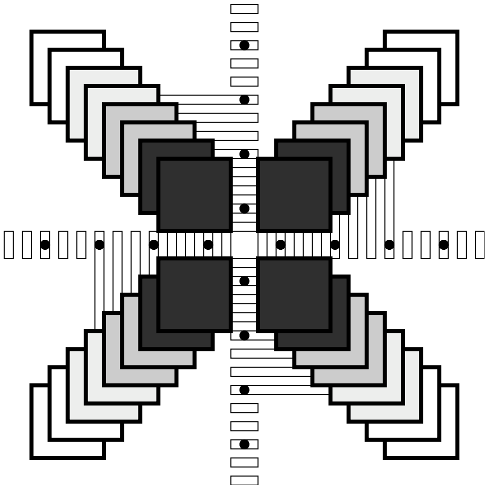


Figure 7. Intersection of quantum bits. In this simplified view, we depict a four-way intersection of quantum bits. An diamond shaped junction is also needed to densely pack junction cells.

Producing a line of quantum bits that overcomes all of the above challenges is possible. We illustrate a design in Figure 6. Note how access lines quickly taper into upper layers of metal and into control areas of a classical scale. These control areas can then be routed to access transistors that can gate on and off the frequencies (in the 10's to 100's of MHz) required to apply specific quantum gates.

Of course, any solution for data transport must also support routing. Routing is not possible without fanout provided by wire intersections. We can extend our linear row of quantum bits to a four-way intersection capable of supporting sparsely intersecting topologies of quantum bits. We illustrate the quantum intersection in Figure 7. This configuration is similar to Figure 6 except that the intersection creates a more challenging tapering.

[3] This value is based on typical electron mean free path distances, given known scattering rates and the electron Fermi wavelength in metals.

4.2 Analysis

We now analyze this short wire to derive two important architectural constraints: the classical-quantum interface boundary and the latency/bandwidth characteristics. We strive to achieve a loose lower bound on these constraints for a given quantum device technology. While future quantum technologies may have different precise numbers, it is almost certain they will continue to be classically controlled, and thus also obey similar constraints based upon this classical-quantum interface.

4.2.1 Pitch Matching

Our first constraint is derived from the need to have classical control of our quantum operations. As previously discussed, we need a minimum wire width to avoid quantum effects in our classical control lines. Referring back to Figure 7, we can see that each quadrant of our four-way intersection will need to be some minimum size to accommodate access to our control signals.

Recall from Figure 3 that each qubit has three associated control signals (one A and two S gates). Each of these control lines must expand from a thin 10 nm tip into a 100 nm access point in an upper metal layer to avoid charge quantization effects at low temperatures (Figure 5). Given this structure, it is possible to analytically derive the minimum width of a line qubits and its control lines, as well as the size of a four-way intersection. For this minimum size calculation, we assume all classical control lines are routed in parallel, albeit spread across the various metal layers. This parallel nature makes this calculation trivial under normal circumstances (sufficiently "large" lithographic feature size λ_c), with the minimum line segment being equal in length to twice the classical pitching, $150nm$ in our case, and the junction size equal to four times the classical pitching, $400nm$, in size. However, we illustrate the detailed computation to make the description of the generalization clearer. We begin with a line of qubits.

Let N be the number of qubits along the line segment. Since there are three gates (an A and two S lines) we need to fit in $3N$ classical access points of 100 nm in dimension each, in the line width. We accomplish this by offsetting the access points in the x and y dimensions (Figure 6) by 20nm. The total size of these offsets will be $100nm$ divided by the qubit spacing $60nm$ times the number of control lines 3 per qubit, times the offset distance of $20nm$. This number $100nm/60nm \times 3 \times 20nm = 100nm$ is divided by 2 because the access lines lines are spread out on each side of the wire. Hence, the minimum line segment will be $100 + 50nm$. Shorter line segments within larger, more specialized cells are possible.

Turning our attention to an intersection (Figure 7), let N be the number of qubits along each "spoke" of the junction. We need to fit $3N$ classical access points in a space of $(60 \text{ nm} \times N)^2$, where each access point is at least 100 nm on a side. As with the case of a linear row of bits, a 20 nm x and y shift in access point positioning between layers is used for via access. Starting with a single access pad of $100nm$, we must fit $100nm/60nm \times 3$ additional pads shifted in x and y within the single quadrant of our intersection. This leads to a quadrant size of $100 + 100nm/60nm \times 3 \times 20nm = 200nm$. Therefore, the minimum size four way intersection is 8 (rounding up) qubits in each direction.

In this construction we have assumed a densely packed edge to each spoke, however, this is easily "unpacked" with a specialized line segment, or by joining to another junction that is constructed inversely from that shown in Figure 7. Obviously, the specific sizes will vary according to technological parameters and assumptions about control logic, but this calculation illustrates the approximate effect of what appears to be a fundamental tension between quantum operations and the classical signals that control them. A minimum intersection size implies minimum wire lengths, which imply a minimum size for computation units.

4.2.2 Technology Independent Limits

Thus far we have focused our discussion on a particular quantum device technology. This has been useful to make the calculations concrete. Nevertheless, it is useful to generalize these calculations to future quantum device technologies. Therefore we parameterize our discussion based on a few device characteristics:

Assuming two-dimensional devices (i.e. not a cube of quantum bits), let p_c be the classical pitching required, and p_q the quantum one. Furthermore, let R be the ratio p_c/p_q of the classical to quantum distance for the device technology, m be the number of classical control lines required per quantum bit, and finally λ_c be the feature size of the lithographic technology. We use two separate variables p_c and λ_c to characterize the "classical" technology because they arise from different physical constraints. The parameter λ_c comes from the lithographic feature size, while p_c (which is a function of λ_c) is related to the charge quantization effect of electrons in gold. With the Kane technology we assume a spacing p_q of $60nm$ between qubits, three control lines per bit of $100nm$ (p_c) each, and a λ_c of 5nm. We can use these to generalize our pitch matching equations. Here we find that the minimum line segment is simply equivalent to $R(1 + 2\lambda_c m/p_q)$ qubits in length.

Examining our junction structure (Figure 7), we note that it is simply four line segments, similar to those calculated above, except that the control lines must be on the same side. Therefore the minimum crossing size of quantum bits in a two-dimensional device is of size $\approx 2R(1 + 4\lambda_c m/p_q)$ on a side.

4.2.3 Latency and Bandwidth

Calculating the latency and bandwidth of quantum wires is similar but slightly different than it is for classical systems. The primary difficulty is decoherence—i.e. quantum noise. Unlike classical systems, if you want to perform a quantum computation, you cannot simply re-send quantum data when an error is detected. The "no-cloning" theorem [31], according to which quantum states cannot be perfectly copied, prohibits transmission by duplication, thereby making it impossible to re-transmit quantum data if it is corrupted. Once the data is destroyed by the noisy channel, you have to start the entire computation over. To avoid this loss, quantum data is encoded in a sufficiently strong error-correcting code that, with high probability, the data will remain coherent for the entire length of the quantum algorithm. Unfortunately, quantum systems will be so error-prone that they will execute right at the limits of their error tolerance [33].

Our goal is to provide a quantum communication layer which sits below higher level error correction schemes. We will discuss our future work, which is the interaction of this layer with quantum error correction and algorithms in Section 6. Consequently, we start our calculation by assuming a channel with no error correction. Then we factor in the effects of decoherence and derive a maximum wire length for our line of qubits.

Recall that data traverses the line of qubits with swap gates, each of which takes approximately 1 μs to execute in the Kane technology. Thus, a single row of quantum bits has latency:

$$latency = 1\ \mu\text{s} \times \text{distance}/60\ \text{nm} \tag{1}$$

This latency can be quite large. A short 1 μm has a latency of 0.000017 seconds! On the plus side, the wire can be fully pipelined and has a sustained bandwidth of $1/1\ \mu\text{s} = 1$M qbps (quantum bits per second). This may seem small compared to a classical wire, but keep in mind that quantum bits hold an exponential amount of information and can enable algorithms with exponential power.

The number of error-free qubits is actually lower than this physical bandwidth. Noise, or decoherence, degrades quantum state and makes the true bandwidth of our wire less than the physical quantum bits per second. Bits decohere over time, so longer wires will have a lower bandwidth than shorter ones.

The stability of a quantum bit over time decays (exactly like an un-error corrected classical bit) as a function $e^{-k\times t}$. Usually, a normalized form of this equation is used, $e^{-\lambda\times t}$, where t in this new equation is the number of operations and λ is related to the time per operation and the original k. As quantum bits traverse through our wire they arrive with a fidelity proportional to the latency, namely:

$$fidelity = e^{-k\times latency} \tag{2}$$

The true bandwidth is then proportional to the fidelity:

$$bandwidth_{true} = bandwidth_{physical} \times fidelity \tag{3}$$

Choosing a reasonable [4] value of $\lambda \approx 10^{-6}$, we find the true bandwidth of a wire to be:

$$1/1\ \mu\text{s} \times e^{-10^{-6}\times distance/60\ \text{nm}} \tag{4}$$

which for a 1 μm wire is close to ideal (999,983 qbps).

This does not seem to be a major effect, until you consider an entire quantum algorithm. Data may traverse back and forth across a quantum wire millions of times. It is currently estimated [2] that a degradation of fidelity more than 10^{-4} makes arbitrarily long quantum computation theoretically unsustainable, with the practical limit being far higher [33]. This limit is derived from the Threshold Theorem, which relates the decoherence of a quantum bit to the complexity of correcting this decoherence [26, 34, 2]. [5] Given our assumptions about λ, the maximum theoretical wire distance is about 6μm, and again the practical wire distance is about two orders of magnitude less than this.

4.2.4 Technology Independent Metrics

Our latency and bandwidth calculations require slightly more device parameters. Let T be the time per basic swap operation. Some technologies will have an intrinsic SWAP, and others will require synthesizing the swap from 3 CNOT operations. Let λ be the decoherence rate, which for small λ and T is equivalent to the decoherence a quantum bit undergoes in a unit of operation time T. This makes the latency of a swapping channel wire equal to:

$$latency = T \times D \tag{5}$$

Where distance D is expressed in the number of qubits. The bandwidth is proportional to the fidelity or:

$$bandwidth_{true} = \frac{1}{T}e^{-\lambda D} \tag{6}$$

This bandwidth calculation is correct so long as the fidelity remains above the critical threshold $C \approx 10^{-4}$ required for fault tolerant computation. Finally, the maximum distance of this swapping channel is the distance when the fidelity drops below the critical threshold:

$$distance_{max} = log_e(1-C)/-\lambda \tag{7}$$

Realize that no amount of error correction will be robust enough to support a longer wire, while still supporting arbitrarily long quantum computation. For this we need a more advanced architecture. One obvious option is to break the wire into segments and insert "repeaters" in the middle. These quantum repeaters are effectively performing state restoration (error correction). However, we can do better, which is the subject of the next section.

[4] This value for λ is calculated from a decoherence rate of 10^{-6} per operation, where each operation requires 1 μs. It is aggressive, but not too unreasonable for phosphorus atoms in silicon. We refer the interested reader to [31].

[5] By "practical" we mean without an undue amount of error correction. The threshold theorem ensures that theoretically we can compute arbitrarily long quantum computations, but the practical overhead of error correction makes the real limit 2-3 orders of magnitude higher [33].

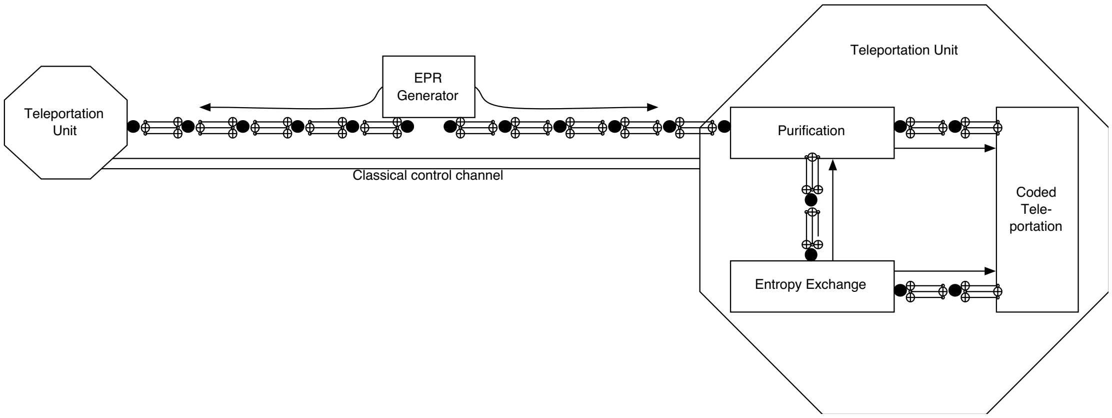


Figure 8. Architecture for a Quantum Wire: Solid double lines represent classical communication channels, while chained links represented a quantum swapping channel. Single lines depict the direction in which the swapping channel is being used for transport.

5 Long Wires

In this section, we introduce an architecture for long quantum wires, shown in Figure 8. These wires make use of the quantum primitive of teleportation. Teleportation involves pre-communication of EPR pairs, followed by a combination of quantum measurement and classical communication to destroy a quantum state at one end of a wire and re-create it on the other end. The key is that the pre-communication can be pipelined. Furthermore, teleportation allows quantum wires to convert quantum data between components that use different error correction codes, a conversion that is impractical without teleportation. In the next few sections, we provide a brief introduction to the core architectural components of this wire.

5.1 Basic Building Blocks

Although teleportation and the mechanisms described in this section are known in the literature, what has been missing is the identification and analysis of which mechanisms form fundamental building blocks of a realistic system. In this section, we highlight three important architectural building blocks: the *entropy exchange unit*, the *EPR generator*, and the *purification unit*. Note that the description of theses blocks is quasi-classical in that it involves input and output ports. Keep in mind, however, that all operations (except measurement) are inherently reversible, and the specification of input and output ports merely provides a convention for understanding the forward direction of computation.

5.1.1 Entropy exchange unit

The physics of quantum computation requires that operations be reversible and conserve energy. The initial state of the system, however, must be created somehow. We need to be able to create zero states, denoted as "$|0\rangle$". Furthermore, errors cause qubits to become randomized; stated equivalently, entropy enters the system through decoherence caused by coupling with the external environment.

Where do these zero states come from? The process can be viewed as one of thermodynamic cooling. Distributed throughout a quantum processor are "cool" quantum bits in a nearly zero state. These can be created by pulling spin-polarized electrons (created, for example, using a standard technique known as optical pumping [23] [49] or directly using spintronics methods, with ferromagnetic materials and spin filters [23]) over the phosphorus atoms.

To arbitrarily increase this probability (and make an extremely cold zero state) we can use a variant of the purification technique described in Section 5.1.3. Specifically, we employ an efficient algorithm for data compression [38] [39] that gathers entropy across a number of qubits into a small subset of highly random qubits. As a result, the remaining quantum bits are reinitialized to the desired pure zero state $|0\rangle$.

5.1.2 EPR Generator

Constructing an EPR pair of quantum bits is straightforward. We start with two $|0\rangle$ state bits from our entropy exchange unit. A Hadamard gate is applied to the first of these quantum bits. We then take this transformed quantum bit that is in a half-way superposition of a zero and a one state and use it as the control bit for a controlled-NOT gate. The target bit that is to be inverted is the other fresh $|0\rangle$ quantum bit from the entropy exchange unit. A controlled-NOT gate is a bit like a classical inverter except the target bit is inverted if the control bit is in the $|1\rangle$ state. Using a control bit of $(|0\rangle + |1\rangle)/\sqrt{2}$ and a target bit of $|0\rangle$ we end up with a two bit entangled state of $(|00\rangle + |11\rangle)/\sqrt{2}$. The quantum bits in this state are called an EPR pair.

The overall process of EPR generation is depicted in Figure 9. Schematically the EPR generator has a single

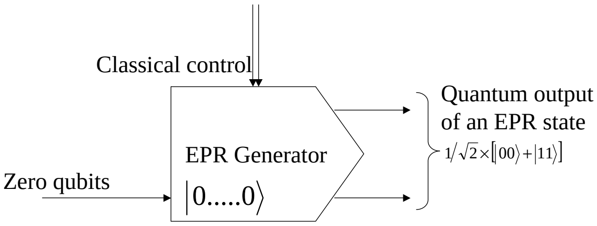


Figure 9. Quantum EPR generator: Solid double lines represent classical communication (or control), while single lines depict quantum wires.

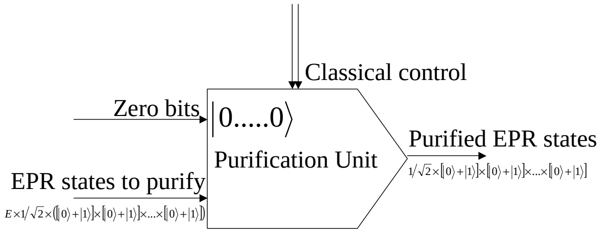

Classical control
Zero bits
$|0.....0\rangle$
Purified EPR states
$1/\sqrt{2}\times[|0\rangle+|1\rangle]\times[|0\rangle+|1\rangle]\times...\times[|0\rangle+|1\rangle]$
Purification Unit
EPR states to purify
$E\times1/\sqrt{2}\times([|0\rangle+|1\rangle]\times[|0\rangle+|1\rangle]\times...\times[|0\rangle+|1\rangle])$

Figure 10. Quantum purification unit: EPR States are sufficiently regular that they can be purified at the ends of a teleportation channel.

quantum input and two quantum outputs. The input is directly piped from the entropy exchange unit and the output is the entangled EPR pair.

5.1.3 Purification unit

The final building block we require is the purification unit. This unit takes as input n EPR pairs which have been partially corrupted by errors, and outputs nE asymptotically perfect EPR pairs. E is the entropy of entanglement, a measure of the number of quantum errors which the pairs suffered. The details of this entanglement purification procedure are beyond the scope of this paper but the interested reader can see [10, 5, 8].

Figure 10 depicts a purification block. The quantum inputs to this block are the input EPR states and a supply of $|0\rangle$ bits. The outputs are pure EPR states. Note that the block is carefully designed to correct only up to a certain number of errors; if more errors than this threshold occur, then the unit fails with increasing probability.

5.2 Analysis

Figure 8 illustrates how we use these basic building blocks and protocols for constructing a long wire. The EPR generator is placed in the middle of the wire and “pumps” entangled quantum bits to each end (via a pipelined swapping channel). These bits are then purified such that only the error-free qubits remain. Purification and teleportation consume zero-state qubits that are supplied by the entropy exchange unit. Finally, the coded-teleportation unit transmits quantum data from one end of the wire to the other using the protocol described in Section 2.3. Our goal now is to analyze this architecture and derive its bandwidth and latency characteristics.

The bandwidth is proportional to the speed with which reliable EPR pairs are communicated. Since we are communicating unreliable pairs we must purify them, so the efficiency of the purification process must be taken into account. Purification has an efficiency roughly proportional to the fidelity of the incoming, unpurified qubits [38]:

$$\text{purification}_{\text{efficiency}} \approx \text{fidelity}^2 \tag{8}$$

Entropy exchange is a sufficiently parallel process that we assume enough zero qubits can always be supplied. Therefore, the overall bandwidth of this long quantum wire is:

$$1/1\ \mu\text{s} \times e^{-2\times10^{-6}\times\text{distance}/60\ \text{nm}} \tag{9}$$

which for a $1\ \mu\text{m}$ wire is 999,967 qbps. Note this result is less than for the simple wiring scheme, but the decoherence introduced on the logical quantum bits is only $O(e^{-\lambda\times10})$. It is this latter number that does not change with wire length which makes an important difference. In the previous short-wire scheme we could not make a wire longer than $6\mu\text{m}$. Here we can make a wire of nearly arbitrary length. For example a wire that is 10 mm long has a bandwidth of 716,531 qbps, while a simple wire has an effective bandwidth of zero at this length (for computational purposes).

The situation is even better when we consider latency. Unlike the simple wire, the wire architecture we propose allows for the pre-communication of EPR pairs at the sustainable bandwidth of the wire. These pre-communicated EPR pairs can then be used for transmission with a constant latency. This latency is roughly the time it takes to perform teleportation, or about $\approx 20\ \mu\text{s}$. Note this latency is much improved compared to the distance-dependent simple wiring scheme.

5.2.1 Technology Independent Metrics

Using the same constants defined above for the swapping channel, we can generalize our analysis of teleportation channels. The latency is simply:

$$latency \approx 10T \tag{10}$$

The bandwidth is:

$$bandwidth_{true} = \frac{1}{T}e^{-2\lambda D} \tag{11}$$

Unlike the short wire, this bandwidth is *not* constrained by a maximum distance related to the threshold theorem since teleportation is unaffected by distance. The communication of EPR pairs before teleportation, however, can be affected by distance, but at a very slow rate. While purification must discard more corrupted EPR pairs as distance increases, this effect is orders-of-magnitude smaller than direct data transmission over short wires and is not a factor in an practical silicon of up to 10's of millimeters on a side.

6 System Bandwidth

Our goal has been to design a reliable, scalable quantum communication layer that will support higher-level quantum error correction and algorithms functioning on top of this layer. A full description of error correction and quantum algorithms is beyond the scope of this paper. A key issue for future evaluation, however, is that the lower latency of our teleportation channel actually translates to even higher bandwidth when the upper layers of a quantum computation are considered. It is for this reason that long wires should not be constructed from chained swapping-channels and quantum "repeaters".

The intuition behind this phenomenon is as follows. Quantum computations are less reliable than any computation technology that we are accustomed to. In fact, quantum error correction consumes an enormous amount of overhead both in terms of redundant qubits and time spent correcting errors. This overhead is so large that the reliability of a computation must be tailored specifically to the run length of an algorithm. The key is that, the longer a computation runs, the stronger the error correction needed to allow the data to survive to the end of the computation. The stronger the error correction, the more bandwidth consumed transporting redundant qubits. Thus, lower latency on each quantum wire translates directly into greater effective bandwidth of logical quantum bits. For more information on quantum error correction and algorithms, we refer the reader to [31].

7 Conclusion

Our study has focused on a critical aspect of any quantum computing architecture, quantum wires to transport quantum data. Building upon key pieces of quantum technology, we have provided an end-to-end look at a quantum wire architecture. We have shown that our teleportation channel scales with distance and that swapping channels do not. We have also discovered fundamental architectural pressures not previously considered. These pressures arise from the need to co-locate physical phenomena at both the quantum and classical scale. Our analysis indicates that these pressures will force architectures to be sparsely connected, resulting in coarser-grain computational components than generally assumed by previous quantum computing studies. We believe that further architectural studies of this nature will be valuable in identifying the research challenges facing quantum technologies of the future.

Acknowledgements

Thanks to Dean Copsey, John Owens and Matt Farrens for their helpful comments on preliminary material for this paper. This work is supported in part by the DARPA Quantum Information Science and Technology Program, by NSF CAREER grants to Fred Chong, Mark Oskin, and John Kubiatowicz, an NSF NER grant, and by a UC Davis Chancellor's Fellowship to Fred Chong.

References

[1] L. Adleman. Toward a mathematical theory of self-assembly. USC Tech Report, 2000.

[2] D. Aharonov and M. Ben-Or. Fault tolerant computation with constant error. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 176–188, 1997.

[3] E. Anderson, V.Boegli, M. Schattenburg, D. Kern, and H. Smith. Metrology of electron beam lithography systems using holographically produced reference samples. *J. Vac. Sci. Technol.*, B-9, 1991.

[4] J. S. Bell. On the Einstein-Podolsy-Rosen paradox. *Physics*, 1:195–200, 1964. Reprinted in J. S. Bell, *Speakable and Unspeakable in Quantum Mechanics*, Cambridge University Press, Cambridge, 1987.

[5] C. H. Bennett, H. J. Bernstein, S. Popescu, and B. Schumacher. Concentrating partial entanglement by local operations. *Phys. Rev. A*, 53(4):2046–2052, 1996. arXive e-print quant-ph/9511030.

[6] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing*, pages 175–179, 1984.

[7] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. Wootters. Teleporting an unknown quantum state via dual classical and EPR channels. *Phys. Rev. Lett.*, 70:1895–1899, 1993.

[8] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters. Purification of noisy entanglement and faithful teleportation via noisy channels. *Phys. Rev. Lett.*, 76:722, 1996. arXive e-print quant-ph/9511027.

[9] C. H. Bennett and D. P. DiVincenzo. Quantum information and computation. *Nature*, 404:247–55, 2000.

[10] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters. Mixed state entanglement and quantum error correction. *Phys. Rev. A*, 54:3824, 1996. arXive e-print quant-ph/9604024.

[11] A. M. Childs, E. Farhi, and J. Preskill. Robustness of adiabatic quantum computation. *Phys. Rev. A*, (65), 2002.

[12] I. L. Chuang. Quantum algorithm for clock synchronization. *Phys. Rev. Lett.*, 85:2006, Aug 2000.

[13] D. P. DiVincenzo. Quantum computation. *Science*, 270(5234):255, 1995. arXive e-print quant-ph/9503016.

[14] D. K. Ferry and S. M. Goodnick. *Transport in Nanostructures*. Cambridge Studies in Semiconductor Physics & Microelectronic Engineering, 6. Cambridge University Press, Cambridge, 1997.

[15] N. Gershenfeld and I. Chuang. Quantum computing with molecules. *Scientific American*, June 1998.

[16] A. Globus, D. Bailey, J. Han, R. Jaffe, C. Levit, R. Merkle, and D. Srivastava. Nasa applications of molecular nanotechnology. *Journal of the British Interplanetary Society*, 51, 1998.

[17] J. M. Goodkind. Proposed fabrication of a quantum computer using electrons on helium. In *Second Annual SQuInT Workshop*, 2000. Poster Abstract.

[18] L. Grover. In *Proc. 28th Annual ACM Symposium on the Theory of Computation*, pages 212–219, New York, 1996. ACM Press.

[19] S. Hallgren. *Quantum Information Processing '02 Workshop*, 2002.

[20] P. Hoyer. *Banff workshop on quantum algorithms*, 2002.

[21] R. Jozsa, D. Abrams, J. Dowling, and C. Williams. Quantum atomic clock synchronization based on shared prior entanglement. *Phys. Rev. Lett.*, pages 2010–2013, August 2000.

[22] B. Kane. A silicon-based nuclear spin quantum computer. *Nature*, 393:133–137, 1998.

[23] B. E. Kane, N. S. McAlpine, A. S. Dzurak, R. G. Clark, G. J. Milburn, H. B. Sun, and H. Wiseman. Single spin measurement using single electron transistors to probe two electron systems. *arXive e-print cond-mat/9903371*, 1999. Submitted to Phys. Rev. B.

[24] D. Kielpinsky, C. Monroe, and D. Wineland. Architecture for a large-scale ion trap quantum computer. *Nature*, 417:709, 2002.

[25] E. Knill, R. Laflamme, R. Martinez, and C.-H. Tseng. A cat-state benchmark on a seven bit quantum computer. *arXive e-print quant-ph/9908051*, 1999.

[26] E. Knill, R. Laflamme, and W. H. Zurek. Resilient quantum computation. *Science*, 279(5349):342–345, 1998. arXive e-print quant-ph/9702058.

[27] K. K. Likhareve. Single-eletron devices and their applications. *Proceedings of the IEEE*, 87, 1999.

[28] S. Lloyd. Quantum-mechanical computers. *Scientific American*, 273(4):44, Oct. 1995.

[29] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland. Demonstration of a fundamental quantum logic gate. *Phys. Rev. Lett.*, 75:4714, 1995.

[30] Y. Nakamura, Y. A. Pashkin, and J. S. Tsai. Coherent control of macroscopic quantum states in a single-cooper-pair box. *Nature*, 398:786–788, 1999.

[31] M. Nielsen and I. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge, England, 2000.

[32] M. T. Niemier and P. M. Kogge. Exploring and exploiting wire-level pipelining in emerging technologies. In *International Symposium on Computer Architecture*, 2001.

[33] M. Oskin, F. Chong, and I. Chuang. Overhead reduction in a architecture for quantum computers. *IEEE Computer*, 35(1):79–87, 2002.

[34] J. Preskill. Fault-tolerant quantum computation. In H.-K. Lo, T. Spiller, and S. Popescu, editors, *Quantum information and computation*. World Scientific, Singapore, 1998.

[35] C. Sackett, D. Kielpinsky, B. King, C. Langer, V. Meyer, C. Myatt, M. Rowe, Q. Turchette, W. Itano, D. Wineland, and C. Monroe. Experimental entanglement of four particles. *Nature*, 404:256–258, 2000.

[36] M. Sanie, M. Cote, P. Hurat, and V. Malhotra. Practical application of full-feature alternating phase-shifting technology for a phase-aware standard-cell design flow. 2001.

[37] L. Schulman and U. Vazirani. Molecular scale heat engines and scalable quantum computation. In *31st STOC*, 1999.

[38] L. J. Schulman and U. Vazirani. Scalable NMR quantum computation. *arXive e-print quant-ph/9804060*, 1998.

[39] L. J. Schulman and U. Vazirani. Molecular scale heat engines and scalable quantum computation. *Proc. 31st Ann. ACM Symp. on Theory of Computing (STOC '99)*, pages 322–329, 1999.

[40] P. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. 35th Annual Symposium on Foundations of Computer Science*, page 124, Los Alamitos, CA, 1994. IEEE Press.

[41] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comp.*, 26(5):1484–1509, 1997.

[42] A. Skinner et al. Hydrogenic spin quantum computing in silicon: a digital approach. *quant-ph/0206159*, 2002.

[43] A. Steane. Error correcting codes in quantum theory. *Phys. Rev. Lett.*, 77, 1996.

[44] J. R. Tucker and T.-C. Shen. Can single-electron integrated circuits and quantum computers be fabricated in silicon? *International Journal of Circuit Theory and Applications*, 28:553–562, 2000.

[45] Q. A. Turchette, C. J. Hood, W. Lange, H. Mabuchi, and H. J. Kimble. Measurement of conditional phase shifts for quantum logic. *Phys. Rev. Lett.*, 75:4710, 1995.

[46] W. van Dam and G. Seroussi. Efficient quantum algorithms for estimating gauss sums. *quant-ph*, page 0207131, 2002.

[47] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, R. Cleve, and I. L. Chuang. Experimental realization of order-finding with a quantum computer. *Phys. Rev. Lett.*, December 15, 2000.

[48] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, R. Cleve, and I. L. Chuang. Experimental realization of order-finding with a quantum computer. *Phys. Rev. Lett.*, to appear, 2000.

[49] A. S. Verhulst, O. Liivak, M. H. Sherwood, Hans-Martin-Vieth, and I. L. Chuang. Non-thermal nuclear magnetic resonance quantum computing using hyperpolarized xenon. *Applied Physics Letters*, 79-15, 2001.

[50] D. Vion, A. Aassime, A. Cottet, P. Joyez, H. Pothier, C. Urbina, D. Esteve, and M. H. Devoret. Maniplating the quantum state of an electrical circuit. *Science*, 296:886, 2002.

A.4 ISCA'05 Paper

An Evaluation Framework and Instruction Set Architecture for Ion-Trap based Quantum Micro-architectures.

Steven Balensiefer, Lucas Kregor-Stickles, and Mark Oskin
Department of Computer Science and Engineering
University of Washington
{alaska, lucasks, oskin}@cs.washington.edu

Abstract:

The theoretical study of quantum computation has yielded efficient algorithms for some traditionally hard problems. Correspondingly, experimental work on the underlying physical implementation technology has progressed steadily. However, almost no work has yet been done which explores the architecture design space of large scale quantum computing systems. In this paper, we present a set of tools that enable the quantitative evaluation of architectures for quantum computers.

The infrastructure we created comprises a complete compilation and simulation system for computers containing thousands of quantum bits. We begin by compiling complete algorithms into a quantum instruction set. This ISA enables the simple manipulation of quantum state. Another tool we developed automatically transforms quantum software into an equivalent, fault-tolerant version required to operate on real quantum devices. Next, our infrastructure transforms the ISA into a set of low-level micro architecture specific control operations. In the future, these operations can be used to directly control a quantum computer. For now, our simulation framework quickly uses them to determine the reliability of the application for the target micro architecture.

Finally, we propose a simple, regular architecture for ion-trap based quantum computers. Using our software infrastructure, we evaluate the design trade offs of this micro architecture.

1 Introduction

Experimental research into quantum computing technologies has been progressing at a steadily. Demonstrations of bulk-spin NMR computers [1], ion-trap based designs [2, 3, 4], and optical cavity wells [5, 6] for quantum computation have been performed. The next step in this area is to scale up from experimental quantum computers consisting of a handful of quantum bits to large scale quantum computing systems. Clearly many technological hurdles still exist, and one of the most basic is the architectural design of these systems.

Why worry about the architecture of a quantum computer now? The most promising technologies are at least five years from demonstrations of a dozen qubits or more, and large scale systems are not even seriously on the drawing board. Architects, however, can make significant contributions by: (1) identifying the serious practical difficulties that will arise from the physical structure of these devices and (2) finding solutions to these and other challenges with the technology.

Identifying the challenges these systems face allows device physicists and quantum theorists to start exploring potential solutions. By understanding the challenges facing the practical implementation of these technologies, architects can find solutions through the proper organization of the structure of these devices. Collectively, what this means is computer architects have the potential to hasten the development of a large scale quantum computer sooner rather than later by identifying and solving scalability problems early.

Where to begin with quantum architecture research? Similar to classical architecture, one begins with the applications. Surprisingly, even though it will be some time before a quantum computer is built the application that computer will execute is already well known: *error correction*. Quantum technologies will operate with error rates far higher than classical machines. Experimental error rates of $1E{-}3$ per bit operation have been measured in NMR systems [1]. Technological advances are expected to lower these rates dramatically, but reaching $1E{-}10$ - $1E{-}12$ is considered highly aggressive. There is only one way to manage these errors in a quantum computer: utilizing software error correction on a well-designed quantum computer architecture.

Our research efforts have been devoted to developing these architectures. To conduct this research in a quantitative fashion, we developed an infrastructure consisting of compilation and modeling tools. This paper will spend significant time describing these software artifacts (Sections 3- 7) because the methodology for applying architectural principles to quantum computers is one of the primary contributions of this work. All existing work on quantum architectures has produced either hand-designed circuits without considerations for scalability [7] or analytical models for performance and reliability that are unable to scale to systems large enough to solve real-world problems [8].

We rely on appropriate technological abstractions and careful design of the ISA, scheduler, and simulator to construct an infrastructure that scales (linearly) to thousands of qubits and billions of time steps. Briefly, our tool chain is the following:

- A compiler from an existing high-level language which enables the manipulation of quantum bits to an instruction set architecture we developed for quantum computers.

- An error correction compiler that automatically transforms a quantum ISA assembly text into equivalent fault tolerant versions.

- A device scheduler that maps an assembly source into a set of device specific primitive operations for controlling a quantum micro-architecture.

- A simulator that models the reliability of the quantum bits in a quantum computer; performance and reliability metrics for an application running on a targeted microarchitecture can be obtained by using this simulator.

Our tools enable researchers to explore architectural tradeoffs directly. Instead of using high-level models or mathematical equations to calculate execution time and reliability our infrastructure provides the proper compilation, scheduling, and simulation tools to compute these results precisely.

Using these tools, in Section 8 we evaluate a few quantum micro-architectures as they perform error correction steps. We find that the realistic constraints exposed by execution on a microarchitecture significantly decrease the acceptable error rates. Idealized theoretical models set the critical threshold – above which sustainable quantum computation is not possible [9] – at approximately $1E{-}4$, but a threshold which accounts for the constraints of the proposed micro-architecture is closer to $1E{-}9$. Our results indicate that more than 4/5 of this difference can be accounted for by resource contention and the impact of ion movement and turning in a real system. Since architects excel at the exploitation of locality and the minimization of resource contention, this suggests that through intelligent design, architects have the potential to have a major impact on the accuracy of quantum computation thus allowing us to achieve a scalable quantum computer sooner rather than later.

The remainder of this paper is structured in a logical progression. In Section 2 we describe the abstractions we use to make quantum architectures accessible. Section 3 presents an overview of our software infrastructures. The ISA we developed is described in Section 4. Sections 6, 5 and 7 elaborate on the design of our device scheduler, error correction compiler, and simulator. In Section 8 we present the result from our exploration of a simple tile-based quantum microarchitecture. In Section 9 we describe where to go next with this work and in Section 10 conclude.

2 Technology abstraction

The science of architecture is the optimization of the hardware / software interface. The nuts and bolts of it is examining applications, working with the realistic constraints of the technology, and developing software infrastructures and hardware designs. Research into architectures for quantum computers is no different. To design architectures, a reasonable abstraction for the underlying technology and understanding of the software applications is required. In this section, we describe a basic set of abstractions for ion trap based quantum computing technology. We will discuss the application characteristics further in Section 4.

We focus our attention on ion trap based designs because they appear to be the most promising in terms of a near term ability to deliver a system with 10's to 100's of qubits. The cost of these systems will not be insignificant with estimates in the hundreds of millions of dollars to develop a single prototype. Proper engineering of their architectural design ahead of time will be required to maximize their scientific and national infrastructure value.

For architectural design, we focus on three circuit components: ions, traps and wires, as depicted in Figure 1. Ions are the entities that realize qubits. The excitation state of the outer electron on a $^{9}\mathrm{BE}+$ ion is the actual quantum property used to realize a qubit [3, 4]. A trap is a device that uses classical support circuitry and lasers to perform quantum operations on ions. This gives it a multi-purpose, ALU-like functionality. Quantum operations can only be performed on ions that are located in traps. Inside of the trap, any arbitrary single qubit operation and a limited number of two qubit operations including CNOT and controlled rotation can be performed. For the two qubit operations, both ions must be located in the same trap. Wires are just two sided structures within the design in which ions can move. Wires can contain corners but care must be taken when moving ions in anything other than a straight line. Ions must move adiabatically (read: slowly) around corners or an unrepairable amount of noise will be introduced.

While the precise timing of all operations is obviously not known yet – it is technology specific and will change as the systems evolve, the relative timing between them, observed from [3, 4], is roughly: moving 1 unit within a wire is $1/10^{th}$ a time step; performing a single qubit operation, 1 time step; performing a two qubit operation, 10 time steps; turning a corner including getting into and out of a trap, 100 time steps. Architects should think of the single-qubit operations as the "clock cycle" of the machine. The classical analogy is that these operations are simple and fast, like an addition. Measurement and two-qubit operations are slow, just like complex classical functions such as divide. Later in Section 5 we will present statistics for the relative instruction mix between single/two qubit operations and measurement.

The basis unit for these time steps is $\approx 1us$. For single- and two-qubit operations, this will not change, as it is a fundamental property of the ions [3, 4] used to realize qubits. For movement and turning, it is a function of the technology, and as this develops, they may become faster. Moving,

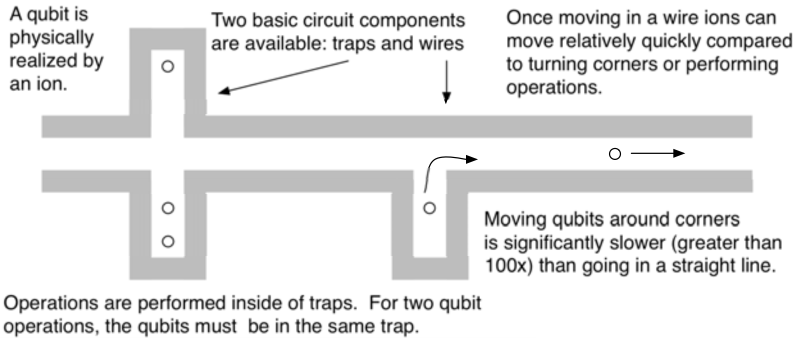

A qubit is physically realized by an ion.
Two basic circuit components are available: traps and wires
Once moving in a wire ions can move relatively quickly compared to turning corners or performing operations.
Moving qubits around corners is significantly slower (greater than 100x) than going in a straight line.
Operations are performed inside of traps. For two qubit operations, the qubits must be in the same trap.

Figure 1: Technological abstraction of ion trap based quantum computers.

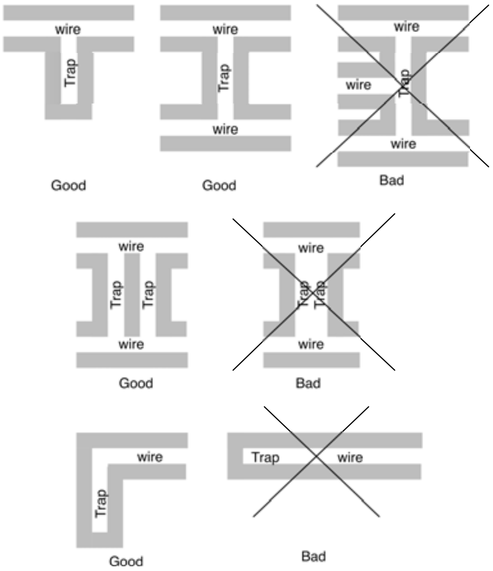


Figure 2: Basic design rules for ion trap systems.

and in particularly turning, induces noise (from heat) on the qubit and must be performed slowly so that the state does not decohere. Current experimental work aims for turning to be 50-300 times slower than single-qubit operations [3]. Since controlling noise is so important for quantum architectures we do not use a highly aggressive turning time for our simulations. We do, however, explore the impact of this parameter on performance in Section 8.

Our review of current ion-trap based designs [3, 7] suggests a few simple design rules that must be observed by architects. These rules are the quantum analog of VLSI design rules:

- Ion traps may only abut one or two wires
- Ion traps may not share any sides
- Ion traps may not abut the end of a wire

These rules are depicted in Figure 2. They serve as an additional level of abstraction by removing the need to consider the exact sizing and space tolerances for layouts. Later, in Section 8, we will explore a simple regular architecture that observes these design constraints.

3 Software overview

To evaluate complex conventional systems, architects utilize a variety of software tools. Starting with a (hopefully) representative set of applications, they compile and execute them on sophisticated simulation infrastructures that model different points in the design space. To properly study large scale, quantum computers we created a corresponding infrastructure. This infrastructure is comprised of four major components: a source compiler, an error correction compiler, a device scheduler, and a simulator. In this section, we describe what these tools do and how they are used. In the next few sections we elaborate more on how they work. Figure 3 contains a pictorial overview of the flow of information through the tools.

Source compiler: To describe quantum algorithms, we utilize the existing QCL [10] work. The QCL toolkit provides an interpreter for a fairly straightforward imperative programming language that includes data types and operation primitives for quantum operations. We did not extend this work significantly except to make minor changes to perform loop unrolling and output instructions in the instruction set described in Section 4.

To allow for aggressive code optimization we require the input to applications at compile time. The resulting assembly output from the compiler contains only the operations required to perform the algorithm on the provided input data. This may seem limiting, but two related reasons motivate this design choice. First, our expectation is that the time required for a quantum computer to execute an algorithm will be significantly longer than the time required to optimize resource usage for a particular algorithm/input dataset combination.

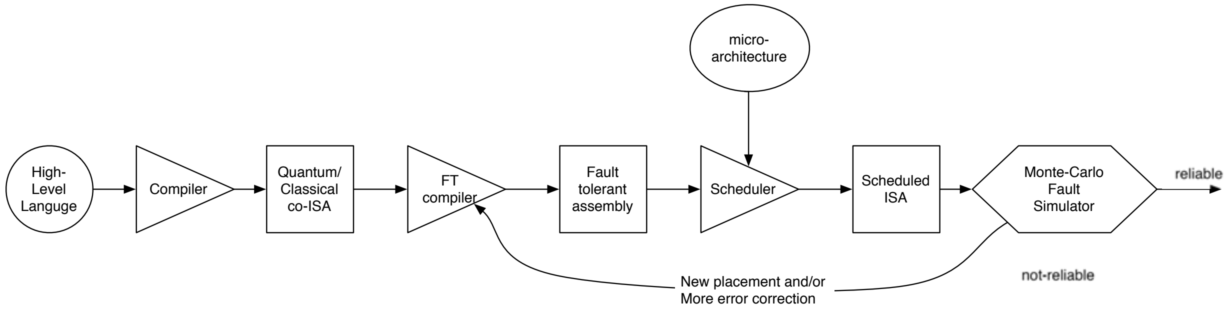


Figure 3: Quantum architecture research infrastructure. This set of tools enables architects to start with a high level language description of an algorithm and a microarchitecture and compile, add fault tolerant steps, schedule for the architecture, and simulate the speed and reliability of the algorithm.

Stated another way, there will be sufficient gains in execution time to spend significant time "up front" optimizing resource usage. The second reason is that error correction incurs a high overhead, suggesting that it should be applied as minimally as possible. More general representations require more general computation and hence more error correction, while an executable targeted to only a single input dataset can be optimized aggressively for just that dataset. Similar findings have been reported in classical computing, where dynamic optimizers aggressively tailor executables, folding in constants, etc [11, 12, 13].

Error correction compiler: The output of our source compiler is an assembly text. This assembly assumes an idealized machine – one with no errors. This is not true at all – quantum computers will have error rates between $1E{-}6$ and $1E{-}10$ per operation. To counteract this, researchers discovered and explored many different types of quantum error correction [14, 15, 16, 17]. For our purpose, we selected the 7 qubit Steane code [14] and the recursive construction process described in [9]. The error correction compiler inputs the assembly text that assumed an ideal computer and an "error correction strength" level and outputs another assembly text that is the same algorithm except with fault tolerant constructs included. This output text is considerably larger – potentially by several orders of magnitude – but is required to coax the right answer from an otherwise noisy quantum device.

Scheduler: The next step in the tool chain is to schedule the resources of the quantum computer. For classical computing devices, the schedule is implicit in the executable – the semantics of von Neumann machines are sequential. For quantum computers, sequential semantics are maintained, but the importance of exploiting parallelism increases dramatically. Ignoring parallelism in a von Neumann machine results in a longer execution time, but the computed result does not change. In a quantum computer, ignoring parallelism could result in a wrong answer. Thus our scheduler takes in an assembly text and a description of the microarchitecture of the quantum computer and creates a parallel schedule of operations that should be performed on the actual microarchitecture.

Simulator: Once the application is scheduled onto the physical resources of the machine, the next step in the tool chain is to decide whether or not the application will actually work. Too little error correction or a poor schedule will produce noise instead of the correct answer. The purpose of this step is to determine how reliable the scheduled application will be on the device. If the simulator determines the schedule will be reliable then we are done. The end results are two facts: how fast the algorithm executed on the microarchitecture and how reliable the result was. If the result is determined to be unreliable, the user has to back up two steps and add more error correction or model a different microarchitecture that might perform better.

A schedule that shows a high rate of reliability under simulation is detailed enough to control the physical computer during the execution of the algorithm and dataset. This step is beyond the scope of this work, but basically, it involves translating the schedule using a fairly straightforward mapping between operation steps and the pulses that control the actual quantum computer.

4 Instruction set architecture

The design of an instruction set architecture (ISA) encompasses many different pieces. The most fundamental is the execution model, which describes how a machine will process a group of instructions. Next are the resources available in the machine, typically memories and their interface. Finally, there are the actual instructions themselves. Figure 4 describes the ISA we designed.

The ISA we describe here is a "high-level ISA" which is not directly executable by any quantum computer. These ISAs have also been referred to as "virtual ISAs" [18] and linear intermediate representations. The purpose of this ISA is to provide a workable representation of an application. By workable, we mean that it has relatively straightforward semantics, tools can process it largely piecemeal operation-by-operation, and it can be translated in a direct way to the actual control sequences a quantum computer requires.

Execution model: We base our ISA on the von Neumann execution model. This means that conceptually, the quantum computer can be thought to fetch, decode, and execute the primitive operations one-by-one. This design choice is motivated by an additional restriction we place on execution: Quantum programs may not contain branches. All loops must be fully unrolled, and all conditionals must be converted into predicates. (see Figure 5).

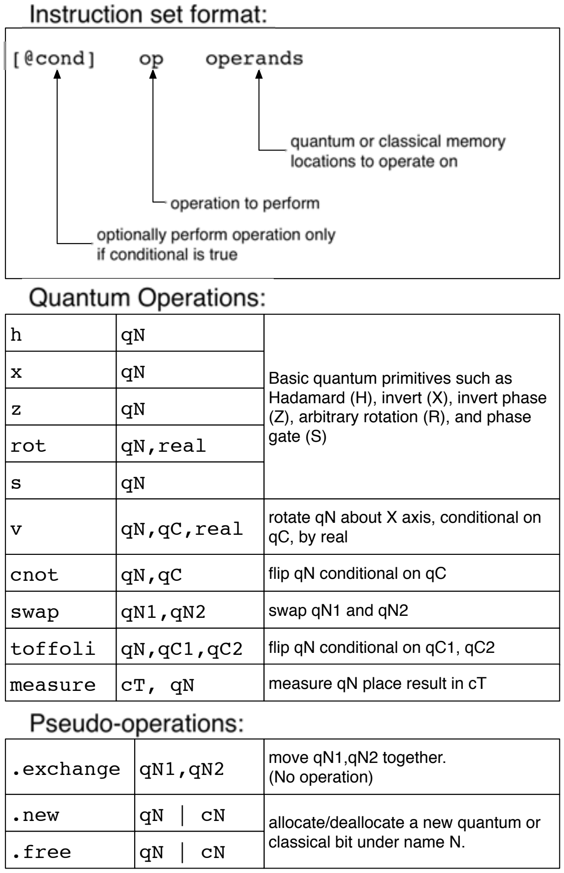
The reason we chose to restrict applications in this way is that it enables our software infrastructure to provide developers with concrete reliability results. Since there are no branches in the compiled binary, every instruction must be scheduled onto a quantum micro-architecture. Once scheduled, our simulator can provide a very precise answer to the question, "Will it work?"

If branches were part of the ISA, then answering that question would no longer be possible. The schedule of low-level operations could vary significantly from execution-run to execution-run based upon branch outcomes, which in quantum software, depends largely on random noise the system experiences and corrects for. These variances make it far more difficult to predict reliably whether or not the schedule will actually compute correctly.

Resources: In our high-level ISA, we assume an infinite number of quantum and classical memory locations are available. Memory is split into two segments, a quantum segment and a classical segment. Quantum bits (qubits) are referred to as $qName1, qName2, ...$, while classical bits are referred to as $cName1, cName2, ...$. Since this is a high-level ISA, there is no need to restrict the name of bits to simple numerical addresses as a simple compilation pass prior to scheduling can assign device specific addresses and resolve any false dependencies caused by name reuse. We do not use a hierarchical memory (i.e. there is no distinction between memory and registers).

Operations: The instruction set we have devised operates on both classical and quantum data. The classical operations are fairly ordinary and encompass a straightforward set of opcodes (logic, arithmetic, etc). For brevity, we do not describe them in detail because they are your typical three operand RISC-like ISA: $cOutput = cInput1 \; op \; cInput2$.

The quantum opcodes are summarized in Figure 4. These operations provide a fairly basic, yet complete set of operations for manipulating quantum state. There are many things to note about this instruction set. First, all quantum operations (except measurement) are, by definition, re-


Instruction set format:

```
[@cond]   op   operands
```

quantum or classical memory locations to operate on
operation to perform
optionally perform operation only if conditional is true

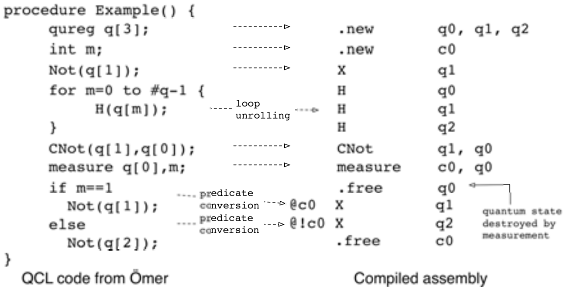
Quantum Operations:

h	qN	Basic quantum primitives such as Hadamard (H), invert (X), invert phase (Z), arbitrary rotation (R), and phase gate (S)
x	qN	
z	qN	
rot	qN,real	
s	qN	
v	qN,qC,real	rotate qN about X axis, conditional on qC, by real
cnot	qN,qC	flip qN conditional on qC
swap	qN1,qN2	swap qN1 and qN2
toffoli	qN,qC1,qC2	flip qN conditional on qC1, qC2
measure	cT, qN	measure qN place result in cT

Pseudo-operations:

.exchange	qN1,qN2	move qN1,qN2 together. (No operation)
.new	qN | cN	allocate/deallocate a new quantum or classical bit under name N.
.free	qN | cN	

Figure 4: The instruction set architecture for quantum computers


```
procedure Example() {
    qureg q[3];
    int m;
    Not(q[1]);
    for m=0 to #q-1 {
        H(q[m]);
    }
    CNot(q[1],q[0]);
    measure q[0],m;
    if m==1
      Not(q[1]);
    else
      Not(q[2]);
}
```

```
        .new       q0, q1, q2
        .new       c0
        X          q1
        H          q0
        H          q1
        H          q2
        CNot       q1, q0
        measure    c0, q0
        .free      q0
@c0     X          q1
@!c0    X          q2
        .free      c0
```

Figure 5: The compiler, based on QCL [10], transforms QCL source text into assembly.

versible. This is a constraint imposed by the quantum computing model itself. One implication of this is there is no distinction between input and output operands. Instead, operations transform all of their operands. Second, this ISA is a balance between high-level primitives, such as multi-qubit complex operations, and low-level device-specific controls. Our guiding principle has been to design an ISA with primitive enough operations that a clear $1 : N$ mapping exists between the operations and device specific control sequences, but high-level enough that the tool infrastructure could manipulate a useful block of related work.

5 Error correction compiler

After compilation to an assembly text, the next step is to transform the application into a fault-tolerant version. Fault tolerant quantum computing is done in pretty much the same way it has been done for decades in the classical domain – through redundancy. Each quantum bit is encoded into a *logical* qubit. Logical qubits utilize several physical qubits to store a coded version of the quantum state. Each operation on the original qubit is transformed into an equivalent set of operations on the qubits that make up the logical qubit. The major drawback in all of these schemes is the increase in the number of qubits required.

In our system, we utilize the 7-qubit Steane code [14] and employ the recursive error correction constructions described in [9]. We do this with an assembly source to source translator. This translator converts a compiled quantum application into an equivalent assembly source file which contains the embedded error correction operations.

The precise fault tolerant constructions are not a contribution of our work. We base them on prior work [14, 9, 19] and refer the interested reader there. However, to the best of our knowledge, our tool is the first to apply them automatically to an application, accounting for all of the required ancilla preparation work and at multiple strength levels (0,1, and 2 levels of error correction). Because of this, we have calculated some useful statistical properties about its output.

The results are shown in Figure 5. Architects should take note of the overhead in both time and space introduced by the error correction processes. The critical path for a single-qubit operation with one layer of error correction (EC1) is 31 operations long. Only 1 of those is devoted to actually performing the operation on the logical qubit. The rest are devoted to the fault tolerant correction step. More realistically, not all operations will be conducted in parallel, the overhead will be substantially higher, and stronger levels of error correction will be required. This sizable overhead is one of the reasons we can design quantum computing architectures now – Amdahl's Law [20] suggests quantum computers are going to spend all of their time error correcting!

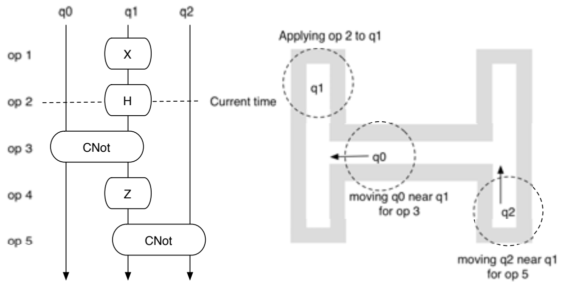


Figure 6: The goal of scheduling is to transform a program source (left) into a sequence of primitive operations that move and manipulate ions in a quantum computer (right). A main requirement is that the scheduler operate in $O(instructions)$ time because the number of instructions is in the billions for a complete fault tolerant run of Shor's algorithm.

6 Device scheduler

Once we have a source assembly file with the error correction compiled in, the next step is produce a schedule for those operations on an ion trap computer micro-architecture. Figure 6 depicts the overall goal: given a source assembly text (represented in graph form on the left), the scheduler produces the parallel sequence of low-level operations (right). In this section, we describe the scheduling process.

6.1 Input

The scheduler takes three pieces inputs: the source assembly text to be scheduled, a description of the architecture to schedule the source on, and a description of the technology parameters and constraints. The source text has been previously described (Section 4).

The architecture description is a low-level description of the ion trap layout. As a classical analogy, this is at the same level as a VLSI layout produced with tools such as Magic [21]. The description includes the precise X/Y coordinates of ion-traps, the operations each trap can perform, and their interconnection wiring.

The technology parameters provided to the scheduler contain timing information for all device-specific operations. This includes the timing of all operations (X, H, CNOT, etc) and the timing for moving ions around the computer. Specific movement parameters are included for moving ions through wires, into and out of wires, and for turning corners.

6.2 Scheduling algorithm

The ability to process billions of operations was paramount in designing the scheduler. Therefore, we chose to trade-off optimality for speed. One of the major costs in scheduling is determining the route an ion should take to travel between traps that do not abut the same wire. This problem has paral-

	% cnot	% measurement	min time	max time	min space	max space
no EC	-	-	1		1	1
EC 1	56.37%	10.31%	31	447	16	96
EC 2	56.74%	9.44%	211	217529	58	12456

Table 1: **Properties of error correction:** In this table, we present results from our error correction compiler and scheduler. The first two columns indicate the percentage of two-qubit CNOT gates and measurement operations. The next two columns min/max time indicate the time (in ops) required to perform the error correction process. Minimum time refers to doing things maximally parallel (no architectural constraints), while maximum refers to doing all operations sequentially. Minimum/max space refers to number of physical qubits required to perform the operation. Minimum space comes from doing all operations sequentially (scheduled perfectly), while maximum space comes from doing as many operations in parallel as possible, reducing execution time but increasing resource requirements.

lels in the routing of signals between logic units within FPGAs, so we adapted the PathFinder algorithm [22] to create a collection of efficient paths from source to destination in the micro-architecture. The biggest change is that we compute 5-10 paths on the first movement between a source-destination pair. This computation only occurs once, and subsequent movements simply pick the best path from those stored.

The next step is to parse the source assembly and schedule the operations. For this we employ a variant of list-scheduling [23]. First, the source text is parsed to completion and a graph representation is produced. We process this graph in reverse order, starting from the leaves, and proceeding to the root(s). By applying an earliest-possible greedy approach in reverse, we approximate latest-possible scheduling if run forward in time. From the view of the simulator, qubits allocated only when absolutely necessary, allowing reuse of “scratch” bits and attempting to minimize the time that qubits must stay coherent.

At any given time point, the scheduler maintains a list of operations that can be scheduled and attempts to allocate the physical resources of the machine for the required number of time steps. In the case of operations, this means simply holding onto the ion trap for that time. For movement, it means referring to the pre-computed path data structure and choosing the path that with no conflicts for the time required. Operations that cannot be scheduled due to resource conflicts are simply delayed and another scheduling attempt is made at the next opportune time step.

7 Simulator

The scheduler produces an exact set of command sequences for controlling a quantum computer. One can directly read the tail end of this schedule to determine the running time of the application. Of critical importance, however, is whether or not the qubits will contain correct values. Noise (decoherence) could have corrupted them so much that the schedule will not produce any meaningful result from a quantum device.

In all other quantum research projects, a precise physical level simulator of the device is used to determine the reliability. In our study, however, we are interested in computers with hundreds to thousands of qubits. Since the running time of precise simulation is exponential in the number of entangled qubits, the number of qubits that can be simulated in reasonable time with current technology (clusters of machines, days of time) is in the low 30’s [24]. Clearly, this approach will not work for 100 - 100,000 qubits.

Instead, we make the observation that if you do not care about simulating the precise state of a quantum computer, Monte Carlo simulation can be used to produce an expected reliability for the device. With Monte Carlo simulation, the expected probability of a phenomenon is determined by performing an action several times and calculating what percentage of the time the phenomenon in question occurs.

In our case, the phenomenon in question is the introduction of error into an ion’s quantum state. To perform our simulation, we start with a base error rate for each step of computation. This base error rate represents the probability that an error occurs in an ion at each time-step. We introduce an error in the ion when our pseudo-random number generator [25] produces a result less than this base error.

Within the simulation, errors are propagated based on the dependencies of the computation. Once an ion is in error, it stays in error and introduces error on any other ions it interacts with. The only exception to this rule is when error correction is applied. Our simulation framework models the effect of the error correction added prior to scheduling. Once an error correction is completed, the simulator examines the qubits of the logical code word. If only one qubit is in error, then the simulator assumes the error correction process fixed that single qubit error. If two or more qubits are in error, it propagates the error and assumes all qubits of that code word are now in error (the upper bound of the effect of error correction on a terminally broken code word).

Naturally, the effectiveness of Monte Carlo simulation depends on the randomness of the pseudo-random number generator used. For this purpose, we have selected a random number generator based on bit-rotation and addition which is considered particularly well suited to Monte Carlo simu-

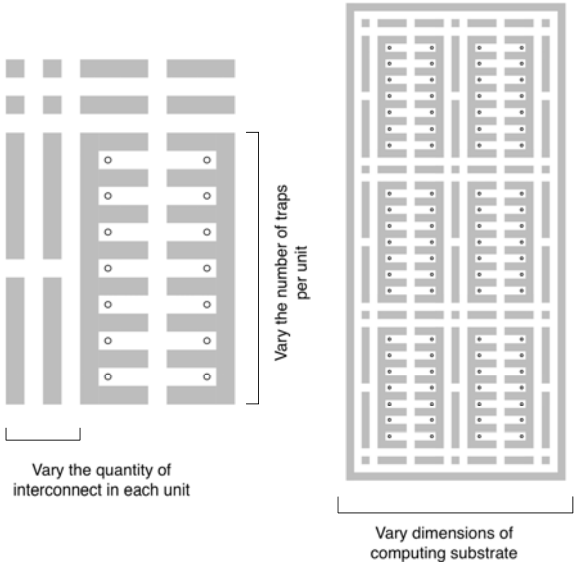

Vary the number of traps per unit
Vary the quantity of interconnect in each unit
Vary dimensions of computing substrate

Figure 7: Basic tile structure (left) and substrate micro architecture (right). The key parameters: ion trap cells, connectivity, and substrate size are varied in this study.

lation [25].

The simulator can be used to quickly ($O(n)$ in the number of scheduled operations) determine whether or not a schedule and micro-architecture will operate accurately. Since the problems targeted by quantum computers are in the complexity class NP (a super-set of NP-complete), despite the fact that solutions to these problems are hard to generate, verification in possible in polynomial time. This means that accuracy need only be around 90% (since incorrect answers can be quickly detected and the process re-run if necessary). The implication is that the reliability of the system can be determined with a mere 100 trials on our simulator.

In addition to a quick test of the reliability of a quantum program and micro-architecture pair, the simulator can also execute an arbitrary number of trials to achieve a fine-grained understanding of the rate of error. In the next section of this paper, we use this technique to explore variations on a canonical quantum micro-architecture and measure their runtime performance and critical thresholds [9].

8 Micro-architecture exploration

In this section, we use our infrastructure to explore basic micro-architectural trade-offs. We begin by first validating the simulation model. Next, we use the tools to explore trap width versus wiring density in a simple quantum micro-architecture. Finally, we conclude by exploring the differences between quantum computing theory and practice, which highlights both the challenges for future technology development, and the importance of architecture to this discipline.

8.1 Validation

Since these tools are the first of their kind and our simulation methodology is a novel approach to modeling reliability in quantum systems, some form of validation is desired. To do this, we produced a single fault-tolerant error correction sequence. This was scheduled onto an architecture and processed by our simulator. The parameters the simulator used to model error were changed such that moving ions around the micro-architecture occurred in zero time, ions that were not being operated on had zero chance of decohering, and CNOT instructions required the same amount of time as single-qubit gates. These parameters match the theoretical model of quantum computing that is used in the literature. Doing this, we found the critical threshold – the maximum error per operation for sustainable fault tolerant computation, to be $4E-4$. This is exactly in line with what one would expect from the theoretical estimate previously calculated [9, 19].

8.2 Exploration

A basic design of a quantum micro-architecture is depicted in Figure 7. The concept is to use a substrate of identical tiles. This design has two basic micro-architectural knobs to vary: the number of ion traps in a tile and the amount of wiring between tiles.

To explore the effects of these two parameters on execution time, we mapped the error-correction (level 1) process onto varying substrates using our scheduler. We chose layouts that provided 150 traps total and organized the tiles to be as square as possible.

The scheduler is non-deterministic (being based on a synthesis of PathFinder and list scheduler), so results vary slightly between runs. Therefore we execute each test 8 times and average the results. The overall results are shown in Figure 8.

The results show three interesting trends. First, except for the smallest design, small numbers of traps per tile are favored. Too few traps and scheduling becomes more difficult. Ions must move into and out of regions too often, increasing execution time. With too many traps, the conflicts over the single wire into the tile begins to counter the increased potential for intra-tile movement. With larger trap numbers, the ions must also move further, leading to longer execution times.

Second, beyond 2 traps / tile, a single surrounding wire (which is 2 wires between ion trap complexes; Figure 7) is less efficient than having more interconnect. However, moving from 2-3 surrounding wires provides no real savings. Looking carefully at each trap complex configuration, there is a corresponding ideal interconnect width: 2 traps / 1 wire, 3 traps / 2 wires, 5 traps / 3 wires. This pairing arises from the schedulers ability to exploit trap resources

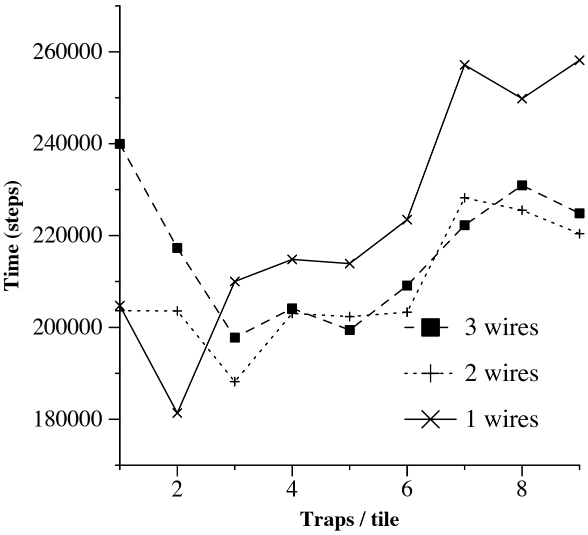


Figure 8: Performance of error-correction on various trap configurations, from designs built from tiles that are 1 trap high and 1 wire in between, to 9 traps and 3 wires.

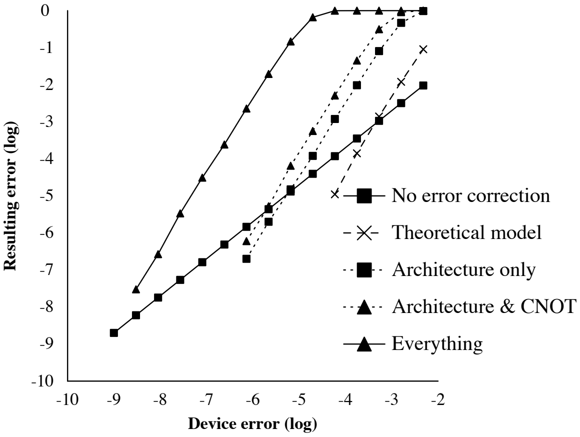


Figure 9: Observed error rates for different technology and architectural assumptions.

and wire-resources, and it makes intuitive sense that the order of these (more traps - more wires) is aligned up until 5 traps. Beyond 5, the resources are not exploited well by the scheduler and simply increase delay.

The final observation is that 2 traps / tile and a surrounding wire per tile performs best on average for this single application. For our scheduler algorithm and the error correction process, this design point minimizes overall length of travel for ions and balances trap complex size against interconnect size.

8.3 Dealing with architectural reality in quantum computers

We conclude our study by examining the critical threshold – the error rate above which error correction processes will not work. In the past [9], theorists have estimated this threshold using an overly idealized model of computation that did not account for the actual microarchitecture of the machine. Using our tools, we can account for this.

Figure 9 plots the reliability of fault-tolerant operations as various technology and architectural features are progressively accounted for. The x-axis of this graph is the rate of error for a single-qubit gate. The y-axis of this graph depicts the rate of error for the qubit measured by our simulator. The straight-line depicts the rate of error for a non-encoded non-fault tolerant single qubit operation. The x-axis points at which the other curves cross this line are their critical-thresholds.

The first line (farthest to the right) is the theoretical quantum computing model. In this model, there is no accounting for architecture or technology implications – such as movement, turns, the difference between single- and two- qubit gates, and the reality that a quantum state naturally decoheres with time, even if no operation is performed on it. This line crosses the non-fault tolerant line where prior literature [9, 19] estimates it should.

The next line over, *architecture only* alters the model to begin to consider the implications of having to perform error correction in a real micro-architecture. For this calculation, the impact of decoherence from having to wait for resources to become available is introduced.

Next comes the *architecture and cnot* line. This line depicts the effects of the micro-architecture and accounts for the fact that CNOT gates require an order of magnitude more time to operate than single qubit gates.

The final result, *everything accounted for*, is one of the main results of our work. In this trial, we introduce the full impact of movement and turns. We found that when operating on an actual micro-architecture and accounting for all of the implications of scheduling, resource conflicts, the cost of moves and turns and single versus two-qubit gates, the true threshold lies at $\approx 1E-9$. This is lower than the theoretical calculation by *5* orders of magnitude.

An important observation from this data is that of these 5 orders of magnitude in difference between the theoretical model and the actual implementation, *3* of these are the result of movement and turning, ≈ 1.5 are the result of basic resource contention and only $\approx 1/2$ is the result of the increased cost of binary operations such as CNOT.

The implication of this is that improving the accuracy of individual quantum operations will only have a minimal impact on the overall accuracy of quantum computation. Instead, our work indicates that physicists should focus on reducing the error rate and improving the execution time for turns, while architects can make a major contribution by designing micro-architectures and schedulers that capitalize on locality to decrease the need for movement and allow for the efficient utilization and placement of resources to decrease contention. In this way, architects can raise the *practical* threshold. Otherwise, it really will be later rather than sooner before quantum computing is a reality.

9 Future work

There is much work left to do on quantum architectures. Right now, and for the foreseeable future, the goal of this work should be to reduce the critical threshold. What we have presented in this paper is a set of tools and architectural analyses that show the real threshold is $\approx 1E - 9$. At least two and perhaps as much as three orders of magnitude of this threshold, however, are due to the micro-architecture and tool chain infrastructure. We will elaborate below on ways to reduce this threshold:

Better error correction processes: Our current infrastructure utilizes the error correction steps described in [19]. More complex, but parallel steps are known [26]. Changing the front end of the tool chain to utilize these alternative constructions could reduce by about 1/3 the minimum-time component in Table 5. This is at the expense of more complex ancilla.

Dynamically adding teleportation-channels: In [27] the authors describe an alternative way to move quantum state around a large micro-architecture. Exploiting these teleportation channels instead of direct movement where appropriate could further parallelize the operations involved in moving quantum state about.

Better micro-architectures: For this paper we did not extensively study micro-architecture designs. Our goal was more on the front end in creating all of the tools required to really study micro-architectures. Thus, the very next step seems to be to design architectures that are better able to exploit parallelism within the error correction processes.

Smarter scheduling: Our current scheduler is essentially a greedy algorithm with a bounded window. Perfect scheduling is NP-hard. There is a middle ground. Right now the scheduler is micro-architecture agnostic. It can schedule any set of quantum algorithms onto any micro-architecture. Making the scheduler more micro-architecture and error-code aware seems a rich area for performance gains. For example, qubits are often operated on in repetitive ways. Having efficient (perhaps hand-done) schedules for these common-case operations that the scheduler could draw upon to create a larger application schedule seems a viable approach.

Hierarchical simulation: Currently, our simulator is pessimistic. It is akin to an automated "counting" simulator (used to count point of failure). If the actual device had the technology characteristics specified it would be more reliable when executing the application. How much more reliable is not yet known, but it is speculated that it is perhaps as much as an order of magnitude. The simulator can be made more precise by integrating a precise device-level physics simulator and grouping operations into large units. These units can be modeled precisely using the device simulator and then their reliability parameters integrated using the counting approach of our existing framework.

10 Conclusion

In this paper, we described our work in designing an instruction set architecture, compiler, device scheduler and simulator for ion trap based quantum computers. Many design choices in each of these components were made to make them scale to real application sizes. Among them: the tools compile-in the input dataset and fully unroll all loops so that the scheduler and simulator can provide concrete results; the error correction compiler automatically transforms arbitrary input programs into fault tolerant versions; the scheduler combines techniques from FPGA/CAD synthesis and traditional processor compilers; finally, the simulator efficiently models errors instead of quantum state in order to quickly provide reliability information.

Using these tools, architects can design and quantitatively evaluate large scale architectures. In the past, quantum researchers have had to make careful analytical models for system reliability and performance. Now, they can evaluate these systems directly by compiling applications for them, scheduling them for performance, and simulating them for reliability. We did this for a few tile based designs and found that a balanced design of 2 traps to 1 interconnect wire laid out in a substrate performed best. We also found that the critical threshold is in fact five orders of magnitude lower than previously found by theoretical models alone. In addition, we determined that much of the difference between the theoretical, and practical breaking point can be attributed to problems that computer architects are particularly well suited to solve.

Acknowledgments: This work is supported in part by the DARPA QuIST Program (AFRL-F30602-01-2-0521), NSF Nanoscale Program (CCF-0210373) and NSF CAREER grants (CCF-0133188). Additional support is provided by the A. P. Sloan foundation. We would like to thank our anonymous reviewers for their constructive feedback.

References

[1] I. L. Chuang, N. Gershenfeld, M. G. Kubinec, and D. W. Leung, “Bulk quantum computation with nuclear-magnetic-resonance: theory and experiment,” *Proc. R. Soc. London A*, vol. 454, no. 1969, pp. 447–467, 1998.

[2] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland, “Demonstration of a fundamental quantum logic gate,” *Phys. Rev. Lett.*, vol. 75, p. 4714, 1995.

[3] M. A. Rowe, A. Ben-Kish, B. DeMarco, D. Leibfried, V. Meyer, J. Beall, J. Britton, J. Hughes, W. M. Itano, B. Jelenkovic, C. Langer, T. Rosenband, and D. J. Wineland, “Transport of quantum states and separation of ions in a dual rf ion trap,” *Quantum Information and Computation*, vol. 2, pp. 251–271, 2002.

[4] D. J. Wineland, M. Barrett, J. Britton, J. Chiaverini, B. L. DeMarco, W. M. Itano, B. M. Jelenkovic, C. Langer, D. Leibfried, V. Meyer, T. Rosenband, and T. Schaetz, “Quantum information processing with trapped ions,” *Phil. Trans. Royal Soc. London A*, vol. 361, pp. 1349–1361, 2003.

[5] G. T. Foster, L. A. Orozco1, H. M. Castro-Beltran, and H. J. Carmichael, “Quantum state reduction and conditional time evolution of wave-particle correlations in cavity qed,” *Phys. Rev. Lett.*, vol. 85, pp. 3149–3152, Oct 2000.

[6] P. Domokos, J. M. Raimond, M. Brune, and S. Haroche, “Simple cavity-qed two-bit universal quantum logic gate: The principle and expected performances,” *Phys. Rev. A*, vol. 52, no. 5, pp. 3554–3559, 1995.

[7] D. Kielpinsky, C. Monroe, and D. Wineland, “Architecture for a large-scale ion trap quantum computer,” *Nature*, vol. 417, p. 709, 2002.

[8] T. Metodiev, A. Cross, D. Thaker, K. Brown, D. Copsey, F. T. Chong, and I.L.Chuang, “Preliminary results on simulating a scalable fault tolerant ion-trap system for quantum computation,” in *3rd Workshop on Non-Silicon Computing*, June 2004.

[9] D. Aharonov, *Noisy Quantum Computation*. PhD thesis, The Hebrew Univesity, Jerusalem, 1999.

[10] B. Ömer, “Quantum programming in qcl,” Master’s thesis, Technical University of Vienna, 2000.

[11] M. U. Mock, C. Chambers, and S. J. Eggers, “Calpa: A tool for automating selective dynamic compilation.,” in *In Proceedings of the 33rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-33)*, pp. 291–302, Dec 2000.

[12] E. Feigin, *A Case for Automatic Run-Time Code Optimization*. PhD thesis, Harvard College, Div. of Eng. and Applied Sciences, 1999.

[13] J. Auslander, M. Philipose, C. Chambers, S. J. Eggers, and B. N. Bershad, “Fast, effective dynamic compilation,” in *SIGPLAN Conference on Programming Language Design and Implementation*, pp. 149–159, 1996.

[14] A. Steane, “Error correcting codes in quantum theory,” *Phys. Rev. Lett.*, vol. 77, 1996.

[15] P. Shor, “Scheme for reducing decoherence in a quantum computer memory,” *Phys. Rev. A*, vol. 52, no. 2493, 1995.

[16] C. H. Bennett, D. P. Vincenzo, J. A. Smolin, and W. K. Wootters, “Mixed state entanglement and quantum error correction,” *Phys. Rev. A*, vol. 54, no. 5, pp. 3824–3851, 1996.

[17] R. Laflamme, C. Miquel, J.-P. Paz, and W. H. Zurek, “Perfect quantum error correction code,” *Phys. Rev. Lett.*, vol. 77, p. 198, 1996. arXive e-print quant-ph/9602019.

[18] V. Adve, C. Lattner, M. Brukman, A. Shukla, and B. Gaeke, “LLVA: A Low-level Virtual Instruction Set Architecture,” in *Proceedings of the 36th annual ACM/IEEE international symposium on Microarchitecture (MICRO-36)*, (San Diego, California), Dec 2003.

[19] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, UK: Cambridge University Press, 2000.

[20] G. M. Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities,” in *Proc. AFIPS Conf.*, (Reston,Virginia), pp. 483–485, 1967.

[21] G. S. Taylor, J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, and W. S. Scott, “Magic: A vlsi layout system.,” in *Proceedings of the 21th Design Automation Conference*, pp. 152–159, 1984.

[22] L. McMurchie and C. Ebeling, “Pathfinder: A negotiation-based performance-driven router for FPGAs,” in *Proceedings of ACM Symp. on Field-Programmable Gate Arrays*, pp. 111–117, 1995.

[23] T. Yang and A. Gerasoulis, “List scheduling with and without communication delays,” *Parallel Comput.*, vol. 19, no. 12, pp. 1321–1344, 1993.

[24] H. Rosé, T. Asselmeyer-Maluga, M. Kolbe, F. Nieh´orster, and A. Schramm, “The fraunhofer quantum computing portal - www.qc.fraunhofer.de - a web-based simulator of quantum computing processes,” tech. rep., Fraunhofer Institute for Computer Architecture and Software Technology, Berlin, 2003.

[25] A. Fog, “Chaotic random number generators with random cycle lengths.” www.agner.org/random/theory, Nov 2001.

[26] A. Steane, “Active stabilisation, quantum computation and quantum state synthesis,” *quant-ph/9611027*, 1996.

[27] M. Oskin, F. Chong, and I. Chuang, “A practical architecture for reliable quantum computers,” in *Proc. International Symposium on Computer Architecture (ISCA 2001)*, (New York), ACM Press, 2001.

A.5 ISCA'06 Paper

To appear in the International Symposium on Computer Architecture (ISCA-33)

Quantum Memory Hierarchies: Efficient Designs to Match Available Parallelism in Quantum Computing

Darshan D. Thaker[†] Tzvetan S. Metodi[†] Andrew W. Cross[‡] Isaac L. Chuang[‡] Frederic T. Chong[⋆]

[†] *University of California at Davis, One Shields Avenue, Davis, CA 95616, USA*
[‡] *Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139*
[⋆] *University of California at Santa Barbara, Santa Barbara, CA 93106, USA*

Abstract

The assumption of maximum parallelism support for the successful realization of scalable quantum computers has led to homogeneous, "sea-of-qubits" architectures. The resulting architectures overcome the primary challenges of reliability and scalability at the cost of physically unacceptable system area. We find that by exploiting the natural serialization at both the application and the physical microarchitecture level of a quantum computer, we can reduce the area requirement while improving performance. In particular we present a scalable quantum architecture design that employs specialization of the system into memory and computational regions, each individually optimized to match hardware support to the available parallelism. Through careful application and system analysis, we find that our new architecture can yield up to a factor of thirteen savings in area due to specialization. In addition, by providing a memory hierarchy design for quantum computers, we can increase time performance by a factor of eight. This result brings us closer to the realization of a quantum processor that can solve meaningful problems.

1 Introduction

Conventional architectural design adheres to the concept of **balance**. For example, the register file depth is matched to the number of functional units, the memory bandwidth to the cache miss rate, or the interconnect bandwidth matched to the compute power of each element of a multiprocessor. We apply this concept to the design of a quantum computer and introduce the *Compressed Quantum Logic Array* (CQLA), an architecture that balances components and resources in terms of exploitable parallelism. The primary goal of our design is to address the problem of large area, approximately 1 m^2 on a side, of our previous design [1].

Specifically, we discover that the prevailing approach to designing a quantum computer, that of supporting maximal parallelism, is area inefficient. We also find that exploitable parallelism is inherently limited by both resource constraints and application structure. This lack of parallelism gives us the freedom to increase density by specializing components as blocks of memory and blocks of computation.

We introduce the idea of periodically reducing our investment in reliability and thereby increasing speed. By encoding the compute regions differently than memory we provide very fast compute regions, while allowing the memory to be slower and more reliable. To ensure that the faster compute region does not suffer from too many stalls, we employ a quantum memory hierarchy wherein the cache utilizes the same encoding mechanism as the compute region. When making this effort to improve speed, it is critical that overall system fidelity is maintained. We show how this can be accomplished.

Due to the quantum no-cloning theorem [2], it is necessary for all quantum data to physically move from source to destination. We cannot create a copy of the data and send the copy. Our architecture focuses on implementation with an array of trapped atomic ions, one of the most mature and scalable technologies that provides a wealth of experimental data. In ion-traps, the physical representation of data are ions that are in constant motion, on a two dimensional grid, throughout the computation. Since this physical movement is slow, yet unavoidable, it limits available parallelism at the microarchitecture level.

At the application level, we find that only a limited amount of parallelism can be extracted from key quantum algorithms. This means that we may only need a few compute blocks for all the qubits in memory. This is in contrast to the popular "sea of qubits" model which allows computation at every qubit. Our results show up to a 13X increase in density, particulary important in addressing our primary goal, and a speedup of about 8. The large area improvement brings the engineering of a quantum architecture closer to the capabilities of current implementation technologies.

The choice of quantum error correction codes (ECC) influences our results and the architecture. In our specialized architecture analysis, we use the previously considered Steane $[[7,1,3]]$ code [3] and utilize a newly optimized Bacon-Shor $[[9,1,3]]$ code [4, 5]. The $[\![9,1,3]\!]$ code, though larger than the $[\![7,1,3]\!]$ code since it uses more physical qubits to encode a single logical qubit, requires far fewer resources for error-correction [6], thus reducing the overall area and increasing the speed.

Furthermore, we find that communication is generally dominated by computation for error correction. This computation allows us to absorb the cost of moving data between different regions of the architecture. Error correction is so substantial, in fact, that quantum computers do not suffer from the *memory wall* faced by conventional computers. Thus our dense structure with a communication infrastructure based on our prior work [1] can accommodate applications with highly-demanding communication patterns.

In summary, the **contributions** of this work are: 1) Our specialized architecture, the CQLA, successfully tackles the issue of size, which has been the biggest drawback facing large-scale realizable quantum computers. 2) We show that current parallelism in quantum algorithms is inherently limited and consideration of physical resources and data movement restrict it even further. 3) We present and analyze the abstractions of memory, cache and computation units for a quantum computer; based on the insight that we can reduce reliability for the compute units and cache without sacrificing overall computation fidelity. This approach helps us significantly increase the performance of the system.

The paper is organized as follows. Section 2 provides a background of the homogeneous QLA architecture and the low-level microarchitecture assumptions of our system. Section 3 motivates the specialized CQLA architecture and introduces the architectural abstractions. Thereafter we discuss how the Steane and Bacon-Shor error correction codes affect the design of the CQLA. Results and analysis of our abstractions are the focus of section 5 following which we provide details of computation versus communication requirements of the most widely accepted quantum applications. We end with future directions in Section 7 and our conclusions in Section 8.

2 Background

Our architectural model is built upon our previous work on the Quantum Logic Array (QLA) architecture [1]. The QLA architecture is a hierarchical array-based design that overcomes the primary challenges of scalability for large-scale quantum architectures. It is a homogeneous, tiled architecture with three main components: logical qubits implemented as self-contained computational tiles structured for quantum error error correction; trapped atomic ions as the underlying technology; finally, teleportation-based communication channels utilizing the concept of quantum repeaters to overcome the long-distance communication constraints.

2.1 The Logical Qubit

The basic structure of the QLA, our prior work, implements a fault-tolerant quantum bit, or a *logical qubit* as a self-contained tile whose underlying construction is intended for quantum error correction, by far the most dominant and basic operation in a quantum machine [7]. Quantum error correction is expensive because arbitrary reliability is achieved by recursively encoding physical qubits at cost of exponential overhead. Recursive error correction works by encoding N physical ion-qubits into a known highly-correlated state that can be used to represent a single logical data qubit. This data qubit is now at level 1 recursion and may have the property of being in a superposition of "0" and "1" much like a single physical qubit. Encoding once more we can create a logical qubit at level 2 recursion with N^2 physical ion-qubits. With each level, L, of encoding the probability of failure of the system scales as $p_0^{2^L}$, where p_0 is the failure rate of the individual physical components given a fault-tolerant arrangement and sequence of operations for the lower level components. The ability to apply logical operations on a logical qubit without the need to decode and subsequently re-encode the data is key to the existence of fault-tolerant quantum microarchitecture design, where arbitrary reliability can be efficiently reached through recursive encoding.

The logical qubits in the QLA are arranged in a regular array fashion, connected with a tightly integrated repeater-based [8] interconnect. This makes the high-level design of the QLA very similar to classical tile based architectures. The key difference is that the communication paths must account for data errors in addition to latency. Integrated repeaters known as *teleportation islands* redirect qubit traffic in the 4 cardinal directions by teleporting data from one island to the next. This interconnect design is one of the key innovative features of QLA architecture, as it allows us to completely overlap communication and computation, thus eliminating communication latency at the application level of the program.

Anticipating technology improvements in the near future we found that for performing large, relevant instances of Shor's factoring algorithm, sufficient reliability is achieved

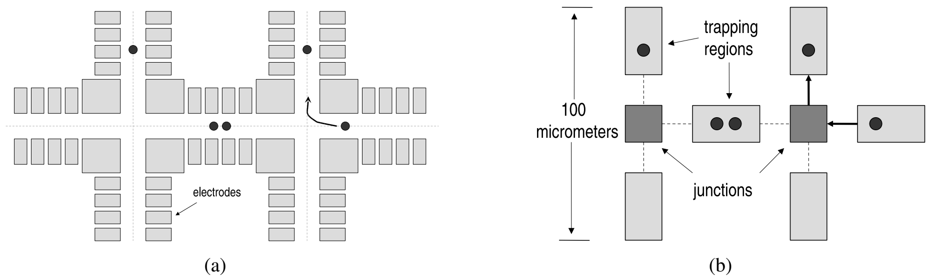


Figure 1. (a) A simple schematic of the basic elements of a planar ion-trap for quantum computing. Ions are trapped in any of the trapping regions shown and ballistically shuttled from one trapping region to another. When two ions are together a two-qubit gate can be performed. (b) Our abstraction of the ion-trap layout. Each trapping region can hold up to two ions for two-qubit gates. The trapping regions are interconnected with the crossing junctions which are treated as a shared resource.

at level 2 encoding per logical qubit using the Steane $[\![7,1,3]\!]$ error correction code [9]. In the QLA, computation could occur at any logical qubit and each logical gate is followed by an error correction procedure. To preserve homogeneity and maximum flexibility for large-scale applications each logical qubit was accompanied by all necessary error correction auxiliary qubit resources such that computational speed was maximized. This amounted to a $(1:2)$ ratio between logical data qubits and ancillary qubits.

2.2 Low-Level Physical Architecture Model

At the lowest level our architecture design is based on the ion-trap technology for quantum computation. Initially proposed by Cirac and Zoller in 1995 [10], the technology uses a number of atomic ions that interact with lasers to quantum compute. Quantum data is stored in the internal electronic and nuclear states of the ions, while the traps themselves are segmented metal traps (or electrodes) that allow individual ion addressing. Two ions in neighboring traps can couple to each other forming a linear chain of ions whose vibrational modes provide qubit-qubit interaction used for multi-qubit quantum gates [11, 12]. Together with single bit rotations this yields a universal set of quantum logic gates. All quantum logic is implemented by applying lasers on the target ions, including measurement of the quantum state [13, 14, 15, 16]. Sympathetic cooling ions absorb vibrations from data ions, which are then dampened through laser manipulation [17, 18]. Recent experiments [19, 20, 21] have demonstrated all the necessary components needed to build a large-scale ion-trap quantum information processor. Finally, multiple ions in different traps can be controlled by focusing lasers through MEMS mirror arrays [22].

Figure 1 shows a schematic of the physical structure of an ion trap computer element. In Figure 1(a) we see a single ion trapped in the middle trapping region. Trapping regions are the locations where ions can be prepared for the execution of a logical gate, which is implemented by an external laser source pulsed on the ions in the trap. In the figure we see an ion moving from the far right trapping region to the top-right for the execution of a two-bit logical operation.

Operation	Time μs now(future)	Failure Rate now(future)
Single Gate	1 (1)	10^{-4} (10^{-8})
Double Gate	10 (10)	0.03 (10^{-7})
Measure	200 (10)	0.01 (10^{-8})
Movement	20 (10)	0.005 $(5\times10^{-8})/\mu$m
Split	200 (0.1)	
Cooling	200 (0.1)	
Memory time	10 to 100 sec	
Trap Size	~ 200 $(1-5)$ μm	

Table 1. Column 1 gives estimates for execution times for basic physical operations used in the QLA model. Currently achieved component failure rates are based on experimental measurements at NIST with $^{9}Be^{+}$ ions, and using $^{24}Mg^{+}$ ions for sympathetic cooling [14, 12]. All parameters are followed by their projected parameters in parenthesis, extrapolated following recent literature [23, 24, 25], and discussions with the NIST researchers; these estimates are used in modeling the performance of our architecture.

Figure 1(b) demonstrates our abstraction of the physical ion-trap layout. The layout can be represented as a collection of trapping regions connected together through shared junctions. A fundamental time-step, or a clock cycle, in an ion-trap computer will be defined as any physical, unencoded logic operation (one-bit or two-bit), a basic move operation from one trapping region to another, and measurement. Table 1 summarizes current experimental parameters and corresponding optimistic parameters for ion-traps.

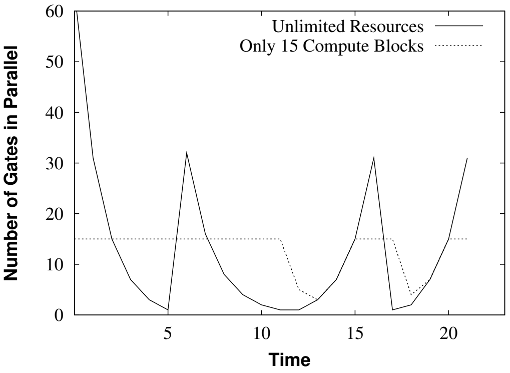


Figure 2. For a 64-qubit adder, the amount of parallelism that can be extracted when resources are unlimited, and when the number of gates per cycle are limited. This figure shows that if 15 gates, or an unlimited number of gates could be performed in each cycle, the total runtime would remain the same. compute blocks increases.

In our subsequent analysis we will assume that each *clock cycle* for a fundamental time-step has a duration of 10 μs, failure rates are 10^{-8} for single-qubit operations and measurement, 10^{-7} for CNOT gates [25], and order of 10^{-6} per fundamental move operation. The movement failure rate is expected to improve from what it is now as trap sizes shrink and electrode surface integrity continues to improve. We will assume trap sizes of 5μm each [26], and on the order of 10 electrodes per trapping region [27], which gives us a trapping region dimension (including the junction) of 50μm. The parameters chosen for our study are optimistic compared to [28] and [29]. Both of those papers, assume more pessimistic near term parameters which are useful for building a 100 bit prototype, but probably not a scalable quantum computer that can factor 1024-bit numbers using Shor's algorithm. Based on the quantum computing ARDA roadmap [23], we feel justified in using aggressive parameters when looking 10-15 years into the future.

3 Architectural Abstractions

This section motivates the need for a compact architecture for quantum processors and describes our design the CQLA (Compressed Quantum Logic Array). We discuss how separation into memory and compute regions benefits the CQLA and then present our quantum memory hierarchy.

3.1 Motivation

Conventional quantum processor designs are based on the *sea-of-qubits* design and allow computation to take place anywhere in the processor. This design philosophy follows the idea of maximum parallelism and is employed in our previous work [1]. The area consumption of such a design however, is untenably large, about 1 m^2 to factor a 1024-bit number.

When we consider the amount of available parallelism in quantum applications, we discover that much is to be gained by limiting computation to a specifically designated location. The remaining area can be optimized for storage of quantum data. A good example for the benefit of specialization in quantum applications is the Draper carry-lookahead quantum adder [30], which forms a basic basic component of Shor's quantum factoring algorithm [31]. Figure 2 shows that providing unlimited computational resources for a 64-bit adder does not offer a performance benefit over limiting the computation to 15 locations. As illustrated in Section 2, the number of ancillary resources for each data location where computation is allowed is twice as large. In this example, by providing only 15 compute locations instead of 64, we can reduce the area consumed by the adder by approximately *half* and yet have no change in performance.

3.2 Specialized Components

The facts that qubits in an ion-trap quantum processor have *large lifetimes* when idle, allows us to improve logical qubit density in the memory. Qubits in memory can wait for a longer time period between two consecutive error corrections. We use this to significantly reduce the error correction ancillary resources in memory, thereby reducing its density. The majority of computation, on the other hand, is an interaction between two distinct logical qubits. To maintain adequate system fidelity, every gate must be followed by an error correction procedure. Consequently, a quantum processor spends most of its time performing error correction and the compute regions are designed to allow fast error correction by providing a greater number of ancilla in the logical qubits. Figure 3(a) shows a specialization into compute and memory regions. The ratio of (data:ancilla) can be seen to be $(8 : 1)$ for memory and $(1 : 2)$ for the compute region.

While specialization helps address our primary goal of reducing size, it can possibly also reduce performance. In Section 5 we show how judiciously choosing the size of the compute region helps maintain adequate performance while simultaneously reducing size .

3.3 Quantum Memory Hierarchy

Another important architectural design choice is the effect of the error correction code chosen in both the memory and the compute regions. Error correction is the most dominant procedure and the resources used increase exponentially with each level of concatenation. In addition to

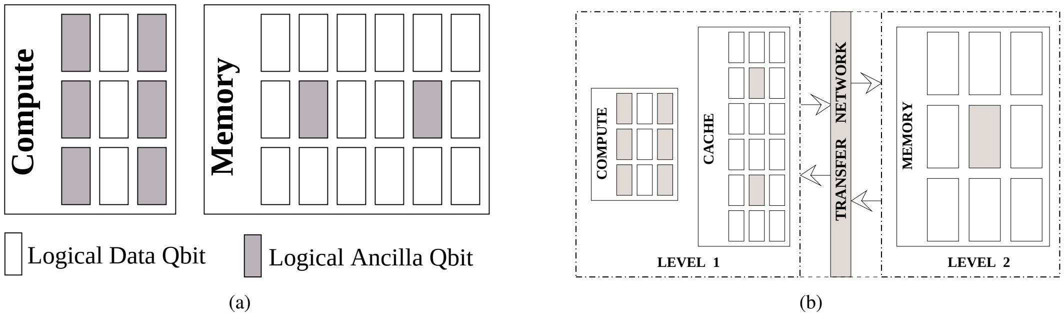


Figure 3. **(a)** Memory is denser since it has fewer ancilla qubits. The figure shows 3 data qubits in the compute block which take the same area as 8 data qubits in memory. In the CQLA each compute block holds nine 9 data qubits and 18 ancilla. Both compute and memory are at level 2 encoding. **(b)** Memory is at level 2 encoding, while the compute and cache are at level 1 encoding. The complete CQLA consists of memory at level 2, compute regions at level 2 and also a cache and compute region at level 1.

resources, the time to error correct increases exponentially with each level of concatenation. The benefits of concatenated error correction are that the reliability of each operation increases double exponentially, thus allowing far greater number of total operations to be performed. For any application, all logical qubits are not being acted upon by gates for the entire duration of the algorithm. In fact, just like classical computers, data locality is a common phenomenon. This implies that a logical qubit could start at level 2 encoding, be encoded at level 1 during the peak in its activity and return to level 2 when idle.

We now introduce a quantum memory hierarchy, in addition to the specialized design. Memory at level 2, which is optimized for area and reliability will be inherently slower than a computational structure, at level 1, optimized for gate execution. This necessitates the need for a cache that can alleviate the need for constant communication.

Figure 3(b) outlines this approach. the separation between memory and compute regions. The cache and the compute regions here are similar to Figure 3(a) in every way save that they are at a lower level of encoding. In the memory hierarchy, memory and cache have a similar design, only memory is at a higher level of encoding, and hence is slower and much more reliable. The critical feature here is the **transfer network** which is more complicated and hence slower than the teleporation channels described above. The transfer network comes into play only when we change the encoding of a logical qubit. For all other communication (within compute blocks, between cache and compute blocks and within memory) teleportation is still the chosen mechanism. Section 4 describes how the transfer process is performed in a fault-tolerant manner.

4 Error Correction and Code Transfer

In this section we describe the cost of the error correction circuits and code-transfer networks we use when a specific physical layout is considered. Section 2.2 describes in detail our technology parameters, which we find to be necessary for such a large-scale architecture. These parameters allow the large scalability to be achieved because the physical component failure rates are below the threshold value needed for efficient error correction [32].

4.1 Error Correction Codes

Some of the best error correction codes (ECC) are ones that use very few physical qubits, and allow "easy" fault-tolerant gate implementations. A requirement of a fault-tolerant system is that computation proceeds without decoding the encoded data. Thus logical gates are implemented directly on encoded qubits, ensuring that errors introduced during the gate can be corrected. Many code choices for EC allow *transversal* logical gate implementation, which means that the same physical gate acts on each lower-level qubit.

Each logical quantum gate is preceded and followed by an error correction procedure. The EC procedure works by encoding ancillary qubits in the logical "0" state of the data and interacting the data and the ancilla. The interaction causes errors in the data to propagate to the ancilla and to be detected when the ancilla is measured. There are several very important logical gates that we must consider during error correction. The bit-flip gate, X flips the value of the qubit by reversing the probabilities between its "0" component and its "1" component. The phase-flip gate, Z, acts only on the qubit's "1" component by changing its sign. The most important gate is the controlled-X gate (denoted as the

CNOT gate) which flips the state of the target qubit whenever the state of the control qubit is set. Errors on the data can be understood as the product of phase-flips and a bit-flips. A syndrome is extracted for each types of error. We only present the cost of error correction networks and details relevant to building a large-scale architecture. The interested reader can refer to the literature for additional theoretical information [33].

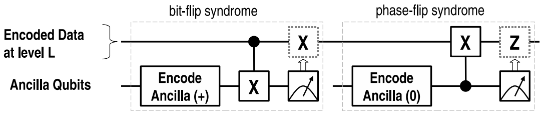


Figure 4. A high-level view of an error correction sequence. Two syndromes for bit-flip and phase-flip errors are extracted.

Figure 4 is a simple schematic of the general error correction procedure, where time flows from left to right and each line represents the evolution of an encoded logical qubit. An error correction code is labeled by $[[n,k,d]]$, encoding k logical qubits into n qubits and correcting $(d-1)/2$ errors. If our target reliability is such that we require L levels of recursion, each line in Figure 4 represents n^L level zero qubits. For the bit-flip error syndrome the ancilla are encoded into the logical $(0+1)$, and the transversal CNOT gate, which is essentially n level $(L-1)$ transversal CNOT gates of which the ancillary qubits are targets. Each of the lower level CNOT gates is followed by a lower level error correction unless the lower level is zero. In our architecture analysis we provide information about two error correcting codes: the Steane $[[7,1,3]]$ code [9], and an improved version of the Shor $[[9,1,3]]$ code [34] denoted as the Bacon-Shor code [4, 5, 6].

The Steane $[[7,1,3]]$ **Code** encodes 1 qubit into 7 qubits, and is the smallest error correction code allowing transversal gate implementation for all gates involved in concatenated error correction algorithms. The addition of the T phase gate, which is harder to implement, provides universal quantum logic using the $[[7,1,3]]$ error correcting code. For this reason it was used as the underlying error correcting code in the analysis of the QLA architecture [1]. It consists of 7 data ions which encode our logical level 1 qubit with 14 ancillary ions used for error correction, seven of which are used in the error correction and the other verify the ancilla.

Considering communication, the level 1 error correction circuit in will take 154 cycles, where each cycle is in the order of 10 microseconds, and can be as large as 0.003 per error correction procedure at level 1. A level 2 $[[7,1,3]]$ qubit will be composed of 7 level 1 data qubits and 7 level 1 ancilla qubits - there is no need for verification ancilla at $L=2$.

Error Correction Metric Summary		
Architecture Metric	Error Code - Level	Value
EC Time (seconds)	$[[7,1,3]]$ - L1	3.1×10^{-3}
	$[[7,1,3]]$ - L2	0.3
	$[[9,1,3]]$ - L1	1.2×10^{-3}
	$[[9,1,3]]$ - L2	0.1
Qubit Size (mm^2)	$[[7,1,3]]$ - L1	0.2
	$[[7,1,3]]$ - L2	3.4
	$[[9,1,3]]$ - L1	0.1
	$[[9,1,3]]$ - L2	2.4
Transversal Gate Time (seconds)	$[[7,1,3]]$ - L1	6.2×10^{-3}
	$[[7,1,3]]$ - L2	0.5
	$[[9,1,3]]$ - L1	2.4×10^{-3}
	$[[9,1,3]]$ - L2	0.2
Size, number of logical qubits	$[[7,1,3]]$ - L1	7
	$[[7,1,3]]$ - L1(ancilla)	21
	$[[7,1,3]]$ - L2	49
	$[[7,1,3]]$ - L2(ancilla)	441
	$[[9,1,3]]$ - L1	9
	$[[9,1,3]]$ - L1(ancilla)	12
	$[[9,1,3]]$ - L2	81
	$[[9,1,3]]$ - L2(ancilla)	298

Table 2. Error Correction Metric Summary. Given the fact that we use optimistic ion-trap parameters all numbers are estimates and are thus rounded to only one significant digit.

The size of a level 2 qubit will be 3.4 mm^2, and a fully serialized error correction will last approximately 0.3 seconds (this is two orders of magnitude more than the time to error correct at level 1).

Bacon-Shor $[[9,1,3]]$ **Code:** The $[[9,1,3]]$ code was the first error correcting code to be discovered for arbitrary errors [34]. Recent observations make this code faster and spatially smaller than the $[[7,1,3]]$ code [4, 5, 6]. The compact structure of the physical layout for the $[[9,1,3]]$ code significantly improves communication requirements. At level 1 the error correction time is only 0.001 seconds and 0.1 seconds at level 2. The level 2 qubit size is approximately 2.4 mm^2. Table 2 summarizes the error correction we have used and their parameters for some useful architecture metrics.

4.2 Code Transfer Networks: Overview

One of the most interesting components of the memory hierarchy are the code transfer regions. This region transfers data encoded in code C1 to a second code C2 without the need to decode. Figure 4.2 illustrates this concept. The transfer network *teleports* the data in C1 to C2, where C1 and C2 may be any two error correcting codes. The code teleportation procedure works much the same way as standard data teleportation that is used for communication. A correlated ancillary pair is prepared first between

(seconds)	7-L1	7-L2	9-L1	9-L2
7-L1	0	0.6	0.02	0.2
7-L2	1.3	0	1.3	1.5
9-L1	0.01	0.5	0	0.1
9-L2	0.4	0.9	0.4	0

Table 3. Transfer network latency for a combination of the $[\![7,1,3]\!]$ and $[\![9,1,3]\!]$ codes.

C1 and C2 through the use of a multi-qubit cat-state (i.e. "$(00...0+11...1)$"). The data qubit interacts with the equivalently encoded ancillary qubit through a CNOT gate, and the two are measured. Following the measurement the state of the data is recreated at the C2 encoded ancillary qubit. This process is required every time we transfer a qubit from memory to the cache or vise-versa. Table 3 summarizes the times for different code transfer combinations between levels 1 and 2 for the $[\![7,1,3]\!]$ and the $[\![9,1,3]\!]$ codes.

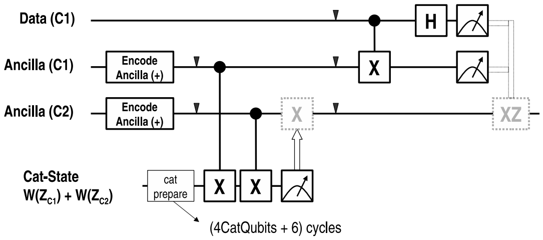


Figure 5. Code Teleportation Network from Code 1 (C1) to Code 2 (C2) C1 and C2 can even be the same error correcting code, but different levels of encoding. The solid triangles denote an error correction step.

5 CQLA Analysis and Results

This section provides analysis of the abstractions presented in

to perform quantum modular exponentiation.

5.1 Specialization into Memory

We now analyze our design, the CQLA, when it separates the quantum processor into memory and compute regions. High density in memory is achieved by greatly reducing the ratio of logical data qubits to logical ancilla qubits, which is $(8:1)$ in memory and is $(1:2)$ in the compute regions. This greatly reduces overall area since prior work had a ratio of $(1:2)$ throughout the architecture. Thus the memory is denser, but slower, which is permissible due to the large memory wait times 1.

Quantum modular exponentiation is the most time consuming part of Shor's algorithm, and the Draper carry-lookahead adder is its most efficient implementation. This adder comprises single qubit gates, two qubit cnot gates and three qubit toffoli gates and is dominated by toffoli gates. The time to perform a single fault-tolerant toffoli is equal to the time for fifteen two qubit gates, each of which is followed by an error-correction step. Table 5.1 shows the savings that can be achieved when using denser memory. Note that performance is minimally impacted for the Steane Code as we exploit the limited parallelism in the adder. We address the parallelism available within the application itself and determine the number of compute blocks to maximally exploit this parallelism with change with problem size. Figure 6(a) shows how for a fixed problem size, utilization of each compute block decreases with an increase the number of compute blocks. Clearly, the decrease in utilization is offset by the increase in overall performance. Thus the challenge here is to find the *balance* between utilization and performance.

We compare all our results to [1], which used only the Steane ECC. Since the Bacon-Shor ECC uses fewer overall resources 2 and allows faster error-correction, a design based on these codes not only is much smaller, but is also faster. The CQLA, thus reduces area required by a **factor of 9** with minimal performance reduction for the Steane ECC and by a **factor of 13** with a **speedup of 2** when using the Bacon-Shor ECC. To compare the relative merit our design choices, we use the *gain product* which can be defined by $GP = (Area_{old} * AdderTime_{old}) / (Area_{CQLA} * AdderTime_{CQLA})$ where *AdderTime* is the average time per adder for modular exponentiation. The gain product indicates the improvement in system parameters relative to our prior work, the QLA. The higher the gain product, the better the collective improvement in area and time of our system.

Communication Issues: Toffoli gates cannot be directly implemented on encoded data and have to be broken down into multiple two qubit gates. Performing a fault-tolerant Toffoli between three logical qubits requires extra logical ancilla and logical cat-state qubits. The flow of data between these nine qubits to complete a single toffoli forms the most intense communication pattern during the entire addition operation. To study the bandwidth requirements during the toffoli gates, we developed a scheduler that would try to have all the requirements for communication (creating EPR pairs, transporting and purifying them) in place while the logical qubit to be transported was undergoing error-correction after completion of the previous gate. With bandwidth of one channel, it was possible to overlap communication with computation for the Steane $[[7,1,3]]$ code. To enable this overlap when using the Bacon-Shor

Input	Compute	Area Reduced (Factor of)		SpeedUp		Gain Product	
Size	Blocks	St-Code	BSr-Code	St-Code	BSr-Code	St-Code	BSr-Code
32-bit	4	6.69	9.80	0.54	1.47	3.61	14.41
	9	3.22	4.74	0.97	2.9	3.14	13.74
64-bit	9	6.36	9.32	0.70	1.92	4.45	17.70
	16	3.79	5.56	0.98	3.0	3.71	16.68
128-bit	16	7.24	10.6	0.72	1.97	5.24	20.88
	25	4.90	7.17	0.96	2.84	4.70	20.36
256-bit	36	6.65	9.47	0.92	2.51	6.12	23.68
	49	5.07	7.43	0.98	2.98	4.96	22.14
512-bit	64	7.42	10.87	0.92	2.50	6.80	27.18
	81	6.06	8.87	0.98	2.91	5.94	25.81
1024-bit	**100**	**9.14**	**13.4**	**0.80**	**2.19**	**7.35**	**29.35**
	121	7.81	11.45	0.97	2.65	7.60	30.34

Table 4. For various size inputs, this table shows how the CQLA performs for Modular Exponentiation. The space saved due to compressing the memory blocks and separating memory and compute regions is shown as compared to prior work [1]. St-Code is the Steane ECC and BSr-Code is the Bacon-Shor code. The Gain Product is compared with our prior work, the QLA, which has a Gain Product of 1.0.

code, the required bandwidth was three channels. Table 2 shows that while a logical qubit encoded in the Bacon-Shor code is smaller when ancilla are considered; it has more data qubits than the Steane code. Since only data qubits are involved during teleportation, the time for teleporting a logical qubit in the Bacon-Shor code is greater. In addition, the Bacon-Shor codes take far fewer error-correction cycles. These two factors push its bandwidth requirement higher. Note that the higher bandwidth is accounted for in results of Table 5.1.

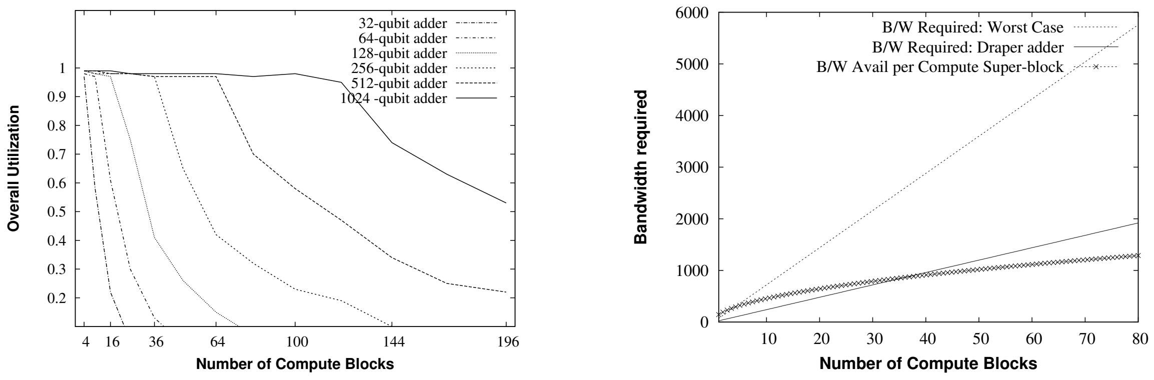
Superblocks: In the CQLA, several compute blocks together form compute superblocks. This is done to exploit the locality inherent to an application. Having larger superblocks also increases the perimeter bandwidth between the compute and memory regions of the CQLA. This increase in bandwidth of a larger superblock is offset by the much greater increase in communication required. Our intuition tells us that at a certain point, it may be more efficient to have multiple small superblocks instead of one large superblock. To determine this number concretely, we plot the change in bandwidth required against change in bandwidth available. Figure 6(b) shows the cross-over point is 36 compute blocks per superblock, immaterial of what error correction code is used. Thereafter it is no longer beneficial to increase the size of an individual compute superblock.

5.2 Memory Hierarchy

Reducing the encoding level of the compute region will dramatically increase its speed. Recall that resources, time and reliability all increase exponentially as we increase the level of encoding. With the compute region at level 1 and memory at level 2, the challenge is the very familiar one of the CPU being an order of magnitude faster than the memory. To maximize the benefit of a much faster compute region, we introduce the quatum memory hierarchy. In our hierarchy, the memory is at level 2 encoding (slow and reliable), cache is at level 1 (faster, less reliable) and the compute region is also at level 1 (fastest and same reliability as cache). The difference in speed between the compute region and the cache is the due to a greater number of ancilla in the compute region.

To study the behavior of the CQLA with a cache and multiple encoding levels, we developed a simulator that models a cache. The simulator takes into account the computation cost in both encoding levels and also the cost of transferring logical qubits between encoding levels. The application under consideration is still the Draper carry-lookahead adder. Input to the simulator is a sequence of instructions; each instruction is similar to assembly language and describes a logical gate between qubits. We have written generators that output this code in a form that can take advantage of an architecture with maximal parallelism.

When the simulator runs this code in the sequence intended by the Draper carry-lookahead adder, the cache hit-rate is limited to 20%. To improve the hit-rate, we utilize the following optimized approach. Since we are scheduling statically, the instruction fetch window for the simulator can be the whole program. The simulator takes advantage of this by first creating a dependency list of all input instructions. Then it carefully selects the next instruction such that probability of finding all required operands in the cache is maximized. This optimized fetch yields a cache hit-rate of



(a) (b)

Figure 6. **(a)** Change in utilization as the number of compute blocks increases. **(b)** The point of intersection of the two bottom curves is the optimal size of a compute superblock. These two curves are bandwidth required (at the perimeter of the compute superblock) in modular exponentiation and bandwidth available. The third steep curve, is the worst case bandwidth required.

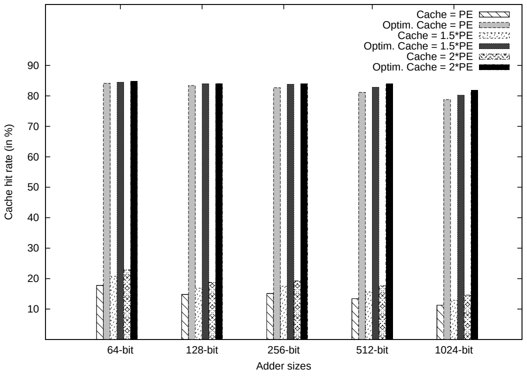


Figure 7. Shows the cache hitrate for different adders when both cache and compute region are at Level 1 recursion. Largest cache considered holds twice the number of logical qubits as the compute block. Results for both the non-optimized version and the optimized version are shown.

almost 85% immaterial of adder size and cache size. The replacement policy in the cache is least recently used. Figure 7 shows the cache hit-rates for different sized adders for the non-optimized and optimized instruction fetch approaches. If n is the number of logical qubits in the compute region, the cache sizes we studied were $n, 1.5n$ and $2n$. As the graph shows, the increase in hit-rate is more pronounced due to the optimized fetch than increasing cache size. For the CQLA, we thus employ a cache size of twice the number of qubits in the compute region. The high hit-rate means the transfer networks will not be overwhelmed.

Fault-tolerance with multiple encoding levels: A quantum computer running an application of size $S = KQ$, where K is the number of time-steps and Q is the number of logical qubits, will need to have a component failure rate of at most $P_f = 1/KQ$. To evaluate the expected component failure rate at some level or recursion we use Gottesman's estimate for local architectures [35] shown in Equation 1 below.

$$P_f = \frac{1}{cr^2 r^L}(cr^2 p_0)^{2^L} = \frac{p_{th}}{r^L}(p_{th}^{-1} p_0)^{2^L} \qquad (1)$$

The value for r is the communication distance between level 1 blocks which are aligned in QLA to allow $r = 12$ cells on average and L denotes the level of recursion. The threshold failure rate, p_{th}, for the Steane $[[7,1,3]]$ circuit accounting for movement and gates was computed in [36] to be approximately 7.5×10^{-5}. Taking as p_0 the average of the expected failure probabilities given in Table 1, and using Equation 1, we find that for our system to be reliable it can spend **only 2%** of the total execution time in level 1. Recall that error-correction is the most frequently pe-

Par Xfer	Adder Size	L1 SpeedUp	L2 SpeedUp	Adder SpeedUp	Area Reduced	Gain Product
Steane $[[7,1,3]]$ Code						
10	256	17.417	0.98	6.25	5.07	31.68
	512	17.41	0.97	6.33	6.06	38.38
	1024	18.18	0.88	**4.93**	**9.14**	**45.06**
5	256	10.409	0.98	4.05	5.07	24.99
	512	10.408	0.97	4.04	6.06	24.48
	1024	10.96	0.88	**2.94**	**9.14**	**26.87**
Bacon-Shor $[[9,1,3]]$ Code						
10	256	9.61	1.53	5.92	7.43	43.99
	512	9.61	2.28	8.82	8.87	78.23
	1024	10.15	2.00	**8.10**	**13.4**	**108.53**
5	256	5.17	1.53	3.66	7.43	27.19
	512	5.17	2.28	5.45	8.87	48.37
	1024	5.49	2.00	**4.99**	**13.40**	**66.90**

Table 5. This table shows the results of incorporating a memory hierarchy and two separate encoding levels. Depending on the number of parallel transfers possible between memory and cache, we can expect different speedup values for the adder at level 1. This combined with results from Table 5.1 give us the final Gain Product. Comparatively, prior work has an Gain Product number of 1.0.

formed operation in the CQLA. For the Steane code, level 2 error correction takes 0.3 sec and level 1 takes 3.1×10^{-3} sec, which is approximately 1% of the level 2 time. Thus if all operations performed by the CQLA were equally divided between level 1 and level 2 operations, the system will maintain its fidelity. The Bacon-Shor ECC can be analyzed in a similar manner and their results are more favourable due to a higher threshold.

The CQLA architecure now consists of a memory at level 2, a compute region also at level 2, a cache and a compute region at level 1 and transfer networks for changing the qubit encoding levels. Since quantum modular exponentiation is perfomed by repeated quantum additions, we could perform half of these additions completely in level 2 and the other half in level 1. To comfortably maintain the fidelity of the system, we perform one level 1 addition for every two level 2 additions. The resulting increase in performance is shown in Table 5.

6 Application Behavior

In this secti compute and memory are at level 2 encoding. Contrary to traditional silicon based processors, in the CQLA a single communication step does not take longer than the computation of a single gate. The reason behind this phenomenon is the lack of reliability of quantum data, which forces us to perform an error-correction procedure after each gate. The time to complete a fault-tolerant Toffoli is about 20 times greater than a two-qubit CNOT gate. The applications we study are modular exponentiation and the quantum fourier transform.

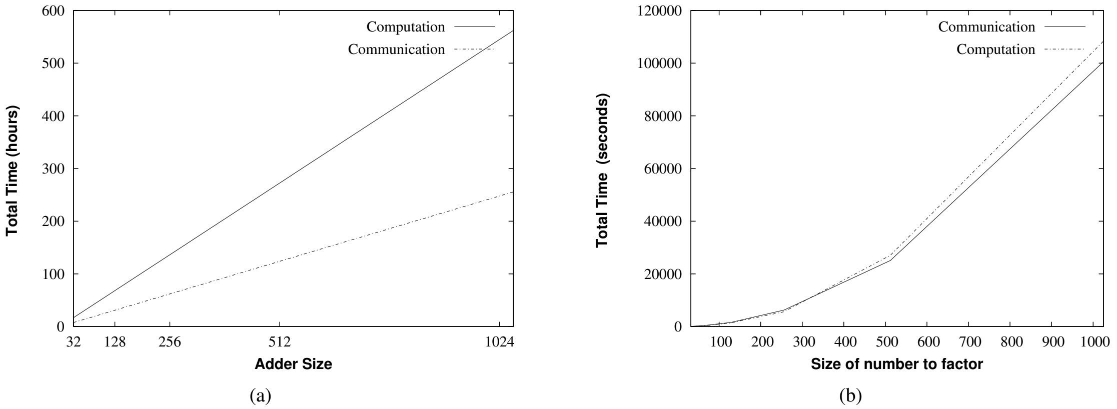
6.1 Shor's Algorithm

Shor's algorithm is the most celebrated of quantum algorithms due to its potential exponential advantage over conventional algorithms and its application to breaking public-key cryptography [31]. Shor's algorithm is primarily composed of two parts, the modular exponentiation and the quantum fourier transform.

Modular Exponentiation: The execution of modular exponentiation is dominated by Toffoli gates. To keep the compute block from having to wait for qubits, and hence stalling, the bandwidth around the perimeter of the compute block has to accommodate the transfer of three qubits to and from memory. Intuitively, since the CQLA is a mesh, and the bottleneck in bandwidth will be at the edge of the compute blocks, having adequate bandwidth at this edge is sufficient for the rest of the mesh.

Based on the communication results from [1], we calculate that a 2 channels on the perimeter of the compute block would provide adequate bandwidth for all required communication. We compute the time required for all communication steps and compare it against the total computation time for differently sized adders. The result is shown in 8(a) and demonstrates that communication requirements do not adversely impact the design.

Quantum Fourier Transform: While the Quantum Fourier Transform (QFT) comprises a small fraction of the overall Shor's algorithm, it requires all-to-all personalized



(a)

(b)

Figure 8. Total communication and computation times for the two components of Shor's algorithm, **(a)** Modular Exponentiation **(b)** Quantum Fourier Transform (QFT). Although communication is significant in the QFT, Modular exponentiation dominates Shor's algorithm. Both these results are for the Bacon-Shor code

communication between data qubits. In addition, it uses only one-qubit and two-qubit gates which consume much less time. As a result, studying the performance of the QFT gives us an insight into how the CQLA will behave when faced with an communication heavy and a computation light application.

In the worst case, all nine data qubits (maximum capacity of the compute block) would have to be transferred to or from memory simultaneously.

Between compute blocks, the QFT's all-to-all personalized communication must be supported on the CQLA mesh network. We leverage the vast amount of prior work done in studying mesh networks, and employ a near-optimal algorithm proposed in [37]. The total time for communication for varying problem sizes is shown in figure 8(b). Note that while communication time is a little less, it closely tracks the computation time for all problem sizes. This is due to the difference in time to error correct a single logical qubit and the time to transport a single qubit; which stays constant immaterial of the problem size.

7 Future Work

A high-level goal of this work is to build abstractions from which architects and systems designers can examine open issues and help guide the substantial basic science and engineering under investment towards building a scalable quantum computer. The primary focus of our work has been system balance. The driving force in this balance has been application parallelism. A key open issue is the restructuring of quantum algorithms to manage this parallelism in the context of system balance. From an architectural point of view, the most relevant abstract properties are density of functional components, the memory hierarchy and communication bandwidth.

While our work has focused on trapped ions, most scalable technologies will have a similar two-dimensional layout where our techniques can be easily applied. This is because the density is determined by the ratio of data to ancilla rather than physical details of the underlying technology.

For ion-traps, lasers can also be a control issue. We plan to study how our architecture can minimize the number of lasers and minimize the power consumed by each laser, since power is proportional to fanout. Efficiently routing control signals to all electrodes in an ion-trap is a challenging proposition, one that has not yet been considered for large systems. Currently, we perform the whole adder at the fast level 1 encoding or at the level 2 encoding; clever instruction scheduling techniques can allow us to improve performance by reducing granularity.

8 Conclusion

The technologies and abstractions for quantum computing have evolved to an exciting stage, where architects and system designers can attack open problems without intimate knowledge of the physics of quantum devices. We explore the amount of parallelism available in quantum algorithms and find that a specialized architecture can serve our needs very well. The CQLA design is an example where architectural techniques of specialization and balanced system design have led to up to a **13X** improvement in density and a **8X** increase in performance, while preserving fault tolerance. We hope that further application of compiler and system optimizations will lead to even more dramatic gains towards a scalable, buildable quantum computer.

References

[1] T. S. Metodi, D. D. Thaker, A. W. Cross, F. T. Chong, and I. L. Chuang, "A quantum logic array microarchitecture: Scalable quantum data movement and computation," *Proceedings of the 38th International Symposium on Microarchitecture* **MICRO-38**, 2005.

[2] W. Wootters and W. Zurek, "A single quantum cannot be cloned," *Nature* **299**, pp. 802–803, 1982.

[3] A. M. Steane, "Space, time, parallelism and noise requirements for reliable quantum computing," *Fortsch. Phys.* **46**, pp. 443–458, 1998.

[4] D. Bacon, "Operator quantum error correcting subsystems for self-correcting quantum memories," *quant-ph/0506023* , 2005.

[5] D. Poulin, "Stabilizer formalism for operator quantum error correction," *quant-ph/0508131* , 2005.

[6] P. Aliferis *Unpublished work based on private conversations with Andrew Cross.* , 2006.

[7] M. Oskin, F. Chong, and I. Chuang, "A practical architecture for reliable quantum computers," *IEEE Computer* , January 2002.

[8] W. Dur, H. J. Briegel, J. I. Cirac, and P. Zoller, "Quantum repeaters based on entanglement purification," *Phys. Rev.* **A59**, p. 169, 1999.

[9] A. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett* **77**, pp. 793–797, 1996.

[10] J. I. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Phys. Rev. Lett* **74**, pp. 4091–4094, 1995.

[11] A. Sorensen and K. Molmer, "Entanglement and quantum computation with ions in thermal motion," *Phys. Lett. A* **62**, p. 02231, 2000.

[12] D. Leibfried and et al., "Experimental demonstration of a robust, high-fidelity geometric two ion-qubit phase gate," *Nature* **422**, pp. 412–415, 2003.

[13] E. Hahn, "Spin echoes," *Phys. Rev.* **80**, pp. 580–594, 1950.

[14] D. Wineland and et al., "Experimental issues in coherent quantum-state manipulation of trapped atomic ions," *Journal of Research of NIST* **103**, pp. 259–328, 1998.

[15] D. Kielpinski, C. Monroe, and D. Wineland, "Architecture for a large-scale ion-trap quantum computer," *Nature* **417**, pp. 709–711, 2002.

[16] J. Porto, S. Rolston, T. Laburthe, C. Williams, and W. Phillips, "Quantum information with neutral atoms as qubits," *Phil. Trans. R. Soc. Lond.* **A361**, pp. 1417–1427, 2003.

[17] B. Blinov, L. Deslauriers, P. Lee, M. Madsen, R. Miller, and C. Monroe, "Sympathetic cooling of trapped ions for quantum logic," *Phys. Rev. A.* **61**, p. 032310, 2000.

[18] B. Blinov, L. Deslauriers, P. Lee, M. Madsen, R. Miller, and C. Monroe, "Sympathetic cooling of trapped cd+ isotopes," *Phys. Rev. A.* **65**, p. 040304, 2002.

[19] M. Barrett, J. Chiaverini, T. Schaetz, J. Britton, and et. al., "Deterministic quantum teleportation of atomic qubits," *Nature* **429**, 2004.

[20] M. Riebe, H. Haffner, C. Roos, and et. al., "Deterministic quantum teleportation with atoms," *Nature* **429**(6993), pp. 734–737, 2004.

[21] J. Chiaverini, R. B. J. Britton, J. Jost, C. Langer, D. Leibfried, R. Ozeri, and D. Wineland, "Surface-electrode architecture for ion-trap quantum information processing," *E-Print: quant-ph/0501147* , 2004.

[22] J. Kim, S. Pau, Z. Ma, H. McLellan, J. Gages, A. Kornblit, and R. Slusher, "System design for large-scale ion trap quantum information processor," *Quantum Information and Computation* **5**(**7**), 2005.

[23] D. Wineland and T. Heinrichs, "Ion trap approaches to quantum information processing and quantum computing," *A Quantum Information Science and Technology Roadmap* , 2004. URL: http://quist.lanl.gov.

[24] A. Steane, "How to build a 300 bit, 1 gop quantum computer," *arXiv:quant-ph/0412165* , 2004.

[25] R. Ozeri and et. al., "Hyperfine coherence in the presence of spontaneous photon scattering," *arXiv:quant-ph/0502063* , 2004.

[26] D. Wineland, D. Leibfried, M. Barrett, A. Ben-Kish, and et.al., "Quantum control, quantum information processing, and quantum-limited metrology with trapped ions," *Proceedings of the International Conference on Laser Spectroscopy (ICOLS)* , 2005.

[27] W. K. Hensinger, S. Olmschenk, D. Stick, D. Hucul, M. Yeo, M. Acton, L. Deslauriers, J. Rabchuk, and C. Monroe, "T-junction ion trap array for two-dimensional ion shuttling, storage and manipulation," *E-Arxiv: quant-ph/0508097* , 2005.

[28] S. Balensiefer, L. Kregor-Stickles, and M. Oskin, "An evaluation framework and instruction set architecture for ion-trap based quantum micro-architectures.," *ISCA-32; Madison, WI* , 2005.

[29] R. V. Meter and M. Oskin, "Architectural implications of quantum computing technologies," *ACM Journal on Emerging Technologies in Computing Systems (JETC)* **2**(**1**), 2006.

[30] T. Draper, S. Kutin, E. Rains, and K. Svore, "A logarithmic-depth quantum carry-lookahead adder," *E-Print: quant-ph/0406142* , 2004.

[31] P. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *35th Annual Symposium on Foundations of Computer Science* , pp. 124–134, 1994.

[32] D. Aharonov and M. Ben-Or, "Fault tolerant computation with constant error," *Symposium on Theory of Computing (STOC 1997)* , pp. 176–188.

[33] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2000.

[34] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A* **54**, p. 2493, 1995.

[35] D. Gottesman, "Fault tolerant quantum computation with local gates," *Journal of Modern Optics* **47**, pp. 333–345, 2000.

[36] K. Svore, B. Terhal, and D. DiVincenzo, "Local fault-tolerant quantum computation," *E-Print: quant-ph/0410047* , 2004.

[37] Y.Yang and J.Wang, "Pipelined all-to-all broadcast in all-port mesh and tori," *IEEE Transactions on Computers* **50**(10), pp. 1020–1032, 2001.

A.6 JQE’03 Paper

Toward a Scalable, Silicon-Based Quantum Computing Architecture

Dean Copsey, Mark Oskin, Francois Impens, Tzvetan Metodiev, Andrew Cross, *Student Member, IEEE,*
Frederic T. Chong, Isaac L. Chuang, and John Kubiatowicz, *Member, IEEE*

Invited Paper

***Abstract*—Advances in quantum devices have brought scalable quantum computation closer to reality. We focus on the system-level issues of how quantum devices can be brought together to form a scalable architecture. In particular, we examine promising silicon-based proposals. We discover that communication of quantum data is a critical resource in such proposals. We find that traditional techniques using quantum SWAP gates are exponentially expensive as distances increase and propose quantum *teleportation* as a means to communicate data over longer distances on a chip. Furthermore, we find that realistic quantum error-correction circuits use a recursive structure that benefits from using teleportation for long-distance communication. We identify a set of important architectural building blocks necessary for constructing scalable communication and computation. Finally, we explore an actual layout scheme for recursive error correction, and demonstrate the exponential growth in communication costs with levels of recursion, and that teleportation limits those costs.**

***Index Terms*—Quantum architecture, quantum computers, silicon-based quantum computing.**

I. Introduction

MANY important problems seem to require exponential resources on a classical computer. Quantum computers can solve some of these problems with polynomial resources, which has led a great number of researchers to explore quantum information processing technologies [1]–[7]. Early-stage quantum computers have involved a small number of components (less than ten) and have utilized molecules in solution and trapped ions [8]–[11]. To exploit our tremendous historical investment in silicon, however, solid-state silicon quantum computers are desirable. Promising proposals along these lines have begun to appear [12], [13]; these even include ideas which merge atomic physics and silicon micromachining [14]. However, as the number of components grows, quantum computing systems will begin to require the same level of engineering as current computing systems. The process of architectural design used for classical silicon-based systems, of building abstractions and optimizing structures, needs to be applied to quantum technologies.

Even at this early stage, a general architectural study of quantum computation is important. By investigating the potential costs and fundamental challenges of quantum devices, we can help illuminate pitfalls along the way toward a scalable quantum processor. We may also anticipate and specify important subsystems common to all implementations, thus fostering interoperability. Identifying these practical challenges early will help focus the ongoing development of fabrication and device technology. In particular, we find that transporting quantum data is a critical requirement for upcoming silicon-based quantum computing technologies.

Quantum information can be encoded in a number of ways, such as the spin component of basic particles like protons or electrons, or in the polarization of photons. Thus, there are several ways in which we might transfer information. First, we might physically transport particles from one point to another. In a large solid-state system, the logical candidate for information carriers would be electrons, since they are highly mobile. Unfortunately, electrons are also highly interactive with the environment and, hence, subject to corruption of their quantum state, a process known as *decoherence*. Second, we might consider passing information along a line of quantum devices. This *swapping channel* is, in fact, a viable option for short distances (as discussed in Section IV), but tends to accumulate errors over long distances.

Over longer distances, we need something fundamentally different. We propose to use a technique called *teleportation* [15] and to call the resulting long-distance quantum wire a *teleportation channel* to distinguish from a swapping channel. Teleportation uses an unusual quantum property called *entanglement*, which allows quantum information to be communicated at a

Manuscript received April 27, 2003; revised August 7, 2003. This work was supported in part by the DARPA Quantum Information, Science and Technology Program. The work of M. Oskin was supported in part by NSF CAREER grant. The work of F. Chong was supported in part by a NSF CAREER grant, in part by a University of California at Davis Chancellor's Fellowship, and in part by the NSF Center for Bits and Atoms at MIT. The work of A. Cross was supported in part by an NDSEG Fellowship. This paper appeared in an earlier form in *Proceedings of the International Symposium on Computer Architecture,* 2003, pp. 374–387, and *Proceedings of the Symposium on Parallelism and Architectures,* 2003, pp. 65–74.

D. Copsey, T. Metodiev, and F. T. Chong are with the Department of Computer Science, University of California, Davis, CA 95616 USA (e-mail: copsey@cs.ucdavis.edu).

M. Oskin is with the University of Washington, Seattle, WA 98195–2350 USA.

F. Impens, A. Cross, and I. L. Chuang are with the Massachusetts Institute of Technology Media Laboratory, Cambridge, MA 02139 USA.

J. Kubiatowicz is with the Computer Science Division, University of California, Berkeley, CA 94720-1776 USA.

Digital Object Identifier 10.1109/JSTQE.2003.820922

distance.[1] To understand the mathematical details and practical implications of teleportation, we will need to cover some background before returning to the subject in Section II-C.

A striking example of the importance of quantum communication lies in the implementation of error-correction circuits. Quantum computations must make use of extremely robust error-correction techniques to extend the life of quantum data. We present optimized layouts of quantum error-correction circuits based upon quantum bits embedded in silicon.

We discover two interesting results from our quantum layouts. First, the recursive nature of quantum error correction results in an H-tree-structured circuit that requires long-distance communication to move quantum data as we approach the root. Second, the reliability of the quantum SWAP operator is perhaps the most important operator for a technology to implement reliably in order to realize a scalable quantum computer.

The remainder of this paper continues with a brief introduction to quantum computing in Section II. We describe our assumptions about implementation technologies in Section III. Next, Section IV discusses how quantum information can be transported in solid-state technologies. This includes a discussion of short-distance *swapping channels* and the more scalable long-distance *teleportation channels*. Section V introduces error-correction algorithms for quantum systems and discusses the physical layout of such algorithms. Then, Section VI probes details of two important error-correction codes. Following this, in Section VII, we demonstrate the need for teleportation as a long-distance communication mechanism in the layout of recursive error-correction algorithms. Finally, Section VIII discusses system bandwidth issues and in Section IX we conclude.

II. Quantum Computation

We begin with a brief overview of the basic terminology and constructs of quantum computation. Our purpose is to introduce the language necessary for subsequent sections; in-depth treatments of these subjects are available in the literature [2].

A. Quantum States: Qubits

The state of a classical digital system $\mathcal{X}$ can be specified by a binary string $\mathbf{x}$ composed of a number of bits x_i, each of which uniquely characterizes one elementary piece of the system. For n bits, there are 2^n possible states. The state of an analogous quantum system ψ is described by a complex-valued vector $|\psi\rangle = \sum_x c_x|\mathbf{x}\rangle$, a weighted combination (a "superposition") of the basis vectors $|\mathbf{x}\rangle$, where the *probability amplitudes* c_x are complex numbers whose modulus squared sums to one, $\sum_x |c_x|^2 = 1$.

A single quantum bit is commonly referred to as a *qubit* and is described by the equation $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$, where the c_i are complex valued. Legal qubit states include "classical" computational basis states $|0\rangle$ and $|1\rangle$, and states in superposition, such as $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, or $\frac{1}{2}|0\rangle - i\frac{\sqrt{3}}{2}|1\rangle$. Larger quantum systems can be composed from multiple qubits, for example, $|00\rangle$, or $\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle - \frac{1}{\sqrt{2}}|11\rangle$. An n-qubit state is described by 2^n basis vectors, each with its own complex probability amplitude, so an n-qubit system can exist in an arbitrary superposition of the possible 2^n classical states of the system.

Unlike the classical case, however, where the total can be completely characterized by its parts, the state of larger quantum systems cannot always be described as the product of its parts. This property, known as *entanglement*, is best illustrated with an example: there exist no single qubit states $|\psi_A\rangle$ and $|\psi_B\rangle$ such that the two-qubit state $|\Psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ can be expressed as the composite state[2] $|\psi_A\rangle \otimes |\psi_B\rangle$. Entanglement has no classical analogue. It is what gives quantum computers their computational powers.

Although a quantum system may exist in a superposition of orthogonal states, only one of those states can be observed, or measured. After measurement, the system is no longer in superposition: the quantum state collapses into the one state measured, and the probability amplitude of all other states goes to zero. For example, when the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ is measured, the result is either 00 or 11, with equal probability; the outcomes $|01\rangle$ or $|10\rangle$ never occur. Furthermore, if a subset of the qubits in a system is measured, the remaining qubits are left in a state consistent with the measurement.

Since measurement of a quantum system only produces a single result, quantum algorithms must maximize the probability that the result measured is the result desired. This may be accomplished by iteratively amplifying the desired result, as in Grover's fast database search, $O(\sqrt{n})$ for a dataset of size n [16]. Another option is to arrange the computation such that it does not matter which of many random results is measured from a qubit vector. This method is used in Shor's algorithm for finding a factor of a composite integer [17], [18], which is built upon modular exponentiation and a quantum Fourier transform. For the interested reader, quantum algorithms for a variety of problems other than search and factoring have been developed: adiabatic solution of optimization problems (a quantum analogue of simulated annealing; complexity unknown) [19], precise clock synchronization (using EPR pairs to synchronize GPS satellites) [20], [21], quantum key distribution (provably secure distribution of classical cryptographic keys) [22], and very recently, Gauss sums [23], and Pell's equation [24].

B. Quantum Gates and Circuits

Just as classical bits are manipulated using gates such as NOT, AND, and XOR, qubits are manipulated with quantum gates such as those shown in Fig. 1. A quantum gate is described by a unitary operator U. The output state vector is the operator applied to the input vector; that is, $|\psi_{\text{out}}\rangle = U|\psi_{\text{in}}\rangle$. The classical NOT has the quantum analogue X which inverts the probabilities of measuring 0 and 1. The quantum analogue of XOR is the two-qubit CNOT gate: the *target* qubit is inverted for those states where the *source* qubit is 1. Most quantum gates, however, have no classical analogue. The Z gate flips the relative phase of the $|1\rangle$ state, thus exchanging $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The Hadamard gate H turns $|0\rangle$ into $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and

[1] The speed of this channel is, however, limited by the rate at which two classical bits can be transmitted from source to destination, without which the quantum information is ambiguous.

[2] The composition operator for quantum systems is the tensor product, $\otimes$: $|\mathbf{x}\rangle \otimes |\mathbf{y}\rangle = \sum_x c_x|x\rangle \otimes \sum_y c_y|y\rangle = \sum_{x,y} c_x c_y|x \otimes y\rangle$, where $x \otimes y$ is simply the string formed by concatenating x and y.

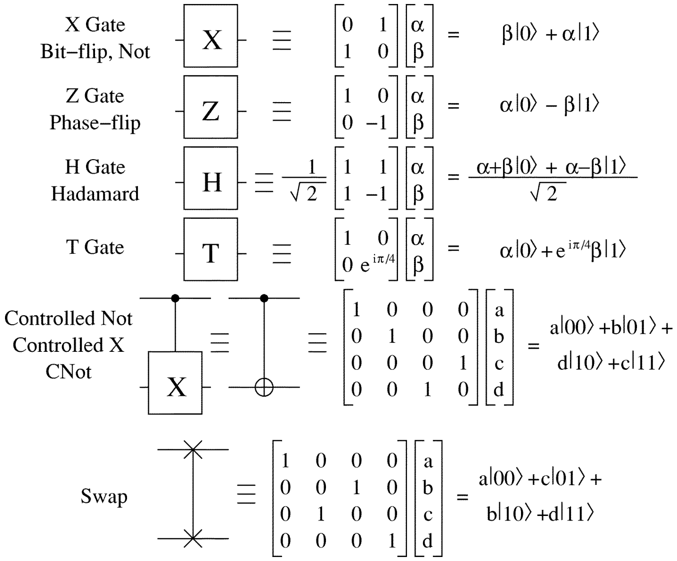

X Gate
Bit−flip, Not
X
$\equiv \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}\begin{bmatrix}\alpha\\\beta\end{bmatrix} = \beta|0\rangle + \alpha|1\rangle$
Z Gate
Phase−flip
Z
$\equiv \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}\begin{bmatrix}\alpha\\\beta\end{bmatrix} = \alpha|0\rangle - \beta|1\rangle$
H Gate
Hadamard
H
$\equiv \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}\alpha\\\beta\end{bmatrix} = \frac{\alpha+\beta|0\rangle + \alpha-\beta|1\rangle}{\sqrt{2}}$
T Gate
T
$\equiv \begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}\begin{bmatrix}\alpha\\\beta\end{bmatrix} = \alpha|0\rangle + e^{i\pi/4}\beta|1\rangle$
Controlled Not
Controlled X
CNot
X
$\equiv \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}a\\b\\c\\d\end{bmatrix} = a|00\rangle + b|01\rangle + d|10\rangle + c|11\rangle$
Swap
$\equiv \begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}a\\b\\c\\d\end{bmatrix} = a|00\rangle + c|01\rangle + b|10\rangle + d|11\rangle$

Fig. 1. Basic quantum gates and their matrix representations.

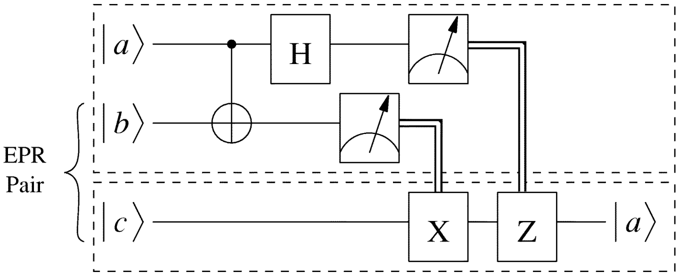


Fig. 2. Quantum Teleportation: Quantum Teleportation of state $|a\rangle$. First, *entangled* qubits $|b\rangle$ and $|c\rangle$ are distributed. Then, $|a\rangle$ is combined with $|b\rangle$ after which measurements produce two *classical* bits of information (double lines). After transport, these bits are used to manipulate $|c\rangle$ to regenerate state $|a\rangle$ at the destination.

$|1\rangle$ into $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$; it can be thought of as performing a radix-2 Fourier transform. Another important single-qubit gate, T, leaves $|0\rangle$ unchanged but multiplies $|1\rangle$ by $\sqrt{i}$. Single qubit gates are characterized by a rotation around an axis: X rotates the qubit by π around the $\hat{x}$-axis; Z rotates by π around the $\hat{z}$-axis; and T rotates by $\pi/4$ around the $\hat{z}$ axis. By composing the T and H gates, any single-qubit gate can be approximated to arbitrary precision. The combination of T, H, and CNOT provide a *universal set*: just as any Boolean circuit can be composed from AND, OR, and NOT gates, any polynomially describable multiqubit quantum transform U can be efficiently approximated by composing just these three quantum gates into a circuit.

One additional important operator is the SWAP gate. Just as two classical values can be swapped using three XORs, a quantum SWAP can be implemented as three CNOTs. However, SWAP is often available natively for a given technology, which is valuable, given its importance to quantum communication.

Fig. 2 shows a *quantum circuit* for teleportation (described in the next section). In quantum circuits, time goes from left to right, where single lines represent qubits, and double lines represent classical bits. A meter is used to represent measurement. By convention, black dots represent control terminals for quantum-controlled gates. The symbol $\oplus$ is shorthand for the target qubit of the CNOT gate.

C. Quantum Teleportation

Quantum teleportation is the recreation of a quantum state at a distance, using only classical communication. It accomplishes this feat by using a pair of entangled qubits, $|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, called an EPR pair.[3]

Fig. 2 gives an overview of the teleportation process. We start by generating an EPR pair. We separate the pair, keeping one qubit, $|b\rangle$, at the source and transporting the other, $|c\rangle$, to the destination. When we want to send a qubit, $|a\rangle$, we first interact $|a\rangle$ with $|b\rangle$ using a CNOT gate. We then measure the phase of $|a\rangle$ and the amplitude of $|b\rangle$, send the two one-bit classical results to the destination, and use those results to recreate the correct phase and amplitude in $|c\rangle$ such that it takes on the original state of $|a\rangle$. The recreation of phase and amplitude is done with X and Z gates, whose application is contingent on the outcome of the measurements of $|a\rangle$ and $|b\rangle$. Intuitively, since $|c\rangle$ has a special relationship with $|b\rangle$, interacting $|a\rangle$ with $|b\rangle$ makes $|c\rangle$ resemble $|a\rangle$, modulo a phase and/or amplitude error. The two measurements allow us to correct these errors and recreate $|a\rangle$ at the destination. Note that the original state of $|a\rangle$ is destroyed when we take our two measurements.[4]

Why bother with teleportation when we end up transporting $|c\rangle$ anyway? Why not just transport $|a\rangle$ directly? First, we can precommunicate EPR pairs with extensive pipelining without stalling computations. Second, it is easier to transport EPR pairs than real data. Since $|b\rangle$ and $|c\rangle$ have known properties, we can employ a specialized procedure known as *purification* to turn a collection of pairs partially damaged from transport into a smaller collection of asymptotically perfect pairs. Third, transmitting the two classical bits resulting from the measurements is more reliable than transmitting quantum data.

III. Solid-State Technologies

With some basics of quantum operations in mind, we turn our attention to the technologies available to implement these operations. Experimentalists have examined several technologies for quantum computation, including trapped ions [26], photons [27], bulk spin NMR [28], Josephson junctions [13], [29], SQUIDS [30], electron spin resonance transistors [31], and phosphorus nuclei in silicon (the “Kane” model) [12], [32]. Of these proposals, only the last three build upon a solid-state platform; they are generally expected to provide the scalability required to achieve a truly scalable computational substrate.

For the purposes of this paper, the key feature of these solid-state platforms are as follows.

1) Quantum bits are laid out in silicon in a two-dimensional (2-D) fashion, similar to traditional CMOS VLSI.
2) Quantum interactions are near-neighbor between bits.

[3] An EPR or Einstein-Podolsky-Rosen pair is a special instance of entanglement noted in the Einstein-Podolsky-Rosen paradox [25], [62].

[4] This is consistent with the *no-cloning* theorem, which states that an arbitrary quantum state cannot be perfectly copied; this is fundamentally because of the unitarity of quantum mechanics.

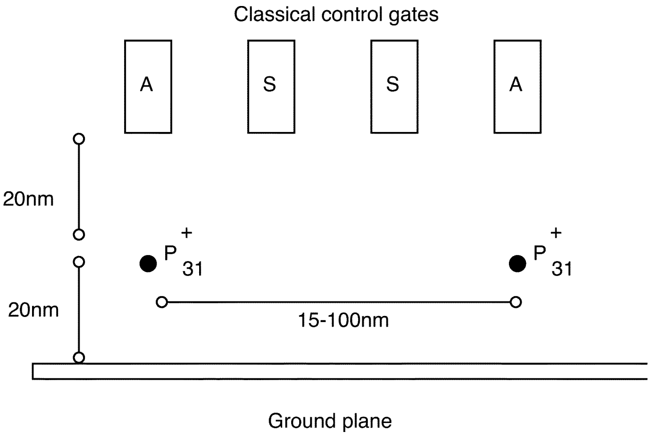


Fig. 3. The basic quantum bit technology proposed by Kane [34]. Qubits are embodied by the nuclear spin of a phosphorus atom coupled with an electron embedded in silicon under high magnetic field at low temperature.

3) Quantum bits cannot move physically, but quantum data can be swapped between neighbors.
4) The control structures necessary to manipulate the bits prevent a dense 2-D grid of bits. Instead, we have linear structures of bits which can cross, but there is a minimum distance between such intersections that is on the order of 20 bits for our primary technology model [33]. This restriction is similar to a "design rule" in traditional CMOS VLSI.

These four assumptions apply to several solid-state technologies. For concreteness, we will focus upon an updated version of Kane's phosphorus-in-silicon nuclear-spin proposal [34]. This scheme will serve as an example for the remainder of the paper, although we will generalize our results when appropriate.

Fig. 3 illustrates important dimensions of the Kane scheme. Shown are two phosphorus atoms spaced 15–100 nm apart. Quantum states are stored in relatively stable electron-donor $(e^- - {}^{31}\mathrm{P}^+)$ spin pairs, where the electron (e) and the donor nucleus (n) have opposite spins. The basis states, $|0\rangle$ and $|1\rangle$ are defined as the superposition states $|0\rangle \equiv |\uparrow_e\downarrow_n\rangle + |\downarrow_e\uparrow_n\rangle$ and $|1\rangle \equiv |\uparrow_e\downarrow_n\rangle - |\downarrow_e\uparrow_n\rangle$. Twenty nanometers above the phosphorus atoms lie three classical control wires, one A gate and two S gates. Precisely timed pulses on these gates provide arbitrary one- and two-qubit quantum gates.

Single qubit operators are composed of pulses on the A-gates, modulating the hyperfine interaction between electron and nucleus to provide Z axis rotations. A globally applied static magnetic field provides rotations around the X axis. By changing the pulse widths, any desired rotational operator may be applied, including the identity operator. Two-qubit interactions are mediated by S-gates, which move an electron from one nucleus to the next. The exact details of the pulses and quantum mechanics of this technique are beyond the scope of this paper and are described in [34].

Particularly apropos to the next few sections of this paper, however, is the interqubit spacing of 15–100 nm. The exact spacing is currently a topic of debate within the physics community, with conservative estimates of 15 nm, and more aggressive estimations of 100 nm. The tradeoff is between noise immunity and difficulty of manufacturing. For our study, we will use a figure (60 nm) that lies between these two. This choice implies that the A and S gates are spaced 20 nm apart. We parameterize our work, however, to generalize for changes in the underlying technology.

The Kane proposal, like all quantum computing proposals, uses classical signals to control the timing and sequence of operations. All known quantum algorithms, including basic error-correction for quantum data, require the determinism and reliability of classical control. Without efficient classical control, fundamental results demonstrating the feasibility of quantum computation do not apply (such as the Threshold Theorem used in Section IV-B.3).

Quantum computing systems display a characteristic tension between computation and communication. Fundamentally, technologies that transport data well do so because they are resistant to interaction with the environment or other quantum bits; on the other hand technologies that compute well do so precisely because they *do* interact. Thus, computation and communication are somewhat at odds.

In particular, atomic-based solid-state technologies are good at providing scalable computation but complicate communication, because their information carriers have nonzero mass. The Kane proposal, for example, represents a quantum bit with the nuclear spin of a phosphorus atom implanted in silicon. The phosphorus atom does not move, hence, transporting this state to another part of the chip is laborious and requires carefully controlled swapping of the states of neighboring atoms. In contrast, photon-based proposals that use polarization to represent quantum states can easily transport data over long distances through fiber. It is very difficult, however, to get photons to interact and achieve any useful computation. Furthermore, transferring quantum states between atomic- and photon-based technologies is currently extremely difficult.

Optimizing these tensions, between communication and computation, between classical control and quantum effects, implies a structure to quantum systems. In this paper, we begin to examine this optimization by focusing on communication in solid-state quantum systems. Specifically, we begin by examining the quantum equivalent of short and long "wires."

IV. Transporting Quantum Information: Wires

In this section, we explore the difficulty of transporting quantum information within a silicon substrate. Any optimistic view of the future of quantum computing includes enough interacting devices to introduce a spatial extent to the layout of those devices. This spatial dimension, in turn, introduces a need for wires. One of the most important distinctions between quantum and classical wires arises from the *no-cloning* theorem [2] is that quantum information cannot be copied but must rather be *transported* from source to destination (see footnote 4).

Section IV-A begins with a relatively simple means of moving quantum data via swap operations, called a *swapping channel*. Unfortunately, the analysis of Section IV-B indicates that swapping channels do not scale well, leading to an alternative called a *teleportation channel*. This long-distance

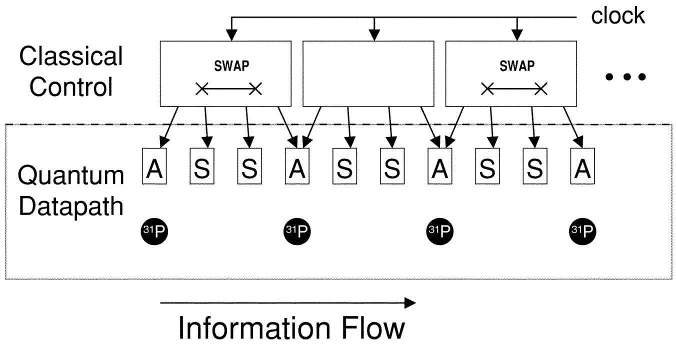


Fig. 4. Short wires are constructed from successive qubits (phosphorus atoms). Information in the quantum data path is swapped from qubit to qubit under classical control. A single SWAP operator requires multiple A- and S-gate voltage pulses. The control circuitry is not to scale.

technology is introduced in Section IV-C and analyzed in Section IV-D.

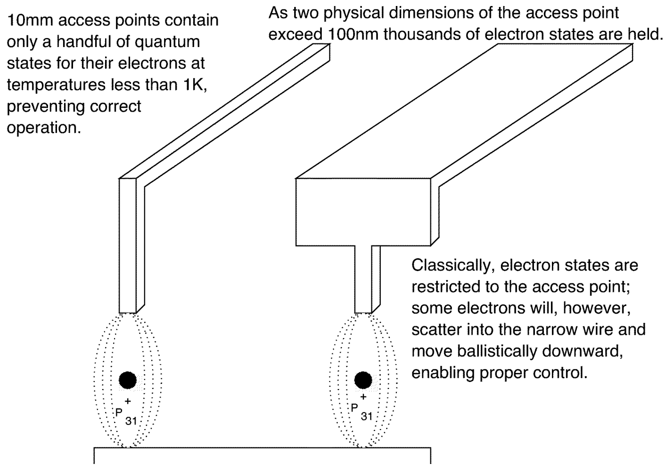
A. Short Wires: Swapping Channel

In solid-state technologies, a line of qubits is one plausible approach to transporting quantum data. Fig. 4 provides a schematic of a *swapping channel* in which information is progressively swapped between pairs of qubits in the *quantum datapath*—somewhat like a bubble sort.[5] Swapping channels require active control from classical logic as illustrated by the *classical control* plane of Fig. 4.

As simple as it might appear, a quantum swapping channel presents significant technical challenges. The first hurdle is the placement of the phosphorus atoms themselves. The leading work in this area has involved precise ion implantation through masks, and manipulation of single atoms on the surface of silicon [35]. For applications where only a few trial devices are desired, slowly placing a few hundred thousand phosphorus atoms with a probe device [36] may be possible. For bulk manufacturing, the advancement of DNA-based or other chemical self-assembly techniques [37] may need to be developed. Note that, while new technologies may be developed to enable precise placement, the key for our work is only the spacing (60 nm) of the phosphorus atoms themselves, and the number of control lines (three) per qubit. The relative scale of quantum interaction and the classical control of these interactions is what will lead our analysis to the fundamental constraints on quantum computing architectures.

A second challenge is the scale of classical control. Each control line into the quantum datapath is roughly 10 nm in width. While such wires are difficult to fabricate, we expect that either electron beam lithography [38], or phase-shifted masks [39] will make such scales possible.

A remaining challenge is the temperature of the device. In order for the quantum bits to remain stable for a reasonable period of time the device must be cooled to less than one degree Kelvin. The cooling itself is straightforward, but the effect of the cooling on the classical logic is a problem. Two issues arise. First, conventional transistors stop working as the electrons become trapped near their dopant atoms, which fail to ionize. Second, the 10-nm classical control lines begin to exhibit quantum-mechanical behavior, such as conductance quantization and interference from ballistic transport [40].


10mm access points contain only a handful of quantum states for their electrons at temperatures less than 1K, preventing correct operation.

As two physical dimensions of the access point exceed 100nm thousands of electron states are held.

Classically, electron states are restricted to the access point; some electrons will, however, scatter into the narrow wire and move ballistically downward, enabling proper control.

Fig. 5. Quantization of electron states overcome by increasing the physical dimension of the control lines beyond 100 nm. The states propagate quantum-mechanically downward through access vias to control the magnetic field around the phosphorus atoms.

Fortunately, many researchers are already working on low-temperature transistors. For instance, single-electron transistors (SETs) [41] are the focus of intense research due to their high density and low power properties. SETs, however, have been problematic for conventional computing because they are sensitive to noise and operate best at low temperatures. For quantum computing, this predilection for low temperatures is exactly what is needed! Tucker and Shen describe this complementary relationship and propose several fabrication methods in [42].

On the other hand, the quantum-mechanical behavior of the control lines presents a subtle challenge that has been mostly ignored to-date. At low temperatures, and in narrow wires, the quantum nature of electrons begins to dominate over normal classical behavior. For example, in 100-nm-wide polysilicon wires at 100 mK, electrons propagate ballistically like waves, through only one conductance channel, which has an impedance given by the quantum of resistance, $h/e^2 \approx 25\ \mathrm{k}\Omega$. Impedance mismatches to these and similar metallic wires make it impossible to properly drive the ac current necessary to perform qubit operations, in the absence of space-consuming impedance matching structures such as adiabatic tapers.

Avoiding such limitations mandates a geometric design constraint: narrow wires must be short and locally driven by nearby wide wires. Using 100 nm as a rule of thumb[6] for a minimum metallic wire width sufficient to avoid undesired quantum behavior at these low temperatures, we obtain a control gate structure such as that depicted in Fig. 5. Here, wide wires terminate in 10-nm vias that act as local gates above individual phosphorus atoms.

[5] For technologies that do not have an intrinsic swap operation, one can be implemented by three CONTROLLED-NOT gates performed in succession. This is a widely known result in the quantum computing field and we refer the interested reader to [2].

[6] This value is based on typical electron mean free path distances, given known scattering rates and the electron Fermi wavelength in metals.

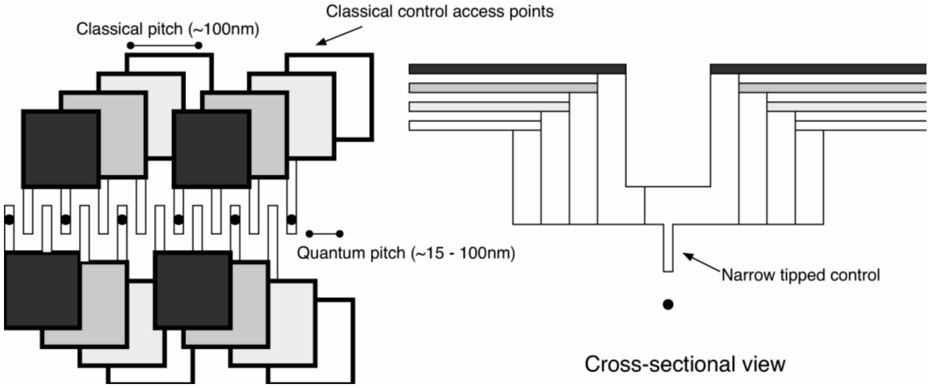


Fig. 6. A linear row of quantum bits: In this figure (not drawn to scale) we depict access control for a line of quantum bits. On the left, we depict a "top down" view. On the right is a vertical cross-section which more clearly depicts the narrow-tipped control lines that quickly expand to classical dimensions.

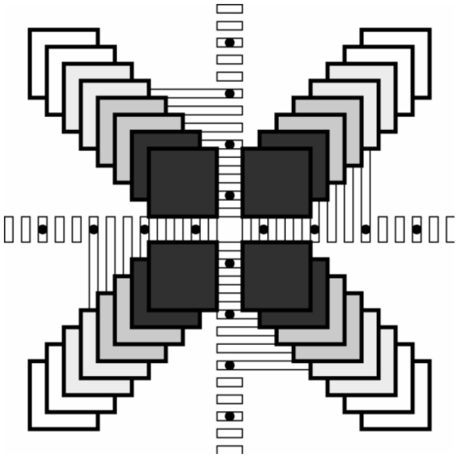


Fig. 7. Intersection of quantum bits. In this simplified view, we depict a four-way intersection of quantum bits. An inversely (diamond shaped) organized junction is also needed to densely pack junction cells.

Producing a line of quantum bits that overcomes all of the above challenges is possible. We illustrate a design in Fig. 6. Note how access lines quickly taper into upper layers of metal and into control areas of a classical scale. These control areas can then be routed to access transistors that can gate on and off the frequencies (in the 10s to 100s of MHz) required to apply specific quantum gates.

Of course, any solution for data transport must also support routing. Routing is not possible without fanout provided by wire intersections. We can extend our linear row of quantum bits to a four-way intersection capable of supporting sparsely intersecting topologies of quantum bits. We illustrate the quantum intersection in Fig. 7. This configuration is similar to Fig. 6 except that the intersection creates a more challenging tapering.

B. Analysis of the Swapping Channel

We now analyze our swapping channel to derive two important architectural constraints: the classical-quantum interface boundary and the latency–bandwidth characteristics. We strive to achieve a loose lower bound on these constraints for a given quantum device technology. While future quantum technologies may have different precise numbers, it is almost certain they will continue to be classically controlled and, thus, also obey similar constraints based upon this classical-quantum interface.

1) Pitch Matching: Our first constraint is derived from the need to have classical control of our quantum operations. As previously discussed, we need a minimum wire width to avoid quantum effects in our classical control lines. Referring back to Fig. 7, we can see that each quadrant of our four-way intersection will need to be some minimum size to accommodate access to our control signals.

Recall from Fig. 3 that each qubit has three associated control signals (one A and two S gates). Each of these control lines must expand from a thin 10 nm tip into a 100 nm access point in an upper metal layer to avoid the effects of charge quantization at low temperatures (Fig. 5). Given this structure, it is possible to analytically derive the minimum width of a line of qubits and its control lines, as well as the size of a four-way intersection. For this minimum size calculation, we assume all classical control lines are routed in parallel, albeit spread across the various metal layers. This parallel nature makes this calculation trivial under normal circumstances (sufficiently "large" lithographic feature size λ_c), with the minimum line segment being equal in length to twice the classical pitching, 150 nm in our case, and the junction size equal to four times the classical pitching, 400 nm, in size. However, we illustrate the detailed computation to make the description of the generalization clearer. We begin with a line of qubits.

Let N be the number of qubits along the line segment. Since there are three gates (an A and two S lines), we need to fit in $3N$ classical access points of 100 nm in dimension each in line width. We accomplish this by offsetting the access points in the x and y dimensions (Fig. 6) by 20 nm. The total size of these offsets will be 100 nm divided by the qubit spacing 60 nm times the number of control lines per qubit (three), times the offset distance of 20 nm. This number 100 nm/60 nm $\times$ 3 $\times$ 20 nm = 100 nm is divided by 2 because the access lines are spread out on each side of the wire. Hence, the minimum line segment will be 100 nm + 50 nm. Shorter line segments within larger, more specialized cells are possible.

Turning our attention to an intersection (Fig. 7), let N be the number of qubits along each "spoke" of the junction. We need to fit $3N$ classical access points in a space of (60 nm $\times N)^2$, where each access point is at least 100 nm on a side. As with the case of a linear row of bits, a 20-nm x and y shift in access point positioning between layers is used for via access. Starting with a single access pad of 100 nm, we must fit 100 nm/60 nm $\times$ 3 additional pads shifted in x and y within the single quadrant of our intersection. This leads to a quadrant size of 100 + 100 nm/60 nm $\times$ 3 $\times$ 20 nm = 200 nm. Therefore, the minimum size four-way intersection is eight (rounding up) qubits in each direction.

In this construction, we have assumed a densely packed edge to each spoke. However, this is easily "unpacked" with a specialized line segment, or by joining to another junction that is constructed inversely from that shown in Fig. 7. Obviously, the specific sizes will vary according to technological parameters and assumptions about control logic, but this calculation illustrates the approximate effect of what appears to be a fundamental tension between quantum operations and the classical signals that control them. A minimum intersection size implies minimum wire lengths, which imply a minimum size for computation units.

2) Technology Independent Limits: Thus far, we have focused our discussion on a particular quantum device technology. This has been useful to make the calculations concrete. Nevertheless, it is useful to generalize these calculations to future quantum device technologies. Therefore, we parameterize our discussion based on a few device characteristics as follows.

Assuming 2-D devices (i.e., not a cube of quantum bits), let p_c be the classical pitching required, and p_q the quantum one. Furthermore, let R be the ratio p_c/p_q of the classical to quantum distance for the device technology, m be the number of classical control lines required per quantum bit, and finally λ_c be the feature size of the lithographic technology. We use two separate variables p_c and λ_c to characterize the "classical" technology because they arise from different physical constraints. The parameter λ_c comes from the lithographic feature size, while p_c (which is a function of λ_c) is related to the charge quantization effect of electrons in gold. With the Kane technology we assume a spacing p_q of 60 nm between qubits, three control lines per bit of 100 nm (p_c) each, and a λ_c of 5 nm. We can use these to generalize our pitch matching equations. Here, we find that the minimum line segment is simply equivalent to $R(1+2\lambda_c m/p_q)$ qubits in length.

Examining our junction structure (Fig. 7), we note that it is simply four line segments, similar to those calculated above, except that the control lines must be on the same side. Therefore, the minimum crossing size of quantum bits in a 2-D device is of size $\approx 2R(1+4\lambda_c m/p_q)$ on a side.

3) Latency and Bandwidth: Calculating the latency and bandwidth of quantum wires is similar to but slightly different than it is for classical systems. The primary difficulty is decoherence (i.e., quantum noise). Unlike classical systems, if you want to perform a quantum computation, you cannot simply resend quantum information when an error is detected. The no-cloning theorem prohibits transmission by duplication, thereby making it impossible to retransmit quantum information if it is corrupted. Once the information is destroyed by the noisy channel, you have to start the entire computation over ("no-cloning" also implies no checkpointing of intermediate states in a computation). To avoid this loss, qubits are encoded in a sufficiently strong error-correcting code that, with high probability, will remain coherent for the entire length of the quantum algorithm. Unfortunately, quantum systems will likely be so error-prone that they will probably execute right at the limits of their error tolerance [43].

Our goal is to provide a quantum communication layer which sits below higher level error-correction schemes. Later, in Section VIII, we discuss the interaction of this layer with quantum error correction and algorithms. Consequently, we start our calculation by assuming a channel with no error correction. Then, we factor in the effects of decoherence and derive a maximum wire length for our line of qubits.

Recall that data traverses the line of qubits with SWAP gates, each of which takes approximately 1 μs to execute in the Kane technology. Hence, to move quantum information over a space of 60 nm requires 0.57 μs. A single row of quantum bits has latency

$$t_{\text{latency}} = d_{\text{qubits}} \times 1\ \mu\text{s} \tag{1}$$

where d_{qubits} is the distance in qubits, or the physical distance divided by 60 nm. This latency can be quite large. A short 1 μm has a latency of 17 μs. On the plus side, the wire can be fully pipelined and has a sustained bandwidth of $1/1\ \ \mu\text{s}\ \ =\ \ 1$ one million quantum bits per second (Mqbps). This may seem small compared to a classical wire, but keep in mind that quantum bits can enable algorithms with exponential speedup over the classical case.

The number of error-free qubits is actually lower than this physical bandwidth. Noise, or decoherence, degrades quantum states and makes the true bandwidth of our wire less than the physical quantum bits per second. Bits decohere over time, so longer wires will have a lower bandwidth than shorter ones.

The stability of a quantum bit decreases with time (much like an uncorrected classical bit) as a function e^{-kt}. Usually, a normalized form of this equation is used, $e^{-\lambda t}$, where t in this new equation is the number of operations and λ is related to the time per operation and the original k. As quantum bits traverse the wire, they arrive with a fidelity that varies inversely with latency, namely

$$\text{fidelity} = e^{-\lambda t_{\text{latency}}}. \tag{2}$$

The true bandwidth is proportional to the fidelity

$$\text{bandwidth}_{\text{true}} = \text{bandwidth}_{\text{physical}} \times \text{fidelity}. \tag{3}$$

Choosing a reasonable[7] value of $\lambda = 10^{-6}$, we find the true bandwidth of a wire to be

$$\frac{1}{1\ \mu\text{s}} e^{-10^{-6} \times d_{\text{qubits}}} \tag{4}$$

which for a 1 μm wire is close to the ideal (999 983 qbps).

This does not seem to be a major effect, until you consider an entire quantum algorithm. Data may traverse back and forth

[7] This value for λ is calculated from a decoherence rate of 10^{-6} per operation, where each operation requires 1 μs. It is aggressive, but potentially achievable with phosphorus atoms in silicon [32], [44].

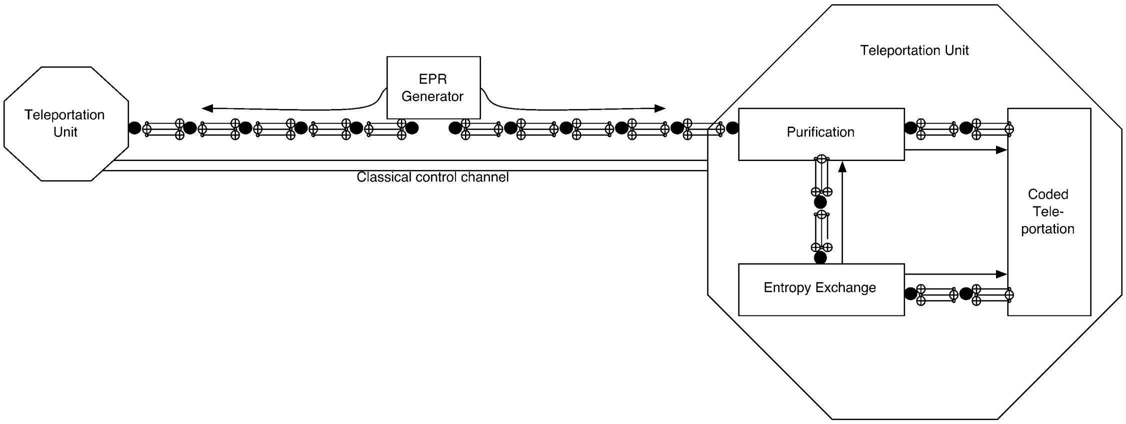


Fig. 8. Architecture for a Quantum Wire: Solid double lines represent classical communication channels, while chained links represented a quantum swapping channel. Single lines depict the direction in which the swapping channel is being used for transport.

across a quantum wire millions of times. It is currently estimated [45] that a degradation of fidelity more than 10^{-4} makes arbitrarily long quantum computation theoretically unsustainable, with the practical limit being far higher [43]. This limit is derived from the Threshold Theorem, which relates the decoherence of a quantum bit to the complexity of correcting this decoherence (as discussed in detail, in Section V) [45]–[47].[8] Given our assumptions about λ, the maximum theoretical wire distance is about 6 μm.

4) Technology Independent Metrics: Our latency and bandwidth calculations require slightly more device parameters. Let t_{swap} be the time per basic SWAP operation. Some technologies will have an intrinsic SWAP, and others will require synthesizing the SWAP from 3 CNOT operations. Let λ be the decoherence rate, which for small λ and t_{swap} is equivalent to the decoherence a quantum bit undergoes in a unit of operation time t_{swap}. This makes the latency of a swapping channel wire equal to

$$t_{\text{latency}} = d_{\text{qubits}} t_{\text{swap}} \tag{5}$$

where the distance d_{qubits} is expressed in the number of qubits. The bandwidth is proportional to the fidelity or:

$$\text{bandwidth}_{\text{true}} = \frac{1}{t_{\text{swap}}} e^{-\lambda d_{\text{qubits}}}. \tag{6}$$

This bandwidth calculation is correct so long as the fidelity remains above the critical threshold $C \approx 10^{-4}$ required for fault tolerant computation. Finally, the maximum distance of this swapping channel is the distance when the fidelity drops below the critical threshold

$$d_{\text{qubits,max}} = \frac{\ln(1-C)}{-\lambda}. \tag{7}$$

No amount of error correction will be robust enough to support a longer wire, while still supporting arbitrarily long quantum computation. For this, we need a more advanced architecture. One obvious option is to break the wire into segments and insert "repeaters" in the middle. These quantum repeaters are effectively performing state restoration (error correction). However, we can do better, which is the subject of the next section.

C. Long Wires: Teleportation Channel

In this section, we introduce an architecture for quantum communication over longer distances in solid-state technologies, shown in Fig. 8. This architecture makes use of the quantum primitive of teleportation (described earlier in Section II-C). In the next few sections, we provide a brief introduction to the core components of this architecture.

Although teleportation and the mechanisms described in this section are known in the literature, what has been missing is the identification and analysis of which mechanisms form fundamental building blocks of a realistic system. In this section, we highlight three important architectural building blocks: the *entropy exchange unit*, the *EPR generator*, and the *purification unit*. Note that the description of theses blocks is quasi-classical in that it involves input and output ports. Keep in mind, however, that all operations (except measurement) are inherently reversible, and the specification of input and output ports merely provides a convention for understanding the forward direction of computation.

1) Entropy Exchange Unit: The physics of quantum computation requires that operations are reversible and conserve energy. The initial state of the system, however, must be created somehow. We need to be able to create $|0\rangle$ states. Furthermore, decoherence causes qubits to become randomized—the entropy of the system increases through qubits coupling with the external environment.

Where do these zero states come from? The process can be viewed as one of thermodynamic cooling. "Cool" qubits are distributed throughout the processor, analogous to a ground plane in a conventional CMOS chip. The "cool" qubits are in a nearly zero state. They are created by measuring the qubit, and inverting if $|1\rangle$. The measurement process itself requires a source of cold spin-polarized electrons (created, for example, using a standard technique known as optical pumping [44], [48]).

As with all quantum processes, the measurement operation is subject to failure but, with high probability, leaves the measured

[8] By "practical," we mean without an undue amount of error correction. The threshold theorem ensures that, theoretically, we can compute arbitrarily long quantum computations, but the practical overhead of error correction makes the real limit 2–3 orders of magnitude higher [43].

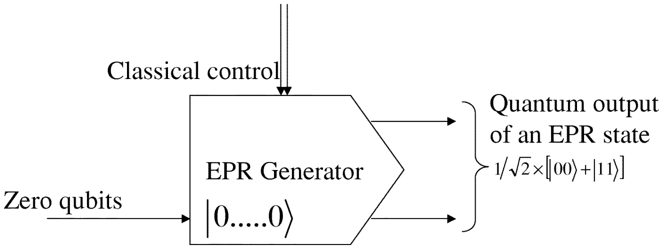


Fig. 9. Quantum EPR generator. Solid double lines represent classical communication (or control), and single lines depict quantum wires.

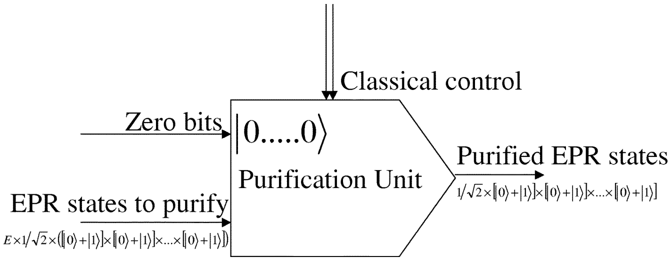

Classical control
Zero bits
$|0.....0\rangle$
Purification Unit
EPR states to purify
$E\times1/\sqrt{2}\times([|0\rangle+|1\rangle]\times[|0\rangle+|1\rangle]\times...\times[|0\rangle+|1\rangle])$
Purified EPR states
$1/\sqrt{2}\times[|0\rangle+|1\rangle]\times[|0\rangle+|1\rangle]\times...\times[|0\rangle+|1\rangle]$

Fig. 10. Quantum purification unit. EPR states are sufficiently regular that they can be purified at the ends of a teleportation channel.

qubit in a known state from which $|0\rangle$s may be obtained. To arbitrarily increase this probability (and make an extremely cold zero state) we can use a technique called *purification*. Specifically, one realization employs an efficient algorithm for data compression [49], [50] that gathers entropy across a number of qubits into a small subset of high-entropy qubits. As a result, the remaining qubits are reinitialized to the desired pure, $|0\rangle$ state.

2) EPR Generator: Constructing an EPR pair of qubits is straightforward. We start with two $|0\rangle$ state qubits from our entropy exchange unit. A Hadamard gate is applied to the first of these qubits. We then take this transformed qubit that is in an equal superposition of a zero and a one state and use it as the control qubit for a CNOT gate. The target qubit that is to be inverted is the other fresh $|0\rangle$ qubit from the entropy exchange unit. A CNOT gate is a qubit like a classical XOR gate in that the target qubit is inverted if the control qubit is in the $|1\rangle$ state. Using a control qubit of $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and a target qubit of $|0\rangle$ we end up with a two-qubit entangled state of $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$: an EPR pair.

The overall process of EPR generation is depicted in Fig. 9. Schematically, the EPR generator has a single quantum input and two quantum outputs. The input is directly piped from the entropy exchange unit and the output is the entangled EPR pair.

3) EPR Purification Unit: The final building block we require is the EPR purification unit. This unit takes as input n EPR pairs, which have been partially corrupted by errors, and outputs nE asymptotically perfect EPR pairs. E is the entropy of entanglement, a measure of the number of quantum errors which the pairs suffered. The details of this entanglement purification procedure are beyond the scope of this paper but the interested reader can see [51]–[53].

Fig. 10 depicts a purification block. The quantum inputs to this block are the input EPR states and a supply of $|0\rangle$ qubits. The outputs are pure EPR states. Note that the block is carefully designed to correct only up to a certain number of errors; if more errors than this threshold occur, then the unit fails with increasing probability.

Fig. 8 illustrates how we use these basic building blocks and protocols for constructing our teleportation channel. The EPR generator is placed in the middle of the wire and "pumps" entangled qubits to each end (via a pipelined swapping channel). These qubits are then purified such that only the error-free qubits remain. Purification and teleportation consume zero-state qubits that are supplied by the entropy exchange unit. Finally, the coded-teleportation unit transmits quantum data from one end of the wire to the other using the protocol described in Section II-C. Our goal now is to analyze this architecture and derive its bandwidth and latency characteristics.

D. Analysis of the Teleportation Channel

The bandwidth of a teleportation channel is proportional to the speed with which reliable EPR pairs are communicated. Since we are communicating unreliable pairs, we must purify them, so the efficiency of the purification process must be taken into account. Purification has an efficiency roughly proportional to the fidelity of the incoming, unpurified qubits [49]

$$\text{purification}_{\text{efficiency}} \approx \text{fidelity}^2. \tag{8}$$

Entropy exchange is a sufficiently parallel process that we assume enough zero qubits can always be supplied. Therefore, the overall bandwidth of this long quantum wire is

$$\frac{1}{1\ \mu\text{s}} \times e^{-2\times10^{-6}\times d_{\text{qubits}}} \tag{9}$$

which for a 1-μm wire is 999 967 qbps. Note that this result is less than for the simple wiring scheme, but the decoherence introduced on the logical qubits is only $O(e^{-\lambda\times10})$. It is this latter number that does not change with wire length which makes an important difference. In the previous short-wire scheme we could not make a wire longer than 6 μm. Here we can make a wire of arbitrary length. For example, a 10-mm-long wire has a bandwidth of 716 531 qbps, while a simple wire has an effective bandwidth of zero at this length (for computational purposes).

The situation is even better when we consider latency. Unlike the simple wire, the wire architecture we propose allows for the precommunication of EPR pairs at the sustainable bandwidth of the wire. These precommunicated EPR pairs can then be used for transmission with a constant latency. This latency is roughly the time it takes to perform teleportation, or $\approx$ 20 μs. Note that this latency is much improved compared with the distance-dependent simple wiring scheme.

Using the same constants defined above for the swapping channel, we can generalize our analysis of teleportation channels. The latency is simply

$$t_{\text{latency}} \approx 10\ t_{\text{swap}} \tag{10}$$

The bandwidth is

$$\text{bandwidth}_{\text{true}} = \frac{1}{t_{\text{swap}}} e^{-2\lambda d_{\text{qubits}}}. \tag{11}$$

Unlike the short wire, this bandwidth is *not* constrained by a maximum distance related to the Threshold Theorem since teleportation is unaffected by distance. The communication of EPR pairs before teleportation, however, can be affected by distance, but at a very slow rate. While purification must discard more corrupted EPR pairs as distance increases, this effect is orders-of-magnitude smaller than direct data transmission over short wires and is not a factor in a practical silicon chip of up to tens of millimeters on a side.

V. Fault-Tolerant Architecture and Geometric Constraints

We turn now to a key *system* requirement for quantum computing. The ability to tolerate and dynamically handle internal faults while preserving the integrity of the computation. Unlike present classical computing systems, where the gate failure probability is extremely low (mosfets in CMOS fail with probability lower than 10^{-16} per operation), current and projected quantum gates have $O(10^{-1})$ to $O(10^{-7})$ probabilities of failure per operation,

Nevertheless, as was mentioned in the introduction, a main result in the field is that by using a construction involving fine-grained fault tolerance, an arbitrarily reliable quantum information processor can be efficiently constructed using unreliable components [45].

In this section, we study geometric constraints on scalable, quantum fault-tolerant construction. Key to our study of quantum wires was a tradeoff between the geometric design of the system and the noise generated during operation: shrinking the wires exposes quantum effects in conductivity and voltage, and lowers the fidelity of the operations performed on the qubit. Allocating more space allows us to reduce the noise; however, there is a different way to use this spatial resource. Instead of making larger gates or wires, space can alternatively be used to perform computations using a fault-tolerant quantum circuit, employing redundant, faulty quantum gates. These two strategies for achieving reliable computation, either at the cost of larger devices, or at the cost of more area for redundant circuits, present different tradeoffs between space and reliability.

In the remainder of this section, we present an explicit analytical mathematical formula capturing this tradeoff, and demonstrate some global geometric bounds on fault-tolerant quantum computation. We begin in Section V-A with an overview of the fault tolerance, which is then described in detail in terms of quantum error correction (Section V-B); how to compute on encoded data (Section V-C); and how to do so recursively (Section V-D). We then introduce our reliability model in Section V-E, and describe how geometry is involved. This, then, leads to our main result of this section, in Section V-F.

A. *Overview of Quantum Fault-Tolerant Strategy*

In order for a system to operate reliably despite a partial corruption of the data it processes, it must introduce a certain amount of redundancy in the form of an error-correction code. This protection can only be effective if the redundancy is present at all times in the computational process. All operators need to be consistently modified as to compute directly on encoded data. The choice of a code is dictated by three criteria. First, it should minimize the complexity overhead due to the aforementioned modification of the circuit. Second, the concentration of redundancy should be focused around strategic operators, whose erroneous behavior is likely to occur and critical for the computation. Third, and this is a general requirement in coding theory, the code should raise a syndrome allowing an identification or/and correction of the expected errors. The freedom in encoding differs in a quantum and in a classical context. The no-cloning theorem forbids any data duplication in a quantum system. On the other hand, coding and decoding schemes might be drastically sped up by the use of quantum resources such as entanglement.

It is possible to develop a fault-tolerant strategy for quantum systems based on the recursive encoding of states by concatenation of quantum error-correction codes (see Section V-D, [2], and [54]). The main result we build upon is the following: *A quantum circuit containing N error-free gates can be simulated with a probability of failure of at most ε using $O(\mathrm{poly}(\log(N/\varepsilon))N)$ imperfect gates which fail with probability p as long as $p < p_{\mathrm{th}}$, where p_{th} is a constant threshold that is independent of N.* This remarkable result, the *Threshold Theorem* [45], is achieved by three steps: 1) using quantum error-correction codes (Section V-B); 2) performing all computations on encoded data, using *fault tolerant procedures* (Section V-C); and 3) recursively encoding until the desired reliability is obtained (Section V-D). All of these results are from prior literature [2], [45], [54]–[56], but we describe them here to make our contributions clearer in later sections.

B. *Quantum Error Correction*

The only errors which can occur to a classical bit are bit-flips and erasures, which can be modeled as conditional and random NOT gates. Quantum bits suffer more kinds of error, because of the greater degree of freedom in their state representation; surprisingly, however, there are general strategies for reducing the universe of possible quantum errors to only two kinds: bit-flips (random X gates) and phase-flips (random Z gates). Classical error-correction codes only take into account bit flip errors and, thus, are insufficient for correcting quantum data. Furthermore, quantum states collapse upon measurement, so strategies must be employed for determining errors without actually measuring encoded data.

Classical error correction relies upon distributing k bits of information across n bits $n > k$ and ensuring enough redundancy to recreate the original information. Because of the no-cloning theorem, quantum information cannot be simply duplicated. Instead, redundancy is achieved through entangled states with known properties. For example, a single logical qubit, $c_0|0_L\rangle + c_1|1_L\rangle$ can be represented using three physical qubits, as the state $c_0|000\rangle + c_1|111\rangle$. A bit flip error on the first (left-most) qubit would turn this into $c_0|100\rangle + c_1|011\rangle$; this error can be detected by computing the *parity* of each pair of qubits, and leaving the result in an extra qubit called an *ancilla*. The three parities give the *error syndrome*, uniquely locating any single bit-flip error. Crucially, this strategy reveals nothing about the coefficients c_0 and c_1, since the parities cannot distinguish between $|000\rangle$ and $|111\rangle$ or any single bit-flip

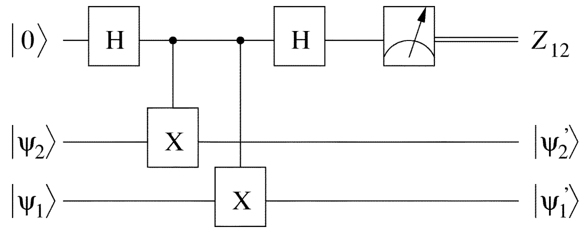


Fig. 11. Measuring Z_{12}, the phase difference between ψ_2 and ψ_1.

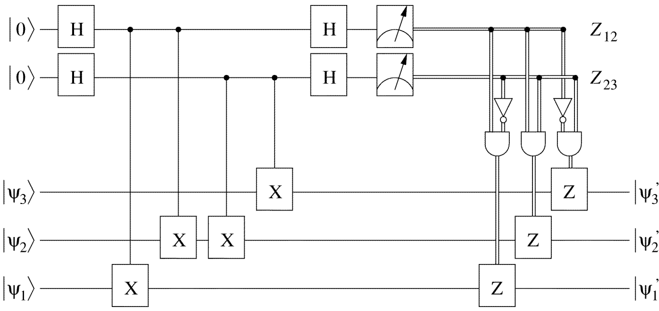


Fig. 12. Syndrome measurement for a 3-qubit code. The meter boxes indicate measurement, and the double lines indicate classical communication controlling the application of the Z operator.

TABLE I
PHASE CORRECTION FOR A 3-QUBIT CODE

Z_{12}	Z_{23}	Error Type	Action
0	0	no error	no action
0	1	qubit 3 flipped	flip qubit 3
1	0	qubit 1 flipped	flip qubit 1
1	1	qubit 2 flipped	flip qubit 2

version of the two three-qubit strings. By measuring parities, errors can be detected without collapsing encoded data.

Correcting phase flips is achieved by measuring differences in phase by using a circuit like the one in Fig. 11. This works by using a Hadamard gate to transform phase flips into bit flips. Parities are then measured as before, the results stored in ancilla qubits, and then the qubits are transformed back into their original basis. Fig. 12 shows how a phase error syndrome can be computed and a corresponding correction procedure applied to correct the error, following the specification of Table I.

A quantum code which encodes one qubit and allows any single bit-flip or phase-flip error to be corrected uses the encoding $c_0|0_L\rangle + c_1|1_L\rangle$, where the logical zero and one qubits are

$$|0_L\rangle = \frac{(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)}{2\sqrt{2}}$$
$$|1_L\rangle = \frac{(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle)}{2\sqrt{2}}.$$

This nine qubit code, discovered by Peter Shor [55], is also known as the $[\![9,1,3]\!]$ code, in the notation $[\![n,k,d]\!]$, where n is the number of physical qubits, k is the number of logical qubits encoded, and d is the quantum Hamming distance of the code. A code with distance d is able to correct $(d-1)/2$ errors.

C. Computing on Encoded Data

The nine qubit code has a remarkable property that illustrates a key requirement for fault tolerance: applying a Z gate to each of the nine qubits takes $|0_L\rangle$ to $|1_L\rangle$ and vice versa. It is the same as applying a logical $\overline{X}$ operator[9] to the encoded qubit! Similarly, $\overline{Z}$ can be performed by applying an X operator to each qubit.

In this paper, we employ Steane's $[\![7,1,3]\!]$ code [57]. The $[\![7,1,3]\!]$ code is the smallest code that allows direct fault-tolerant application of nearly all the operators in the universal set of operators discussed in Section II-B, namely the subset $\{X, Z, H, \text{CNOT}\}$. The T gate can also be performed fault-tolerantly, using a slightly more involved procedure. Thus, universal computation is possible without requiring that the data be decoded.

Merely computing on encoded data is not sufficient, however; one additional step is required, which is frequent error correction. Because all gates used in this task are assumed to be subject to failure, this must be done in a careful manner, such that no single gate failure can lead to more than one error in each encoded qubit block. Such constructions are known as *fault tolerant procedures*, and the impact of this requirement on our study is twofold: 1) no single operation may cause multiple failures and 2) measurement errors must not be allowed to propagate excessively. To achieve 1), no two encoding qubits are allowed to both interact directly with a third qubit. Instead, the "third" qubit is replaced with a *cat state* (a generalization of an EPR pair), $\frac{1}{\sqrt{2}}|00\ldots0\rangle + \frac{1}{\sqrt{2}}|11\ldots1\rangle$, that has itself been verified. Cat states are used because they do not transmit errors through CNOT gates. To achieve 2), measurements are performed in a multiple fashion. While it is not possible to copy a value before measuring, it is possible to form a three-qubit state, similar to the three-qubit bit-flip encoding (Section V-B), where all of the qubits should measure to the same value—if one of the measurements differs, it is assumed to be in error. The implications are explored in detail in later examples.

Any logical operator may be applied as a fault tolerant procedure, as long as the probability, p, of an error for a physical operator is below a certain threshold, $1/c$, where c is determined by the implementation of the error-correction code. For the Steane $[\![7,1,3]\!]$ code, c is about 10^4. The overall probability of error for the logical operator is cp^2. That is, at some step in the application of the operator, and subsequent error correction, two errors would have to occur in order for the logical operator to fail.

D. Recursive Error Correction

A very simple construction allows us to tolerate additional errors. If a logical qubit is encoded in a block of n qubits, it is possible to encode each of those n qubits with an m-qubit code to produce an mn encoding. Such recursion, or *concatenation*, of codes can reduce the overall probability of error even further. For example, concatenating the $[\![7,1,3]\!]$ code with itself gives a $[\![49,1,7]\!]$ code with an overall probability of error of $c(cp^2)^2$ (see Fig. 13). Concatenating it $k-1$ times gives $(cp)^{2^k}/c$, while the size of the circuit increases by d^k and the time complexity increases by t^k, where d is the increase in circuit complexity for a single encoding, and t is the increase in operation time for a

[9] The overscore denotes an operator on a logical qubit: a logical operator.

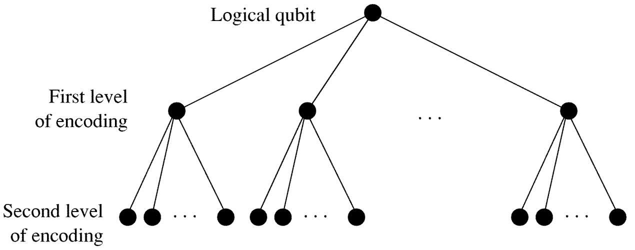


Fig. 13. Tree structure of concatenated codes.

single encoding. For a circuit of size $p(n)$, to achieve a desired probability of success of $1 - \varepsilon$, k must be chosen such that [2]

$$\frac{(cp)^{2^k}}{c} \le \frac{\varepsilon}{p(n)}. \tag{12}$$

The number of operators required to achieve this result is

$$O\left(\text{poly}\left(\frac{\log p(n)}{\varepsilon}\right) p(n)\right). \tag{13}$$

E. Reliability Versus Resources

Given recursive codings, we can, in principle, reduce the probability of error to arbitrarily low levels. Another way to view this is that there is a close relationship between the spatial resources used by a gate and its reliability. The first part of this paper pointed out that (at least in the Kane model), an essential limitation comes from the rise of quantum effects in the wires driving the fields used to control individual qubits and their interactions. As the dimensions of the wires shrink, their current becomes quantized, leading to an imperfect field profile around the controlled qubit. This reduces the fidelity of the quantum gate performed on the qubit. This observation is very interesting, and general. Fundamentally, *classical control circuitry becomes unreliable at small length scales*, but the reliability increases with area used.

This failure behavior can be modeled in the following manner. Assuming we have a quantum circuit consuming an area A on a layout, we may let $p(A)$ be its failure probability. The argument above justifies the assumption that p is a decreasing function of A, and is given generically by a graph similar to Fig. 14. For example, it is likely p decreases exponentially, as $p(A) \sim e^{-\gamma A}$, or for statistical errors, as a complimentary error function, $p(A) \sim \text{erfc}(-\gamma A)$, for some technology-dependent parameter γ.

F. Criteria for the Efficiency of Fault Tolerance

Given our model for failure probability as a function of area, $p(A)$, and the resources required for the fault-tolerant scheme using recursive encoding, from Section V-D, we can now analytically express the tradeoff between the area required to achieve a system of some specified reliability. We consider two approaches. The first is simply to allocate a large area, such that $p(A)$ is as small as desired. The alternative is to apply the fault-tolerant construction using elementary building blocks with a small area A_0, which fails with higher probability $p(A_0)$, requiring an area of d^k.

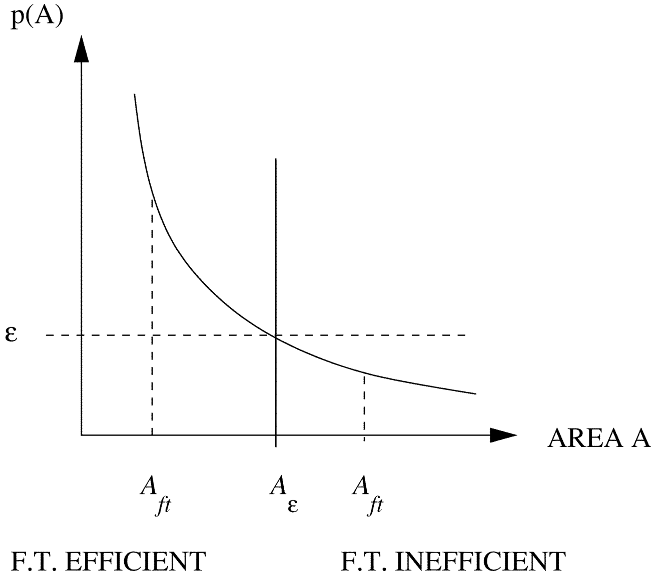


Fig. 14. Relation between circuit reliability and area required, showing the general decreasing trend expected for $p(A)$, and achievable configurations using either of two approaches, requiring area A_ε, or A_{ft}, to achieve $p(A) < \varepsilon$.

Suppose we want to obtain a circuit whose failure probability is bounded by $p \le \varepsilon$. The first approach involves using a large area, $A_\varepsilon = p^{-1}(\varepsilon)$. The second approach utilizes a recursive, fault tolerant construction, which makes sense if the component area A_0 is chosen such that $\varepsilon \le p(A_0) \le p_{th} = 1/c$, that is, the component failure probability is smaller than the failure probability threshold tolerated by the error-correction code. The overall area required is then $A_{ft} = A_0 d^k$, where the recursion level k is determined by the solution to (12),

$$k = \left\lceil \log \frac{\log\left(\frac{p_{\text{th}}}{\varepsilon}\right)}{\log\left(\frac{p_{\text{th}}}{p(A_0)}\right)} \right\rceil. \tag{14}$$

The fault tolerant construction will be more efficient if and only if $A_{\text{ft}} \le A_\varepsilon$ or, equivalently, if there exists an A_0 such that

$$p\left(A_0 d^{\left\lceil \log \frac{\log\left(\frac{1}{\varepsilon c}\right)}{\log\left(\frac{1}{p(A_0)c}\right)} \right\rceil}\right) \ge \varepsilon \tag{15}$$

$$\varepsilon \le p(A_0) \le \frac{1}{c}. \tag{16}$$

This is an interesting, and nonlinear inequality; solutions may be visualized using Fig. 14. The existence of an area efficient fault-tolerant implementation depends on the structure of the function $p(A)$. If p decreases slowly enough with the area (as inverse power of the area for instance), then such an implementation exists.

Fault tolerance through recursive encoding can drastically improve the reliability of quantum circuits, and perhaps even in an error efficient manner. (16) gives a method for determining the appropriate redundancy in the design of a quantum circuit from the standpoint of area efficiency. The possibilities offered by this strategy are far from being entirely explored. Some solutions to this equation are presented elsewhere [58], but other approaches, using different fault-tolerant schemes (such as nonrecursive constructions), or quantum resource assisted fault tolerance [59], may lead to modifications of this bound. In general, however, the concept expressed by (16) will remain: fine-grained fault tolerant circuit constructions can provide valuable means for resource efficiency tradeoffs in future quantum architectures.

G. Layout of Error-Correction Circuits

While our high-level analysis shows that recursive error correction has desirable efficiency properties, we shall see that the details of implementing such schemes will reveal some key issues. The most important of these issues is the need for reliable, long-distance communication.

Given the pitch-matching constraints of linearity with infrequent junctions from IV-B-1, there are still several ways to lay out physical and logical qubits. Optimally, qubits should be arranged to minimize communication overhead.

In a fault tolerant design, the main activity of a quantum computer is error correction. To minimize communication costs, qubits in an encoding block should be in close proximity. Assuming that the distance between junctions is greater than the number of qubits in the block, the closest the qubits can be is in a straight line.

A concatenated code requires a slightly different layout. Error correction is still the important operation, but the logical qubits at all but the bottom level of the code are more complicated. For the second level, the qubits are themselves simple encodings, and so can be laid out linearly. However, we want these qubits in as close proximity to each other as possible, for the same reasons we wanted the qubits in the simple code close. Hence, we need to arrange the bottom level as branches coming off of a main bus. Similarly, the third level would have second-level branches coming off of a main trunk, and so on for higher levels.

In the next two sections, we describe a basic error-correction algorithm and its recursive application, focusing on illustrating realistic space and time costs such as those described above, imposed by 2-D implementation technologies.

VI. Error-Correction Algorithms

A. $[\![7,1,3]\!]$ Code

Error correcting using the $[\![7,1,3]\!]$ code consists of measuring the error syndrome parities of the encoding qubits in various bases, and correcting the codeword based on the measured syndrome. As shown in Fig. 15, the qubits are rotated to the different measurement bases with Hadamard gates. Parity is then measured in much the same way as with a classical code, using two-qubit CNOT operators acting as XORs. Conceptually, the parity can be measured in the same way as the three-qubit code in Section V-B, gathering the parity on ancilla $|0\rangle$s. To perform a fault-tolerant measurement, however, a cat state is used in place of a $|0\rangle$. Fig. 15 shows all six parity measurements using cat states. Not shown are cat-state creation and cat-state verification.

A parity measurement consists of the following steps.

1) Prepare a cat state from four ancillae, using a Hadamard gate and three CNOT gates.

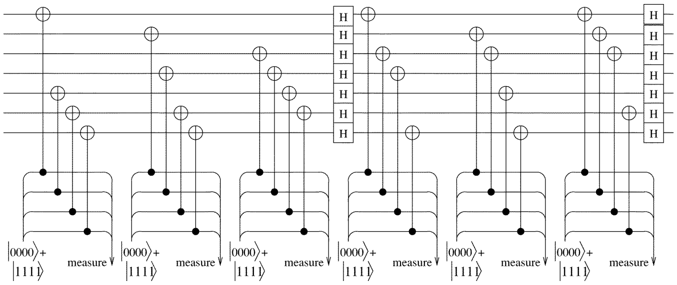


Fig. 15. Measuring the error syndrome for the $[\![7,1,3]\!]$ error-correction code.

2) Verify the cat state by taking the parity of each pair of qubits. If any pair has odd parity, return to step 1. This requires six additional ancillae, one for each pair.
3) Perform a CNOT between each of the qubits in the cat state and the data qubits whose parity is to be measured (See Fig. 15).
4) Uncreate the cat state by applying the same operators used to create it in reverse order. After applying the Hadamard gate to the final qubit, $|A_0\rangle$, that qubit contains the parity.
5) Measure $|A_0\rangle$:
   A With $|A_0\rangle = \alpha|0\rangle + \beta|1\rangle$, create the three-qubit state, $\alpha|000\rangle + \beta|111\rangle$ by using $|A_0\rangle$ as the control for two CNOT gates, and two fresh $|0\rangle$ ancillae as the targets.
   B Measure each of the three qubits.
6) Use the majority measured value as the parity of the cat state.

Each parity measurement has a small probability of introducing an error, either in the measurement, or in the data qubits. Hence, the entire syndrome measurement must be repeated until two measurements agree. The resulting syndrome determines which, if any, qubit has an error, and which X, Z, or Y operator should be applied to correct the error. After correction, the probability of an error in the encoded data is $O(p^2)$.

For the Steane $[\![7,1,3]\!]$ code, each parity measurement requires twelve ancillae—four for the cat state to capture the parity, six to verify the cat state, and two additional qubits to measure the cat state. The six parity measurements are each performed at least twice, for a minimum of 144 ancillae to measure the error syndrome!

The minimum number of operations required for an error correction is 38 Hadamards, 288 CNOTs, and 108 measurements. With parallelization, the time required for the operations is $24S + 156C + M$, where S is the time required for a single qubit operator, C is the time required for a CNOT, and M is the time required for a measurement. (We assume all but the last measurement are performed in parallel with other operations.)

B. Concatenated Codes

The $[\![7,1,3]\!] \times [\![7,1,3]\!]$ two-level concatenated code is measured in the same way as the $[\![7,1,3]\!]$ code, except the qubits are encoded, and each parity measurement uses a 12-qubit cat state.[10]

[10] In the $[\![7,1,3]\!]$ code, an $\overline{X}$ consists of an X on each qubit. The parity of the logical qubit is the same as that of the physical qubits. Since a logical qubit is a valid codeword, a four-qubit subset of the qubits has even parity, and the remaining three qubits has the same parity as the logical qubit.

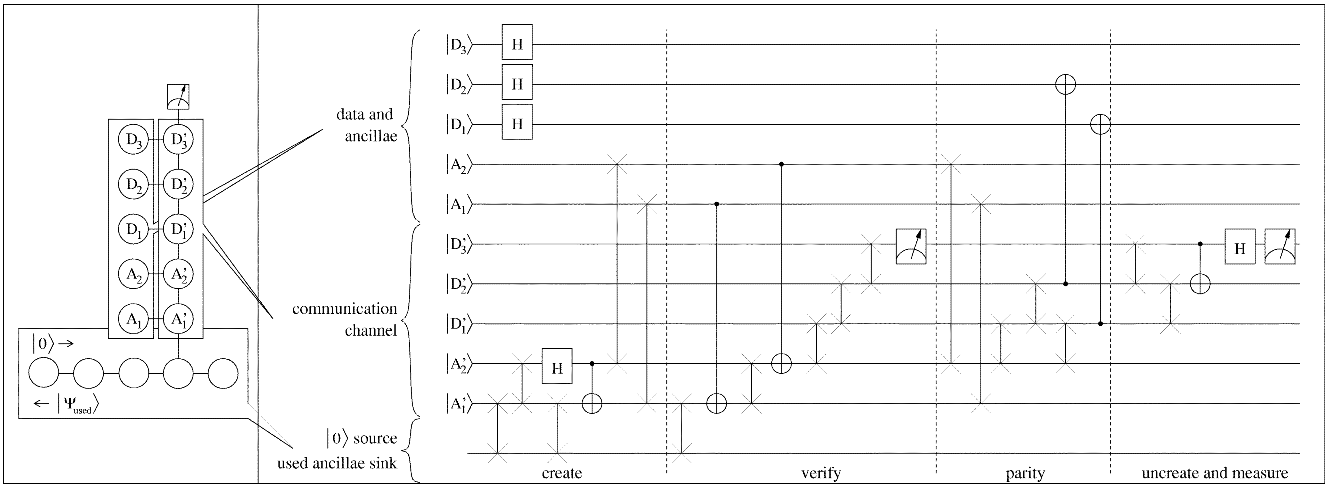


Fig. 16. "Two-rail" layout for the three-qubit phase-correction code. The schematic on the left shows qubit placement and communication, where D_is indicate data qubits, and A_is are cat-state ancillae. The column of D_i's and A_i's form a swapping channel and can also interact with the data and cat-state ancilla. The open qubit-swapping channel at the bottom brings in fresh ancillae, and removes used ancillae. The same layout is shown as a quantum circuit on the right, with the operations required to create and verify an ancillary cat state, and to measure the parity of a pair of data qubits.

The error syndrome measurement is analogous to the singly-encoded $[\![7,1,3]\!]$ case, except that the lower-level encodings must be error corrected between the following operations.

1) Prepare 12 ancillae in a cat state.
2) Verify the cat state (66 ancillae for pairwise verification.)
3) Perform CNOTs between the cat state qubits and the qubits encoding the data qubits whose parity is to be measured.
4) Error correct the four logical data qubits.
5) Uncreate the cat state, and measure the resulting qubit.

As in the singly-encoded case, each syndrome measurement must be repeated, in this case at least four times. The resulting syndrome determines which, if any, logical qubit has an error. The appropriate $\overline{X}$, $\overline{Z}$, or $\overline{Y}$ operator can be applied to correct the error. After the correction operator is applied to a logical qubit, that qubit must be error-corrected. The probability of an error in the encoded data is $O(p^4)$ after correction.

Each parity measurement requires 154 Hadamards, 1307 CNOTs, and 174 measurements, in time $26S + 201C + M$, using the same assumptions as for the nonconcatenated case.

Of course, the $[\![7,1,3]\!]$ code can be concatenated more than once. The error-correction procedure for higher levels of concatenation is similar to the above. The key is that probability of error for each parity measurement must be $O(p^{2^k})$, for a code concatenated $k-1$ times.

VII. Communication Costs and Error Correction

In this section, we model the communication costs of the error-correction algorithms of Section VI, under the constraint of having only near neighbor interactions. While it has previously been proven that under such constraints, the Threshold Theorem can still be made to apply (given suitably reduced failure probability thresholds) [60], a detailed study was not performed with layout constraints on quantum error-correction circuits. We first study the growth rate of errors when using SWAP operations. Second, we analyze quantum teleportation as an alternative to SWAP operations for long-distance communication. Finally, we show that teleportation is preferable both in terms of distance and in terms of the accumulating probability of correlated errors between redundant qubits in our codewords.

A. Error-Correction Costs

The error-correction algorithms in the previous section are presented for the ideal situation, where any qubit can interact with any other qubit. Usually, qubits can only interact with their near neighbors, so before applying a two-qubit operator, one of the operand qubits must be moved adjacent to the other.

One of the easiest ways to move quantum data is to use the SWAP operator. By applying SWAPs between alternating pairs of qubits, the values of alternating qubits are propagated in one direction, while the remaining qubit values are propagated in the reverse direction. This swapping channel can be used to supply $|0\rangle$ ancillae for the purpose of error correction, remove "used" ancillae, and allow for qubit movement. Fig. 16 illustrates this for the three-qubit example, using two columns of qubits, one for the data and cat-state qubits, and one for communication.

The same layout can be applied to the $[\![7,1,3]\!]$ code, giving a minimum time for an error-correction parity check of

$$t_{ecc} = 12(t_{cc} + t_{cv} + t_p + t_{cd} + t_m) \qquad (17)$$

where

t_{cc} time for cat-state creation;
t_{cv} time for cat-state verification;
t_p time to entangle the cat state with the parity qubits;
t_{cu} time to uncreate the cat state; and
t_m time to perform a triply-redundant measurement.

For $[\![7,1,3]\!]$ in the ideal, parallel, "sea-of-qubits" model, $t_{cc} = t_{\text{single}} + 3t_{\text{cnot}}$, $t_{cv} = 6t_{\text{cnot}} + t_{\text{meas}}$, $t_p = t_{\text{cnot}}$, and $t_{cu} = 3t_{\text{cnot}} + t_{\text{single}}$, where

t_{single} time required for a single-qubit operator;
t_{cnot} time required for a CNOT operator;
t_{swap} time required for a SWAP operator;
t_{meas} time required for redundant measurement.

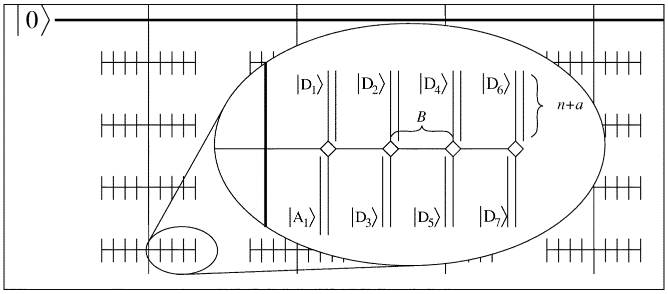


Fig. 17. Schematic layout of the H-tree structure of a concatenated code. The branches labeled $|D_i\rangle$ are for logical data qubits, and consist of two rails of eleven qubits each—seven qubits for data and four for ancillae. The branch labeled $|A_1\rangle$ is for creating, verifying, and uncreating the cat state.

If communication by swapping is used

$$t_{cc} = \max(t_{\text{single}}, t_{\text{swap}}) + 6t_{\text{swap}} + 3\max(t_{\text{cnot}}, t_{\text{swap}}) \tag{18}$$

$$t_{cv} = \max(t_{\text{single}}, t_{\text{swap}}) + 9t_{\text{swap}} + 11\max(t_{\text{cnot}}, t_{\text{swap}}) \tag{19}$$

$$t_p \leq 7t_{\text{swap}} + 4\max(t_{\text{cnot}}, t_{\text{swap}}) \tag{20}$$

$$t_{cd} = t_{\text{swap}} + 3t_{\text{cnot}} + t_{\text{single}} + t_{\text{meas}}. \tag{21}$$

In the Kane model, $t_{\text{single}} < t_{\text{swap}} < t_{\text{cnot}} < t_{\text{meas}}$. Including parallelism between parity measurements, the minimum time for a syndrome measurement is

$$t_{ecc} = 221t_{\text{swap}} + 210t_{\text{cnot}} + t_{\text{single}} + t_{\text{meas}}.$$

Since measurement is fully parallelizable, these times assume that there are enough measurement units to perform measurement in parallel with the other operations in the error-correction cycle.

B. Multilevel Error Correction

For the singly concatenated code, the data movement in the upper level is more complicated, although (17) still holds. The first step in the error correction is creating and verifying the 12-qubit cat state. Fig. 17 shows how the ancillae "branches" are incorporated into the data branches. After verification, the cat state is moved to the appropriate data branches, where it is CNOTed with the data qubits. The cat state is then moved back and uncreated, while the data branches are error-corrected. Finally, a Hadamard is applied to the last cat-state ancilla, which is then redundantly measured. The layout in Fig. 17 is not necessarily optimal.

For $[\![7,1,3]\!]$ concatenated with itself k times

$$t_{cc,k} \approx \lceil \log_2(a_k) \rceil\, t_{\text{cnot}} + \left(\frac{5}{2}a_k - 3\right) t_{\text{swap}} \tag{22}$$

$$t_{cv,k} = 2a_k t_{\text{cnot}} + (a_k(a_k-2)+2)\, t_{\text{swap}} \tag{23}$$

$$t_{p,k} = a_k + 3t_{b,k} + 3t_{b,k-1} + t_{ecc,k-1} \tag{24}$$

$$t_{cu,k} = t_{cc,k} + t_{\text{single}} + t_m \tag{25}$$

$$a_k = 4 \times 3^{k-1} \tag{26}$$

$$t_{b,k} = \begin{cases} 1, & k=1 \\ B, & k=2 \\ t_{b,k-1} + (n+a_1)t_{b,k-2}, & k=3 \\ t_{b,k-1} + 2\left\lceil \frac{n}{2} \right\rceil t_{b,k-2}, & k>3 \end{cases} \tag{27}$$

where the subscript k indicates the level of encoding, a_k is the number of qubits in the cat state at level k, $t_{b,k}$ is the branch distance between logical qubits at level k, B is the minimum number of qubits between two branches for a given architectural model, and n is the number of physical qubits in the nonconcatenated code.

With communication by swapping channel, the SWAP operator becomes very important. In the sea-of-qubits model, SWAPs are not required. In the model described above, SWAPs account for over 80% of all operations.

C. Avoiding Correlated Errors

An important assumption in quantum error correction is that errors in the redundant qubits of a codeword are uncorrelated. That is, we do not want one error in a codeword to make a second error more likely. To avoid such correlation, it is important to try not to interact qubits in a codeword with each other.

Unfortunately, we find that a 2-D layout cannot avoid indirect interaction of qubits in a codeword. At some point, all the qubits in a codeword must be brought to the same physical location in order to calculate error syndromes. In order to do this, they must pass through the same line of physical locations. Although we can avoid swapping the codeword qubits with each other, we cannot avoid swapping them with some of the same qubits that flow in the other direction.

For concreteness, if two qubits of codeword d_0 and d_1 both swap with an ancilla a_0 going in the opposite direction, there is some probability that d_0 and d_1 will become correlated with each other through the ancilla. This occurs if both SWAPs experience a partial failure. In general, if p is the probability of a failure of a SWAP gate, the probability of an error from swapping a logical qubit is

$$n^k b_k p + \binom{n^k}{2} b_k p^2 + \binom{nk}{3} b_k p^3 + \cdots$$

where b_k is the number of qubits between branches at level k, and the higher order terms are due to correlation between the qubits. From this form, it is clear that correlated errors are dominated by uncorrelated errors, when $n^k p \ll 1$.

By calculating the number of basic computation and communication operations necessary to use teleportation for long-distance communication, we can quantify when we should switch from swapping to teleportation in our tree structure. Fig. 18 illustrates this tradeoff. We can see that for $B = 22$, teleportation should be used when $k \geq 5$.

D. Teleportation

Table II lists the number of SWAP operations required to move an unencoded qubit from one level-k code word to the adjacent code word for different minimum branch distances, as well as the total operations to teleport the same qubit. Since a teleportation channel precommunicates EPR pairs, it has a fixed cost. To use teleportation for our circuit, we must evaluate the number of computation and communication operations within the teleportation circuit. By comparing this number of operations with the swapping costs from the previous section, we can decide at what level k of the tree to start using teleportation instead of swapping for communication.

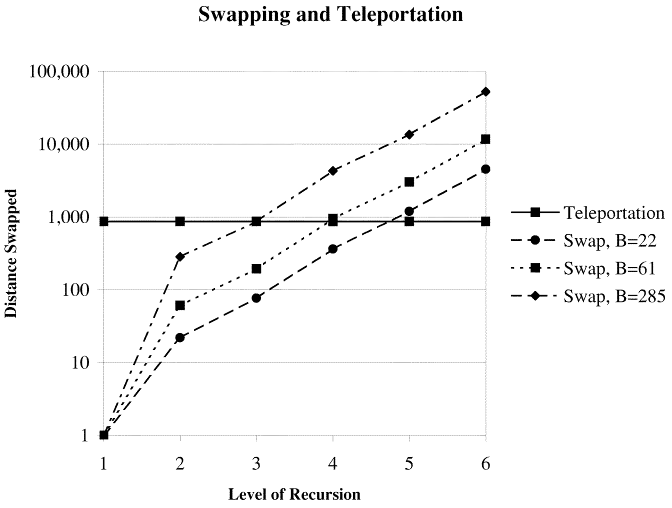


Fig. 18. Cost of teleportation compared to swapping. The B-values chosen illustrate break-even points for different levels of recursion.

TABLE II
COMPARISON OF THE COST OF SWAPPING AN ENCODED QUBIT TO THE COST OF TELEPORTING IT. THE B-VALUES ARE THE DISTANCE BETWEEN ADJACENT QUBITS

k	Teleportation	Swapping, $B=22$	Swapping, $B=61$	Swapping, $B=285$
1	864	1	1	1
2	864	22	61	285
3	864	77	194	866
4	864	363	948	4,308
5	864	1,199	3,032	13,560
6	864	4,543	11,680	52,672

Teleportation has another advantage, which is beyond the scope of this study. By suitably modifying the EPR pairs, teleportation can be used to perform operations at a distance [59]. It does not eliminate the need for error correction, and correctly modifying the EPR pairs has its own costs. This is an interesting area for future research.

VIII. SYSTEM BANDWIDTH

Our goal has been to design a reliable, scalable quantum communication layer that will support higher-level quantum error correction and algorithms functioning on top of this layer. A key issue for future evaluation, however, is that the lower latency of our teleportation channel actually translates to an even higher bandwidth when the upper layers of a quantum computation are considered. It is for this reason that long wires should not be constructed from chained swapping-channels and quantum "repeaters."

The intuition behind this phenomenon is as follows. Quantum computations are less reliable than any computation technology that we are accustomed to. In fact, quantum error correction consumes an enormous amount of overhead both in terms of redundant qubits and time spent correcting errors. This overhead is so large that the reliability of a computation must be tailored specifically to the run length of an algorithm. The key is that, the longer a computation runs, the stronger the error correction needed to allow the data to survive to the end of the computation. The stronger the error correction, the more bandwidth consumed transporting redundant qubits. Thus, lower latency on each quantum wire translates directly into greater effective bandwidth of logical quantum bits.

IX. CONCLUSION

Quantum computation is in its infancy, but now is the time to evaluate quantum algorithms under realistic constraints and derive the architectural mechanisms and reliability targets that we will need to scale quantum computation to its full potential. Our work has focused upon the spatial and temporal constraints of solid-state technologies.

Building upon key pieces of quantum technology, we have provided an end-to-end look at a quantum wire architecture. We have exploited quantum teleportation to enable pipelining and flexible error correction. We have shown that our teleportation channel scales with distance and that swapping channels do not. Finally, we have discovered fundamental architectural pressures not previously considered. These pressures arise from the need to colocate physical phenomena at both the quantum and classical scale. Our analysis indicates that these pressures will force architectures to be sparsely connected, resulting in coarser-grain computational components than generally assumed by previous quantum computing studies.

At the systems level, the behavior of wires becomes a crucial limiting factor in the ability to construct a reliable quantum computer from faulty parts. While the Threshold Theorem allows fault-tolerant quantum computers to be realized in principle, we showed that in practice many assumptions must be carefully scrutinized, particularly for implementation technologies that force a 2-D layout scheme for qubits and their interconnects. Our analysis suggests that, rather counterintuitively, fault-tolerant constructions can be more resource efficient than equivalent circuits made from more reliable components, when the failure probability is a function of resources required. And a detailed study of the resources required to implement recursive quantum error-correction circuits highlights the crucial role of qubit communication, and in particular, the dominant role of SWAP gates. We find that at a certain level of recursion, resources are minimized by choosing a teleportation channel instead of the SWAP. It is likely that the reliability of the quantum SWAP operator used in short-distance communication will be the dominant factor in future quantum architecture system reliability.

ACKNOWLEDGMENT

Thanks to John Owens, Matt Farrens, and Mark Whitney for their helpful comments on preliminary material for this paper.

REFERENCES

[1] S. Lloyd, "Quantum-mechanical computers," *Sci. Amer.*, vol. 273, p. 44, Oct. 1995.

[2] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[3] D. P. DiVincenzo, "Quantum computation," *Science*, vol. 270, no. 5234, p. 255, 1995.

[4] N. Gershenfeld and I. Chuang, "Quantum computing with molecules," *Sci. Amer.*, June 1998.

[5] Y. Maklin, Y. Schön, and A. Schnirman, "Quantum state engineering with Josephson-junction devices," *Rev. Modern Phys.*, vol. 73.

[6] C. H. Bennett and D. P. DiVincenzo, "Quantum information and computation," *Nature*, vol. 404, pp. 247–255, 2000.

[7] P. S. M. I. Dykman and P. M. Platzman, "Qubits with electrons on liquid helium," *Phys. Rev. B*, vol. 67.

[8] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, R. Cleve, and I. L. Chuang, "Experimental realization of order-finding with a quantum computer," *Phys. Rev. Lett.*, vol. 85, no. 15, pp. 5453–5455.

[9] E. Knill, R. Laflamme, R. Martinez, and C. Tseng, "An algorithmic benchmark for quantum information processing," *Nature*, vol. 404, pp. 368–370, 2000.

[10] C. Sackett, D. Kielpinsky, B. King, C. Langer, V. Meyer, C. Myatt, M. Rowe, Q. Turchette, W. Itano, D. Wineland, and C. Monroe, "Experimental entanglement of four particles," *Nature*, vol. 404, pp. 256–258, 2000.

[11] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland, "Quantum state engineering with Josephson-junction devices," *Rev. Modern Phys.*, vol. 75, pp. 281–324, 2003.

[12] B. Kane, "A silicon-based nuclear spin quantum computer," *Nature*, vol. 393, pp. 133–137, 1998.

[13] D. Vion, A. Aassime, A. Cottet, P. Joyez, H. Pothier, C. Urbina, D. Esteve, and M. H. Devoret, "Manipulating the quantum state of an electrical circuit," *Science*, vol. 296, p. 886, 2002.

[14] D. Kielpinsky, C. Monroe, and D. Wineland, "Architecture for a large-scale ion trap quantum computer," *Nature*, vol. 417, p. 709, 2002.

[15] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. Wootters, "Teleporting an unknown quantum state via dual classical and EPR channels," *Phys. Rev. Lett.*, vol. 70, pp. 1895–1899, 1993.

[16] L. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, 1996, pp. 212–219.

[17] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Foundations Comput. Sci.*, Los Alamitos, CA, 1994, p. 124.

[18] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comp.*, vol. 26, no. 5, pp. 1484–1509, 1997.

[19] A. Childs, E. Farhi, and J. Preskill, "Robustness of adiabatic quantum computation," *Phys. Rev. A*, vol. 65, 2002.

[20] R. Jozsa, D. Abrams, J. Dowling, and C. Williams, "Quantum atomic clock synchronization based on shared prior entanglement," *Phys. Rev. Lett.*, pp. 2010–2013, 2000.

[21] I. L. Chuang, "Quantum algorithm for clock synchronization," *Phys. Rev. Lett.*, vol. 85, p. 2006, 2000.

[22] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *Proc. IEEE Int. Conf. Comput., Syst., Signal Process.*, 1984, pp. 175–179.

[23] W. van Dam and G. Seroussi, Efficient Quantum Algorithms for Estimating Gauss Sums.

[24] S. Hallgren, "Polynomial time quantum algorithms or Pell's equation and the principal ideal problem," in *Proc. Symp. Theory Comput.*, May 2002, pp. 653–658.

[25] J. S. Bell, "On the Einstein-Podolsky-Rosen paradox," *Physics*, vol. 1, pp. 195–200, 1964.

[26] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland, "Demonstration of a fundamental quantum logic gate," *Phys. Rev. Lett.*, vol. 75, p. 4714, 1995.

[27] Q. A. Turchette, C. J. Hood, W. Lange, H. Mabuchi, and H. J. Kimble, "Measurement of conditional phase shifts for quantum logic," *Phys. Rev. Lett.*, vol. 75, p. 4710, 1995.

[28] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, R. Cleve, and I. L. Chuang, "Experimental realization of order-finding with a quantum computer," *Phys. Rev. Lett.*, 2000.

[29] Y. Yu, S. Han, X. Chu, S.-I. Chu, and Z. Wang, "Coherent temporal oscillations of macroscopic quantum states in a Josephson junction," *Science*, pp. 889–892, May 2002.

[30] M. W. Coffey, "Quantum computing based on a superconducting quantum interference device: Exploiting the flux basis," *J. Modern Opt.*, vol. 49, no. 14, pp. 2389–2398, 2002.

[31] R. Vrijen, E. Yablonovitch, K. Wang, H. W. Jiang, A. Balandin, V. Roychowdhury, T. Mor, and D. DiVincenzo, Electron Spin Resonance Transistors for Quantum Computing in Silicon-Germanium Heterostructures, 1999.

[32] A. Skinner, M. Davenport, and B. Kane, "Hydrogenic spin quantum computing in silicon: A digital approach," *Phys. Rev. Lett.*, vol. 90, February 2003.

[33] M. Oskin, F. Chong, I. Chuang, and J. Kubiatowicz, "Building quantum wires: the long and the short of it," in *Proc. Int. Symp. Comput. Architec. (ISCA 2003)*, 2003.

[34] A. Skinner *et al.*, Hydrogenic Spin Quantum Computing in Silicon: A Digital Approach, 2002.

[35] B. Kane and N. McAlpine *et al.*, "Single-spin measurement using single-electron transistors to probe two-electron systems," *Phys. Rev. B*, vol. 61, pp. 2961–2972, 2000.

[36] A. Globus, D. Bailey, J. Han, R. Jaffe, C. Levit, R. Merkle, and D. Srivastava, "NASA applications of molecular nanotechnology," *J. Brit. Interplanetary Soc.*, vol. 51, 1998.

[37] L. Adleman, "Toward a Mathematical Theory of Self-Assembly," Univ. Southern California Tech. Rep., 2000.

[38] E. Anderson, V. Boegli, M. Schattenburg, D. Kern, and H. Smith, "Metrology of electron beam lithography systems using holographically produced reference samples," *J. Vac. Sci. Technol.*, vol. B-9, p. 3606, 1991.

[39] M. Sanie, M. Cote, P. Hurat, and V. Malhotra, "Practical application of full-feature alternating phase-shifting technology for a phase-aware standard-cell design flow," in *Proc. Design Automation Conf. (DAC)*, 2001, pp. 93–96.

[40] D. K. Ferry and S. M. Goodnick, *Transport in Nanostructures*. Cambridge, U.K.: Cambridge Univ. Press, 1997, vol. 6, Cambridge Studies in Semiconductor Physics & Microelectronic Engineering.

[41] K. K. Likhareve, "Single-electron devices and their applications," *Proc. IEEE*, vol. 87, pp. 602–632, 1999.

[42] J. R. Tucker and T.-C. Shen, "Can single-electron integrated circuits and quantum computers be fabricated in silicon?," *Int. J. Circuit Theory Applicat.*, vol. 28, pp. 553–562, 2000.

[43] M. Oskin, F. Chong, and I. Chuang, "Overhead reduction in a architecture for quantum computers," *IEEE Comput.*, vol. 35, pp. 79–87, 2002.

[44] B. E. Kane, N. S. McAlpine, A. S. Dzurak, R. G. Clark, G. J. Milburn, H. B. Sun, and H. Wiseman, "Single spin measurement using single electron transistors to probe two electron systems," Phys. Rev. B., 1999, submitted for publication.

[45] D. Aharonov and M. Ben-Or, "Fault tolerant computation with constant error," in *Proc. 29th Annu. ACM Symp. Theory Comput.*, 1997, pp. 176–188.

[46] E. Knill, R. Laflamme, and W. H. Zurek, "Resilient quantum computation," *Science*, vol. 279, no. 5349, pp. 342–345, 1998.

[47] J. Preskill, "Fault-tolerant quantum computation," in *Quantum Information and Computation*, H.-K. Lo, T. Spiller, and S. Popescu, Eds. Singapore: World Scientific, 1998.

[48] A. Verhulsta *et al.*, "Non-thermal nuclear magnetic resonance quantum computing using hyperpolarized xenon," *Appl. Phys. Lett.*, 2001.

[49] L. J. Schulman and U. Vazirani, "Scalable NMR quantum computation," in *Proc. 31st Annu. ACM Symp. Theory Comput. (STOC)*.

[50] ——, "Molecular scale heat engines and scalable quantum computation," in *Proc. 31st Annu. ACM Symp. Theory Comput. (STOC)*, 1999, pp. 322–329.

[51] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed state entanglement and quantum error correction," *Phys. Rev. A*, vol. 54, p. 3824, 1996.

[52] C. H. Bennett, H. J. Bernstein, S. Popescu, and B. Schumacher, "Concentrating partial entanglement by local operations," *Phys. Rev. A*, vol. 53, no. 4, pp. 2046–2052, 1996.

[53] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, "Purification of noisy entanglement and faithful teleportation via noisy channels," *Phys. Rev. Lett.*, vol. 76, p. 722, 1996.

[54] D. Gottesman, "Theory of fault-tolerant quantum computation," *Phys. Rev. A*, vol. 57, no. 1, pp. 127–137, 1998.

[55] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 54, p. 2493, 1995.

[56] A. Steane, "Simple quantum error correcting codes," *Phys. Rev. Lett.*, vol. 77, pp. 793–797, 1996.

[57] ——, "Error correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, pp. 793–797.

[58] F. Impens, "Resource Optimization Using Fine-Grained Fault Tolerance Constructions," Master's thesis, Massachusetts Inst. Technol., Cambridge, 2004.

[59] D. Gottesman and I. L. Chuang, "Quantum teleportation is a universal computational primitive," *Nature*, vol. 402, pp. 390–392, 1999.

[60] D. Gottesman, Fault Tolerant Quantum Computation With Local Gates, 1999.

[61] D. Copsey *et al.*, "The effect of communication costs in solid-state quantum computing architectures," in *Proc. Symposium on Parallelism in Algorithms and Architectures (SPAA 2003)*. New York: ACM Press, 2003.

[62] J. S. Bell, *Speakable and Unspeakable in Quantum Mechanics*. Cambridge, U.K.: Cambridge Univ. Press, 1987.

Dean Copsey received the B.Sc. degree in chemical engineering and the M.S. degree in 1986 and 1996, respectively, in computer science from the University of California at Davis, where he is a Ph.D. student in computer science.

His current research interests include quantum-computing architectures, fault tolerance in nanoscale transistor technologies, and tile architectures for digital signal processing.

Mark Oskin received the Ph.D. degree from the University of California, Davis, in 2001.

He is an Assistant Professor at the University of Washington in computer science and engineering. His current research interests include quantum architectures and languages, and scalable execution substrates for future silicon technologies.

Francois Impens is a graduate student at the Massachusetts Institute of Technology, Cambridge.

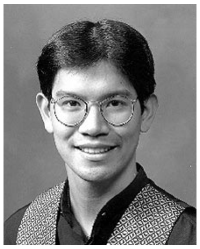
His current research interests focus on the design of new fault tolerant schemes for classical and quantum circuits.

Tzvetan Metodiev received the B.S. degree in physics from the University of California at Davis, where he is a graduate student.

He joined the Computer Science Department, University of California, in September 2002. His current research interests are architectures and algorithms for quantum computation.

Andrew Cross (S'99) received the B.S. degree in electrical engineering from Case Western Reserve University, Cleveland, OH. He is a graduate student at the Massachusetts Institute of Technology.

His current research interests include quantum-computing architectures and composite pulse techniques for quantum control.



Frederic T. Chong received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1990, 1992, and 1996, respectively.

He is an Associate Professor of Computer Science and a Chancellor's Fellow at the University of California at Davis. His current work focuses on architectures and applications for novel computing technologies.

Prof. Chong is a recipient of the National Science Foundation's CAREER award.

Isaac L. Chuang received the B.S. degree in physics and the B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1990, 1991, and 1991, respectively, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1997.

He is an Associate Professor in the Media Laboratory and the Department of Physics, Massachusetts Institute of Technology, Cambridge. He leads the Quanta Research Group at the Center for Bits and Atoms, where his group seeks to understand and create information technology and intelligence from the fundamental building blocks of physical media, atoms and molecules. His other research interests include the physics of information and computation, information theory and cryptography, and silicon biology.

John Kubiatowicz (M'98) received the B.S., M.S, and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1987, 1993, and 1998, respectively.

He is an Associate Professor of Computer Science at the University of California at Berkeley. His research interests include quantum computing, reconfigurable computing, and biological computing. Other interests include security, privacy, and resilience to faults and denial of service attacks in Internet-scale systems.

Prof. Kubiatowicz is a recipient of the National Science Foundation's PECASE Award.

A.7 Nature’03 Paper

Implementation of the Deutsch–Jozsa algorithm on an ion-trap quantum computer

Stephan Gulde*, Mark Riebe*, Gavin P. T. Lancaster*, Christoph Becher*, Jürgen Eschner*, Hartmut Häffner*, Ferdinand Schmidt-Kaler*, Isaac L. Chuang*† & Rainer Blatt*

* *Institut für Experimentalphysik, Universität Innsbruck, Technikerstraße 25, A-6020 Innsbruck, Austria*
† *MIT Media Laboratory, Cambridge, Massachusetts 02139, USA*

Determining classically whether a coin is fair (head on one side, tail on the other) or fake (heads or tails on both sides) requires an examination of each side. However, the analogous quantum procedure (the Deutsch–Jozsa algorithm[1,2]) requires just one examination step. The Deutsch–Jozsa algorithm has been realized experimentally using bulk nuclear magnetic resonance techniques[3,4], employing nuclear spins as quantum bits (qubits). In contrast, the ion trap processor utilises[5] motional and electronic quantum states of individual atoms as qubits, and in principle is easier to scale to many qubits. Experimental advances in the latter area include the realization of a two-qubit quantum gate[6], the entanglement of four ions[7], quantum state engineering[8] and entanglement-enhanced phase estimation[9]. Here we exploit techniques[10,11] developed for nuclear magnetic resonance to implement the Deutsch–Jozsa algorithm on an ion-trap quantum processor, using as qubits the electronic and motional states of a single calcium ion. Our ion-based implementation of a full quantum algorithm serves to demonstrate experimental procedures with the quality and precision required for complex computations, confirming the potential of trapped ions for quantum computation.

Laser-cooled trapped ions are ideally suited to the investigation and implementation of quantum information processing[12] because they exhibit these properties: (1) localization of the single particle to less than a few tens of nanometres[13–15]; (2) control of the motional state down to the zero point of the trapping potential[8,16]; (3) a high degree of isolation from the environment and thus a very long time available for manipulations of their quantum state[17]; and (4) the ability to detect the ion's quantum state with high precision by the electron shelving technique[18]. The same properties make single trapped ions well suited for storing quantum information in long-lived internal states[19].

In our experiment we implement the Deutsch–Jozsa algorithm on a quantum processor based on a single trapped $^{40}\mathrm{Ca}^+$ ion which is driven by laser pulses. A compensation technique for frequency shifts allows us to achieve the required control over the optical phases of the pulses[20]. Following a recent proposal[10], we also successfully combine ion-trap techniques for quantum state manipulation with the method of composite pulses[11] adopted from NMR technology. Thus we achieve complete control over the ion's motional and electronic state. The implementation of a quantum algorithm on an ion-trap processor, which we demonstrate here, serves as a test of the suitability of these techniques, particularly in view of their scalability towards a larger number of qubits.

To illustrate the Deutsch–Jozsa algorithm, we represent the four possible coins by four functions f that map one input bit ($a = 0{,}1$ standing for 'which side of the coin') onto one output bit ($f(a) = 0{,}1$ standing for 'head or tail'). These functions can be divided into two constant functions $f_1(a) = 0$, $f_2(a) = 1$, representing the fake coins, and two balanced functions $f_3(a) = a$, $f_4(a) =$ NOT a, which stand for the fair coins (see Table 1). An unknown function is characterized as constant or balanced by evaluating $f(0) \oplus f(1)$ which yields 0 (or 1) for a constant (or balanced) function ($\oplus$ denotes addition modulo 2). This evaluation classically requires two function calls, whereas the Deutsch–Jozsa quantum algorithm allows us to obtain the desired information with a single evaluation of the unknown f. The circuit diagram shown in Fig. 1 describes the implementation of the Deutsch–Jozsa algorithm with basic quantum operations[21]. The two qubits required for the Deutsch–Jozsa algorithm are encoded in the electronic state and in the phonon (vibrational quantum) number of the axial vibration mode of the single trapped ion (see Fig. 2). Qubit operations are realized by applying laser pulses on the 'carrier' or the 'blue sideband' of the electronic quadrupole transition as described in the Methods.

In general, a quantum algorithm is implemented by a sequence of such pulses on the carrier and sideband, but two major sources of error have to be overcome. First, as the simplest algorithms already require several pulses, we need to control precisely the relative optical phases of these pulses or, at least, to keep track of them such that the required pulse sequences lead to the desired operations. In particular, this requires the precise investigation and subsequent compensation of all phases introduced by the light shifts of the exciting laser beams. These light shifts arise as we have to drive

Table 1 **Truth table for the four possible functions**

	Constant functions		Balanced functions	
	Case 1	Case 2	Case 3	Case 4
$f(0)$	0	1	0	1
$f(1)$	0	1	1	0
$w \oplus f(a)$	ID	NOT	CNOT	Z-CNOT

The third line is the effect of the logic function U_{f_n} on the qubit w: ID denotes the identity, CNOT is a controlled NOT operation, Z-CNOT is a zero controlled NOT, and the control bit in cases 3 and 4 is the input bit a.

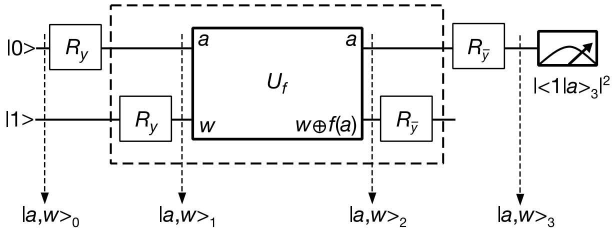


Figure 1 Quantum circuit for implementing the Deutsch–Jozsa algorithm with basic quantum operations. The upper line shows the input qubit $|a\rangle$ ('which side of the coin' information), the lower line an auxiliary working qubit $|w\rangle$ (corresponding to the channel on which the answer is provided). The rotations R_y (see Methods for details) create superpositions $|a\rangle_1 = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|w\rangle_1 = (|0\rangle - |1\rangle)/\sqrt{2}$ from the inputs $|a\rangle_0 = |0\rangle$ and $|w\rangle_0 = |1\rangle$. The box U_{f_n} represents a unitary operation specific to each of the functions f_n, which applies f_n to a and adds the result to w modulo 2. Table 1 lists the logic operations required for transforming $|w\rangle$ into $|w \oplus f_n(a)\rangle$. The output of the box is $|a, w\rangle_2 = (|0, w_{\mathrm{in}} \oplus f_n(0)\rangle + |1, w_{\mathrm{in}} \oplus f_n(1)\rangle)/\sqrt{2}$. Up to an overall sign $|w\rangle$ is left unchanged, but the positive superposition $(|0\rangle + |1\rangle)/\sqrt{2}$ on $|a\rangle$ is transformed into a negative superposition $|a\rangle_2 = (|0\rangle - |1\rangle)/\sqrt{2}$ if f is balanced; otherwise it is unchanged. After the final rotations $R_{\bar{y}}$, a measurement on $|a\rangle$ is performed with result $|a\rangle_3 =$ either $|0\rangle$ or $|1\rangle$. Because of the sign change in $|a\rangle_2$ if f is balanced, $|\langle 1 \mid a\rangle_3|^2 = f_n(0) \oplus f_n(1)$, that is, $|a\rangle_3$ yields the desired information whether the function f_n is balanced or constant. The working qubit w resumes its initial value $|w\rangle_3 = |w\rangle_0 = |1\rangle$.

sideband transitions (which couple much more weakly than carrier transitions) with high laser intensity. We cancel the unwanted light shifts with an additional off-resonant laser field, inducing a light shift of equal strength but opposite sign[20].

Second, a peculiarity of encoding a qubit within the ion's motional state is that we must ensure that the system does not leave the computational subspace $\{|S,0_z\rangle,\ |D,0_z\rangle,\ |S,1_z\rangle, |D,1_z\rangle|\}$ (for notation, see Methods). The main problem here is that owing to the degenerate spectrum of a harmonic oscillator, sideband pulses work simultaneously on all levels. Therefore any population in $|S,1_z\rangle$ prior to a blue sideband pulse will leave the computational subspace. To avoid this, we use composite pulses, that is, a sequence of carrier and/or sideband pulses that—up to an overall phase—constrain the system to the subspace[10]. We adopted this method from NMR technology[11]. The translation of the Deutsch—Jozsa algorithm into composite pulses acting on the two qubits is described in the Methods.

For our experiments we load Ca ions into a linear Paul trap with axial frequency $\omega_z \approx 2\pi \times 1.7\,\mathrm{MHz}$. Figure 2 shows the relevant optical transitions used for laser cooling, state preparation and detection. Each experimental cycle starts with Doppler cooling for 2 ms on the $S_{1/2} \rightarrow P_{1/2}$ transition yielding average vibrational quantum numbers $\bar{n}_z \approx 20$. Further cooling of the axial motion to a ground state occupation of more than 99% is achieved by about 12 ms of sideband cooling[8]. To initalize the quantum processor in $|01\rangle = |S,0_z\rangle$, we optically pump the ion to the $S_{1/2}$ $(m = -1/2)$ state. Manipulations of both qubits are achieved by pulses from a stabilized titanium–sapphire laser (linewidth $< 100\,\mathrm{Hz}$, relative intensity noise $< 0.02_{\mathrm{r.m.s.}}$) emitting at the $S_{1/2} \leftrightarrow D_{5/2}$ transition wavelength near 729 nm. In order to switch between R and R^+ rotations we shift the laser frequency with an acousto-optical modulator. The phase of the light field is switched via the phase of the radio frequency driving the acousto-optical modulator with an inaccuracy of less than 0.06 rad. Using the electron shelving technique[8] we detect the ion's electronic state ($S_{1/2}$ or $D_{5/2}$) with a fidelity of 99.9% within a detection time of 3 ms.

We measure the fidelity of the implemented algorithm by repeating several thousand times the experimental sequence of cooling, initialization of both qubits, laser pulses for the algorithm and final measurement. Table 2 displays the achieved results. For cases 1, 3 and 4, the fidelity of identifying the function's class with a single measurement exceeds 97%; for case 2, it is above 90%. Note that to decide whether the function is constant or balanced, only $|\langle 1 \mid a\rangle_3|^2$ at the end of the algorithm needs to be measured. We also verified that the working qubit $|w\rangle$ is reset to its initial value by reading out the phonon number through a measurement of the Rabi frequency of the blue sideband transition[8,16].

Table 2 **Expected and measured results of the complete Deutsch–Jozsa algorithm**

	Constant		Balanced	
	Case 1	Case 2	Case 3	Case 4
Expected $|\langle 1 \mid a\rangle|^2$	0	0	1	1
Measured $|\langle 1 \mid a\rangle|^2$	0.019(6)	0.087(6)	0.975(4)	0.975(2)
Expected $|\langle 1 \mid w\rangle|^2$	1	1	1	1
Measured $|\langle 1 \mid w\rangle|^2$	–	0.90(1)	0.931(9)	0.986(4)

The numbers in brackets are statistical 1σ uncertainties.

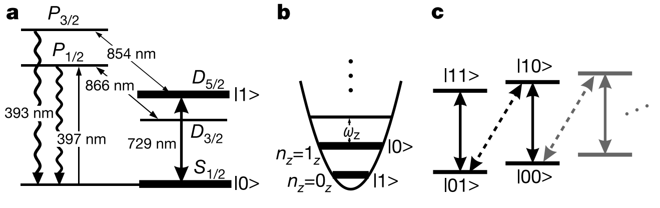


Figure 2 Quantum mechanical energy levels relevant for the ion-trap quantum computer. **a**, Ca$^+$ level scheme. The upper and lower electronic states $S_{1/2}$ $(m = -1/2)$ and $D_{5/2}$ $(m = -1/2)$ of the narrow quadrupole transition ($\tau_D \approx 1$ s) at 729 nm serve to implement one of the qubits, $|a\rangle$. Coherent radiation of a titanium–sapphire laser at 729 nm drives the qubit transition. Lasers at 397 nm, 866 nm and 854 nm are used for the excitation of resonance fluorescence, for Doppler cooling, and optical pumping. The laser system is described in detail elsewhere[19]. **b**, The lowest two number states, $n_z = 0_z, 1_z$, of the axial vibrational motion in the trap form the other qubit, $|w\rangle$. **c**, The combination of electronic states and energy eigenstates of the harmonic oscillator potential span the computational subspace. Numbers in ket notation denote the quantum logical values assigned to the respective states. Solid lines show carrier transitions; dashed lines show blue sideband transitions.

The measured output of the algorithm shown in Table 2 slightly deviates from the ideal result. We identified the major sources for this infidelity and attribute it mainly to decoherence of the laser-atom phase, in particular caused by ambient magnetic field fluctuations[22]. Furthermore, in the implementation of case 2, which requires the most complex pulse sequence, we used higher laser power of the sideband transitions in order to speed up the algorithm and thus reduce the sensitivity to phase decoherence. This in turn caused off-resonant carrier excitation which limited the obtainable fidelity.

A major advantage of our state detection technique is the ability to follow the evolution of $|\langle 1 \mid a\rangle|^2$ during the quantum algorithm. For this, we truncate the pulse sequence at a certain time t and reveal $|\langle 1 \mid a(t)\rangle|^2$ by measuring the probability of finding the ion in the $D_{5/2}$ state. In Fig. 3 we display this probability as a function of time for all four cases. The data agree very well with the calculated ideal

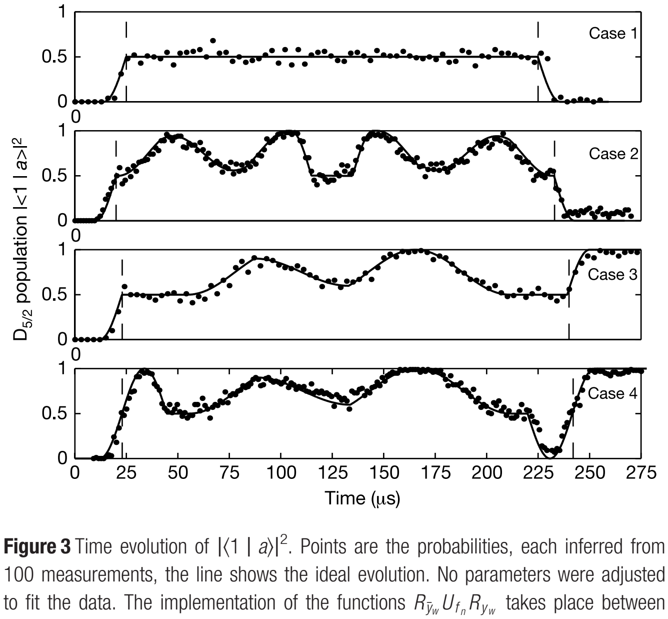


Figure 3 Time evolution of $|\langle 1 \mid a\rangle|^2$. Points are the probabilities, each inferred from 100 measurements, the line shows the ideal evolution. No parameters were adjusted to fit the data. The implementation of the functions $R_{\bar{y}_w} U_{f_n} R_{y_w}$ takes place between the dashed lines. An initial R_{y_a} and a final $R_{\bar{y}_a}$ rotation on $|a\rangle$, implemented by carrier pulses, complete the algorithm. Taking case 3 as an the example, R_{y_a} lasts from 12 μs to 22 μs. Then $R_{\bar{y}_w} U_{f_n} R_{y_w}$ on $|a,w\rangle$ is implemented from 54 μs to 212 μs with the laser tuned to the blue sideband. The laser phase is switched at 87, 133 and 166 μs according to Table 3. The final $R_{\bar{y}_a}$ pulse is applied from 240 to 250 μs.

Table 3 **Implementations of $R_{\bar{y}_w} U_{f_n} R_{y_w}$**

	Logic	Laser pulses
f_1	$R_{\bar{y}_w} R_{y_w}$	No pulses
f_2	$R_{\bar{y}_w}$ SWAP^{-1} NOT$_a$ SWAP R_{y_w}	$R^+\left(\frac{\pi}{\sqrt{2}},0\right)R^+\left(\frac{2\pi}{\sqrt{2}},\varphi_{\mathrm{SWAP}}\right)R^+\left(\frac{\pi}{\sqrt{2}},0\right)$ $R(\frac{\pi}{2},0)R(\pi,\frac{\pi}{2})R(\frac{\pi}{2},\pi)$ $R^+\left(\frac{\pi}{\sqrt{2}},\pi\right)R^+\left(\frac{2\pi}{\sqrt{2}},\pi+\varphi_{\mathrm{SWAP}}\right)R^+\left(\frac{\pi}{\sqrt{2}},\pi\right)$
f_3	$R_{\bar{y}_w}$ CNOT R_{y_w}	$R^+\left(\frac{\pi}{\sqrt{2}},0\right)R^+\left(\pi,\frac{\pi}{2}\right)R^+\left(\frac{\pi}{\sqrt{2}},0\right)R^+\left(\pi,\frac{\pi}{2}\right)$
f_4	$R_{\bar{y}_w}$ Z-CNOT R_{y_w}	$R(\pi,0)R^+\left(\frac{\pi}{\sqrt{2}},0\right)R^+\left(\pi,\frac{\pi}{2}\right)R^+\left(\frac{\pi}{\sqrt{2}},0\right)R^+\left(\pi,\frac{\pi}{2}\right)R(\pi,0)$

The rotation angle for $R^+(\theta,\varphi)$ is given for the $|10\rangle \rightarrow |01\rangle$ transition. θ and φ denote the pulse duration and phase, respectively. $\varphi_{\mathrm{SWAP}} = \arccos(\cot^2(\pi/\sqrt{2}))$, where the SWAP operation is explained in the Methods.

evolution (solid lines in Fig. 3, no fit parameters), demonstrating the high precision of the applied pulse sequence, especially the control over the optical phases.

The results demonstrate a high degree of control of all relevant experimental parameters, that is, laser frequency and intensity, optical phases, and trap frequency ω_z, over long pulse sequences. Therefore, the procedures presented here pave the way for implementing more complex algorithms and for scaling the system to multi-qubit operation. In particular, the light shift compensation technique demonstrated in this experiment can be directly transferred and advantageously applied to a several-qubit quantum processor. This technique will become increasingly important for scaling such a system because as the ion crystal becomes heavier, the higher laser intensities required to drive sideband transitions result in increased light shifts. Furthermore, by merging the composite pulse technique with our trapped-ion quantum computer we gain full access to all gate operations on the motional qubit. The employed composite-pulse phase gate also simplifies the Cirac–Zoller scheme[5] for a universal set of quantum gates, by dispensing with the auxiliary level transition. Thus our procedures become applicable to a wider choice of ion species including ^{43}Ca$^+$, which offers a potentially much longer coherence time than ^{40}Ca$^+$. □

Methods

Encoding of qubits and single-qubit rotations

The two qubits required for the Deutsch–Jozsa algorithm are encoded in the electronic quantum state ($S_{1/2}$ $(m = -1/2) \equiv |0\rangle \equiv |S\rangle$ and $D_{5/2}$ $(m = -1/2) \equiv |1\rangle \equiv |D\rangle$) and in the phonon number of the axial vibration mode of the single trapped ion ($n_z = 0_z \equiv |1\rangle$ and $n_z = 1_z \equiv |0\rangle$. Note the counterintuitive encoding of the vibrational mode, which simplifies the desired initial state preparation in $|01\rangle = |S, 0_z\rangle$. The operations which modify the electronic qubit ('single-qubit rotations') are performed with laser pulses on the carrier ($|S, n_z\rangle \leftrightarrow |D, n_z\rangle$) transition, that is, no change of vibrational quantum number, laser on resonance. To connect the two qubits ('two-qubit rotations') the laser is detuned by $+\omega_z$ from the $|S\rangle \leftrightarrow |D\rangle$ resonance to the 'blue sideband' ($|S, n_z\rangle \leftrightarrow |D, n_z + 1\rangle$) as indicated in Fig. 2. Qubit rotations can be written as unitary operations in the following way[12]:

Carrier rotations are given by

$$R(\theta, \phi) = \exp\left[i\frac{\theta}{2}(\mathrm{e}^{i\phi}\sigma^+ + \mathrm{e}^{-i\phi}\sigma^-)\right]$$

whereas transitions on the blue sideband are denoted as

$$R^+(\theta, \phi) = \exp\left[i\frac{\theta}{2}(\mathrm{e}^{i\phi}\sigma^+ b^\dagger + \mathrm{e}^{-i\phi}\sigma^- b)\right]$$

Here $\sigma^\pm$ are the atomic raising and lowering operators which act on the electronic quantum state of the ion, that is, the first qubit, by inducing transitions from the $|S\rangle$ to $|D\rangle$ state and vice versa (notation: $\sigma^+ = |D\rangle\langle S|$). The operators b and $b^\dagger$ stand for the annihilation and creation of a phonon at the trap frequency, that is, they work on the motional quantum state, the second qubit. The parameter θ depends on the strength and the duration of the applied pulse and ϕ is its phase, that is, the relative phase between the optical field and the atomic polarization. We use the definitions $R_y = R(\pi/2, 0)$ and $R_{\bar{y}} = R(\pi/2, \pi)$.

Translation of the Deutsch–Jozsa algorithm into composite pulses

The quantum circuit shown in Fig. 1 shows the quantum logic operations used for the implementation and Table 1 lists the logic functions corresponding to the unitary operations U_{f_n}. The R_y rotations on the electronic qubit $|a\rangle$ are carrier pulses. For efficient computation we combine the rotations $R_{\bar{y}}, R_y$ on $|w\rangle$ and the manipulations for implementing U_{f_n} into an optimized pulse sequence, $R_{\bar{y}_w} U_{f_n} R_{y_w}$ (dashed box in Fig. 1). As these operations act also on the motional state, we implement them with pulses on the carrier and the blue axial sideband. However, sideband pulses operate on both qubits simultaneously. Thus, for operations on $|w\rangle$ alone, we first swap the information from $|w\rangle$ into $|a\rangle$ with a sequence of three blue sideband pulses, then we rotate $|a\rangle$ as desired and swap back.

For a swap operation one might be tempted to use a single π-pulse on the blue sideband. However, applying this to the state $|00\rangle = |S, 1_z\rangle$ leads to a population of states with two phonons outside the computational subspace. Therefore we use a composite pulse sequence consisting of three pulses, whose lengths are chosen such that starting from $|S,1_z\rangle$ the ion is rotated by π,2π and π, respectively. As a result the ion is rotated by 4π back to $|S,1_z\rangle$ independently of the pulses' relative phases. In addition, using the blue sideband ensures that $|11\rangle \equiv |D, 0_z\rangle$ also stays unchanged as required for the swap operation.

The desired swap operation $|S, 0_z\rangle \leftrightarrow |D, 1_z\rangle$ is possible because compared to the $|S, 1_z\rangle \leftrightarrow |D, 2_z\rangle$ transition, the Rabi frequency for the $|S, 0_z\rangle \leftrightarrow |D, 1_z\rangle$ transition is smaller by $1/\sqrt{2}$ (refs 8, 16). So in this manifold the three pulses' lengths correspond to rotation angles of $\pi/\sqrt{2}, 2\pi/\sqrt{2}, \pi/\sqrt{2}$. It can be shown that choosing the laser-atom phase of the second pulse to be $\mathrm{arcos}(\cot^2(\pi/\sqrt{2})) = \pi 0.3033\ldots$ relative to the first and the third pulses, the populations of $|10\rangle = |D, 1_z\rangle$ and $|01\rangle = |S, 0_z\rangle$ are exchanged. This realises the desired swap. Table 3 (case 2) lists the complete pulse sequence for the implementation of $R_{\bar{y}_w} U_{f_2} R_{y_w}$. Similar procedures are applied to realise the pulse sequences for cases 3 and 4. In these cases the rotations $R_{\bar{y}_w}, R_{y_w}$ and the operations required for U_{f3}, U_{f4} can be combined in such a way that swap operations become unnecessary.

Received 21 October; accepted 27 November 2002; doi:10.1038/nature01336.

1. Deutsch, D. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. R. Soc. Lond. A* **400,** 97–117 (1985).
2. Deutsch, D. & Jozsa, R. Rapid solution of problems by quantum computation. *Proc. R. Soc. Lond. A* **439,** 553–558 (1992).
3. Chuang, I. I., Vandersypen, I. M. K., Zhou, X., Leung, D. W. & Lloyd, S. Experimental realization of a quantum algorithm. *Nature* **393,** 143–146 (1998).
4. Jones, T. F. & Mosca, M. Implementation of a quantum algorithm to solve Deutsch's problem on a nuclear magnetic resonance quantum computer. *J. Chem. Phys.* **109,** 1648–1653 (1998).
5. Cirac, J. I. & Zoller, P. Quantum computations with cold trapped ions. *Phys. Rev. Lett.* **74,** 4091–4094 (1995).
6. Monroe, C., Meekhof, D. M., King, B. E., Itano, W. M. & Wineland, D. J. Demonstration of a fundamental quantum logic gate. *Phys. Rev. Lett.* **75,** 4714–4717 (1995).
7. Sackett, C. A. *et al.* Experimental entanglement of four particles. *Nature* **404,** 256–259 (2000).
8. Roos, Ch. *et al.* Quantum state engineering on an optical transition and decoherence in a Paul trap. *Phys. Rev. Lett.* **83,** 4713–4716 (1999).
9. Meyer, V. *et al.* Experimental demonstration of entanglement-enhanced rotation angle estimation using trapped ions. *Phys. Rev. Lett.* **86,** 5870–5873 (2001).
10. Childs, A. M. & Chuang, I. M. Universal quantum computation with two-level trapped ions. *Phys. Rev. A* **63,** 012306 (2001).
11. Levitt, M. H. Composite pulses (NMR spectroscopy). *Prog. Nucl. Magn. Reson. Spectrosc.* **18,** 61–122 (1986).
12. Šašura, M. & Bužek, V. Cold trapped ions as quantum information processors. *J. Mod. Opt.* **49,** 1593–1647 (2002).
13. Eschner, J., Raab, Ch., Schmidt-Kaler, F. & Blatt, R. Light interference from single atoms and their mirror images. *Nature* **413,** 495–498 (2001).
14. Guthöhrlein, G. R., Keller, M., Hayasaka, K., Lange, W. & Walther, H. A single ion as a nanoscopic probe of an optical field. *Nature* **414,** 49–51 (2001).
15. Mundt, A. B. *et al.* Coupling a single atomic quantum bit to a high finesse optical cavity. *Phys. Rev. Lett.* **89,** 103001 (2002).
16. Meekhof, D. M., Monroe, C., King, B. E., Itano, W. M. & Wineland, D. J. Generation of nonclassical motional states of a trapped atom. *Phys. Rev. Lett.* **76,** 1796–1799 (1996).
17. Monroe, C., Meekhof, D. M., King, B. E. & Wineland, D. J. A "Schrödinger Cat" superposition state of an atom. *Science* **272,** 1131–1136 (1996).
18. Dehmelt, H. Proposed $10^{14}\Delta\nu > \nu$ laser fluorescence spectroscopy on TI$^+$ mono-ion oscillator. *Bull. Am. Phys. Soc.* **20,** 60 (1975).
19. Nägerl, H. C. *et al.* Investigating a qubit candidate: Spectroscopy on the S$_{1/2}$ to D$_{5/2}$ transition of a trapped calcium ion in a linear Paul trap. *Phys. Rev. A* **61,** 023405 (2000).
20. Häffner, H. *et al.* Precision measurement and compensation of optical Stark shifts for an ion-trap quantum processor. Preprint available at ⟨http://arXiv.org/abs/physics/0212040⟩ (2002).
21. Nielsen, M. A. & Chuang, I. J. *Quantum Computation and Quantum Information* (Cambridge Univ. Press, Cambridge, 2000).
22. Schmidt-Kaler, F. *et al.* Coherence of qubits based on single Ca ions. Preprint available at ⟨http://arXiv.org/abs/quant-ph/0211059⟩ (2002).

Acknowledgements We gratefully acknowledge support by the European Commission (QSTRUCT, QI, QUEST and QUBITS networks), by the Austrian Science Fund (FWF), and by the Institut für Quanteninformation GmbH.

Competing interests statement The authors declare that they have no competing financial interests.

Correspondence and requests for materials should be addressed to F.S.-K. (e-mail: Ferdinand.Schmidt-Kaler@uibk.ac.at).

A.8 PRL'03 Paper

Experimental Implementation of an Adiabatic Quantum Optimization Algorithm

Matthias Steffen,[1,2,*] Wim van Dam,[3,4] Tad Hogg,[3] Greg Breyta,[5] and Isaac Chuang[1]
[1]*Center for Bits and Atoms–MIT, Cambridge, Massachusetts 02139*
[2]*Solid State and Photonics Laboratory, Stanford University, Stanford, California 94305-4075*
[3]*HP Labs, Palo Alto, California 94304-1126*
[4]*MSRI, Berkeley, California 94720-5070*
[5]*IBM Almaden Research Center, San Jose, California 95120*

(Received 1 October 2002; published 14 February 2003)

We report the realization of a nuclear magnetic resonance computer with three quantum bits that simulates an adiabatic quantum optimization algorithm. Adiabatic quantum algorithms offer new insight into how quantum resources can be used to solve hard problems. This experiment uses a particularly well-suited three quantum bit molecule and was made possible by introducing a technique that encodes general instances of the given optimization problem into an easily applicable Hamiltonian. Our results indicate an optimal run time of the adiabatic algorithm that agrees well with the prediction of a simple decoherence model.

DOI: 10.1103/PhysRevLett.90.067903 PACS numbers: 03.67.Lx, 03.65.Yz, 76.60.–k

Since the discovery of the algorithms of Shor [1] and Grover [2], the quest of finding new quantum algorithms proved a formidable challenge. Recently, however, a novel algorithm was proposed, using adiabatic evolution [3,4]. Despite the uncertainty in its scaling behavior, this algorithm remains a remarkable discovery because it offers new insights into the potential usefulness of quantum resources for computational tasks.

Experimental realizations of quantum algorithms in the past demonstrated Grover's search algorithm, the Deutsch-Jozsa algorithm, order finding, and Shor's algorithm [5,6]. Recently, Hogg's algorithm was implemented using only one computational step [7]; however, a demonstration of an adiabatic quantum algorithm thus far has remained beyond reach.

Here, we provide the first experimental implementation of an adiabatic quantum optimization algorithm using three qubits and nuclear magnetic resonance (NMR) techniques [8]. NMR techniques are especially attractive because several tens of qubits may be accessible, which is precisely the range that could be crucial in determining the scaling behavior of adiabatic quantum algorithms [9]. Compared to earlier implementations of search problems [5,10], this experiment is a full implementation of a true optimization problem which does not require a black box function or ancilla bits.

This experiment was made possible by overcoming two experimental challenges. First, an adiabatic evolution requires a smoothly varying Hamiltonian over time, but the terms of the available Hamiltonian in our system cannot be smoothly varied and may even have fixed values. We developed a method to approximately smoothly vary a Hamiltonian despite the given restrictions by extending NMR average Hamiltonian techniques [11]. Second, general instances of the optimization algorithm may require the application of Hamiltonians that are not easily accessible. We developed methods to implement general instances of a well-known classical NP-complete (nondeterministic, polynomial time) optimization problem given a fixed natural system Hamiltonian.

We provide a concrete procedure detailing these methods. We then apply the results to the Maximum Cut (MAXCUT) [12] optimization problem. Our experiments indicate there exists an optimal total running time which can be predicted using a decoherence model based on independent stochastic relaxation of the spins.

An adiabatic quantum algorithm evolves the quantum state with a slowly varying, time-dependent Hamiltonian. Suppose we are given some time-dependent Hamiltonian $H(t)$, where $0 \leq t \leq T$, and at $t = 0$ we start in the ground state of $H(0)$. By varying $H(t)$ slowly, the quantum system remains in the ground state of $H(t)$ for all $0 \leq t \leq T$ provided the lowest two energy eigenvalues of $H(t)$ are never degenerate [13]. Now suppose we can encode an optimization problem into $H(T)$. Then the state of the quantum system at time $t = T$ represents the solution to the optimization problem [3]. The total run time T of the adiabatic algorithm scales as $g_{\min}^{-2}$, where $g_{\min}$ is the minimum separation between the lowest two energy eigenvalues of $H(t)$ [3,14]. The scaling behavior of $g_{\min}$ will ultimately determine the success of adiabatic quantum algorithms. Classical simulations of this scaling behavior are hard due to the exponentially growing size of Hilbert space. In contrast, sufficiently large quantum computers could simulate this behavior efficiently.

Smoothly varying some time-dependent Hamiltonian appears straightforward but contrasts with the traditional picture of discrete unitary operations including fault tolerant quantum circuit constructions [15]. Fortunately, we can approximate a smoothly varying Hamiltonian using methods of quantum simulations [16] and recast adiabatic evolution in terms of unitary operations.

Discretizing a continuous Hamiltonian is a straightforward process and changes the run time T of the adiabatic algorithm only polynomially [14]. For simplicity, let the discrete time Hamiltonian $H[m]$ be a linear interpolation from some beginning Hamiltonian $H[0] = H_b$ to some final problem Hamiltonian $H[M] = H_p$ such that $H[M] = (m/M)H_p + (1 - m/M)H_b$. The unitary evolution of the discrete algorithm can be written as

$$U = \prod_m U_m = \prod_m e^{-i[(1-m/M)H_b+(m/M)H_p]\Delta t}, \qquad (1)$$

where $\Delta t = T/(M + 1)$, and $M + 1$ is the total number of discretization steps. The adiabatic limit is achieved when both $T, M \rightarrow \infty$ and $\Delta t \rightarrow 0$.

Full control over the strength of H_b and H_p is needed to implement Eq. (1). However, this may not necessarily be a realistic experimental assumption. We will next show how the discrete time adiabatic algorithm can still be implemented when H_b and H_p cannot both be applied simultaneously *and* when they are both fixed in strength.

When both H_b and H_p are fixed, we can approximate U_m to second order by using the Trotter formula $\exp[(A + B)\Delta t] = \exp(A\Delta t/2)\exp(B\Delta t)\exp(A\Delta t/2) + \mathcal{O}(\Delta t^2)$ [16]. Higher order approximations can be constructed if more accuracy is required.

Now suppose H_b and H_p are both constant. Since any unitary matrix is generated by an action $-iH\Delta t$, we can increase the effect of a constant Hamiltonian H by lengthening the time Δt. Thus, we can implicitly increase the strength of H_b and H_p even when they are constant by simply increasing the time during which they are applied.

This technique also allows cases when the accessible Hamiltonians are not of the required strength, for example, when we are given $H_b' = gH_b$ and $H_p' = hH_p$ but still wish to implement H_b and H_p. Using all of the described techniques, we can now write U_m as

$$U_m \approx e^{-iH_b'[(1-m/M)\Delta t/2g]} \circ e^{-iH_p'[(m/M)\Delta t/h]}, \qquad (2)$$

where $A \circ B = ABA$. Each discretization step is of length $(1 - m/M)\Delta t/g + (m/M)\Delta t/h$, which is not constant when $g \neq h$. As an illustration consider Fig. 1(a).

We choose $\Delta t = T/(M + 1)$ to be constant as we vary the number of discretization steps $M + 1$. This way, the total run time T increases with $M + 1$, allowing us to test the behavior of the algorithm when approaching one of the conditions for the adiabatic limit. Even when the discrete approximation is not close to the adiabatic limit, the implemented algorithm can often find solutions using relatively few steps but lacks the guaranteed performance of the adiabatic theorem [17].

Adiabatic evolution has been proposed to solve general optimization problems, including NP-complete ones. In this general setting, the algorithm can depend on the existence of a black box function or the usage of large amounts of workspace. Our goal here is to optimize a hard natural problem in a way that avoids these difficulties. We will first describe which problem we chose and later explain why it does not require ancilla qubits.

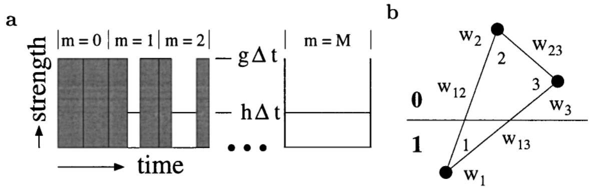


FIG. 1. (a) Illustration of Eq. (2). The shaded and clear boxes denote the strength and duration of the Hamiltonians H_b and H_p, respectively. (b) Illustration of a graph consisting of three nodes and three edges. The edges carry weights w_{12}, w_{13}, and w_{23}. When $\min(w_{ij}) = w_{23}$ as indicated by the length of the edges, the MAXCUT corresponds to the drawn cut. The solution is therefore $s = 100$ and also $s = 011$ due to symmetry. This symmetry can be broken by assigning the weights w_1, w_2, and w_3 to the nodes.

We found the MAXCUT problem to be a well-suited problem to demonstrate an adiabatic quantum algorithm because it allows a variety of interesting test cases. It also appears in the study of spin glasses [18], among others. The decision variant of the MAXCUT problem is part of the core NP-complete problems [12], and even the approximation within a factor of 1.0624 of the perfect solution is NP complete [19].

The MAXCUT problem can be understood as follows. A *cut* is defined as the partitioning of an undirected n-node graph with edge weights into two sets. We define the payoff as the sum of weights of edges crossing the cut. The maximum cut is a cut that maximizes this payoff. By assigning either $s_i = 0$ or $s_i = 1$ to each node i, depending on its location with respect to the cut, the MAXCUT problem can be restated as finding the n-bit number s that maximizes the payoff. An extension of the MAXCUT problem is to let the nodes themselves carry weights, which can be regarded as the nodes having a preference on their location. As an illustration consider a graph with three nodes as drawn in Fig. 1(b).

The payoff as a function of the cut defined by s is

$$P(s) = \sum_i w_i s_i + \sum_{i,j} s_i(1 - s_j)w_{ij}, \qquad (3)$$

where w_{ij} are the edge weights, w_i denotes the node weights, and s_i is the value of the ith bit of s.

The smallest meaningful test case of the MAXCUT problem requires three nodes and admits a variety of interesting cases by varying w_i and w_{ij}. We aimed at two goals when choosing a representative set of weights. First, we wanted the minimum energy gap $g_{\min}$ to be smaller than the one for a three-qubit adiabatic Grover search. Second, we wanted a resulting energy landscape with both a global and local maximum such that a greedy classical search would incorrectly find the local maximum half the time [20]. These goals are met by the choice

$w_1 = w_2 = w_3 = 2$, $w_{12} = 2$, $w_{13} = 1$, and $w_{23} = 3$. The payoff function for this set of weights is $P(s) = [0\ 6\ 7\ 7\ 5\ \mathbf{9}\ 8\ 6]$, where $s = [000\ 001\ 010\ 011\ 100\ 101\ 110\ 111]$. The global maximum lies at $s = 101$ so the answer on the quantum computer following measurement should be $|101\rangle$, and not at the local maximum $s = 110$

In the quantum setting, this payoff function $P(s)$ can be encoded into the Hamiltonian H_p by rewriting Eq. (3) using Pauli matrices:

$$H_p = \sum_i w_i(I - \sigma_{zi})/2 + \sum_{i<j} w_{ij}(I - \sigma_{zi}\sigma_{zj})/2, \quad (4)$$

where I is the $2^n \times 2^n$ identity matrix and σ_{zi} is the Pauli Z matrix on spin i. The identity matrices in the equation above only lead to an overall phase which cannot be observed and, hence, they can be ignored. The diagonal values of Eq. (4) are equal to $P(s)$. Because of the direct encoding of $P(s)$ into H_p, no black box function or ancilla qubits are required, which makes this a full implementation of an optimization problem.

Similar to Eq. (4), the natural Hamiltonian of n weakly coupled spin-1/2 nuclei subject to a static magnetic field B_0 is well approximated by [21]

$$\mathcal{H} = -\sum_i \omega_i \sigma_{zi}/2 + \sum_{i<j} \pi J_{ij}\sigma_{zi}\sigma_{zj}/2 + \mathcal{H}_{\text{env}}, \quad (5)$$

where the first term represents the Larmor precession of each spin i about $-B_0$, and ω_i is its Larmor frequency. The second term describes the scalar spin-spin coupling of strength J_{ij} between spins i and j. The last term represents coupling to the environment, causing decoherence. Note the resemblances between $\mathcal{H}$ and H_p.

Despite the similarities, the spin-spin couplings of Eq. (5) are generally different from a randomly chosen set of weights. Therefore, we require a procedure to turn the fixed J_{ij} into any specified weights w_{ij}. This is achieved using refocusing schemes that are typically used to turn on only one of the couplings while turning all others off [21].

We have modified a refocusing scheme to effectively change the couplings to any arbitrary value. Consider the pulse sequence drawn in Fig. 2. Based on this scheme, we can derive the underconstrained system $(\alpha + \beta - \gamma - \delta)J_{12} = w_{12}$, $(\alpha - \beta - \gamma + \delta)J_{13} = w_{13}$, and $(\alpha - \beta + \gamma - \delta)J_{23} = w_{23}$, which can be solved for positive α, β, γ, and δ such that $J_{ij} \rightarrow w_{ij}$.

The single weights w_i are implemented by introducing a reference frame for each spin i which rotates about $-B_0$ at frequency $(w_i - w_i)/2$. In order to apply the single qubit rotations of our refocusing scheme on resonance, we apply the reference frequency shift only during the delay segment α, which we can always choose to be a positive value. Thus, H_p is implemented by applying the refocusing scheme from Fig. 2 while going off resonance during the delay segment α.

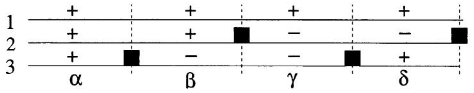


FIG. 2. Refocusing scheme to effectively change J_{ij} into w_{ij}. The horizontal lines denote qubits 1, 2, and 3 and time goes from left to right. The black rectangles represent 180° rotations. The delay segments are of length α, β, γ, and δ. When all segments are of equal length, all couplings are effectively turned off [22] because $\sigma_{xi}e^{-i\sigma_{zi}\sigma_{zj}t}\sigma_{xi} = e^{i\sigma_{zi}\sigma_{zj}t}$. In our experiment, $\alpha = 0.42$ ms, $\beta = 0$ ms, $\gamma = 4$ ms, and $\delta = 2.9$ ms in the last slice $M + 1$. The rf pulses that implement Hb' perform 33.75° rotations on the qubits in the first slice.

A full implementation of an adiabatic algorithm also requires a proper choice of H_b. We choose $H_b = \sum_i \sigma_{xi}$ for several reasons. First, its highest two excited states are nondegenerate. Second, it can be easily generated using single qubit rotations. Third, its highest excited state is created from a pure state with all qubits in the $|0\rangle$ state by applying a Hadamard gate on all qubits (we require the initial state to be the *highest excited* state of H_b because we are optimizing for the *maximum* value of H_p).

The full adiabatic quantum algorithm is now implemented by first creating the highest excited state of H_b. We then apply $M + 1$ unitary matrices as given by Eq. (2) and illustrated by Fig. 1(a). Accordingly, from slice to slice, we decrease the time during which H_b is active while increasing the time during which H_p is active. Finally, we measure the quantum system and read out the answer.

We selected ^{13}C-labeled CHFBr$_2$ for our experiments [10]. The Hamiltonian of the ^{1}H-^{19}F-^{13}C system is of the form of Eq. (5) with measured couplings $J_{\text{HC}} = 224$ Hz, $J_{\text{HF}} = 50$ Hz, and $J_{\text{FC}} = -311$ Hz. Experiments were carried out at MIT using an 11.7 Tesla Oxford Instruments magnet and a Varian Unity Inova spectrometer with a triple resonance (H-F-X) probe from Nalorac.

The experiments were performed at room temperature at which the thermal equilibrium state is highly mixed and cannot be turned into the required initial state by just unitary transforms. We thus first created an approximate effective pure state as in Ref. [10] by summing over three temporal labeling experiments.

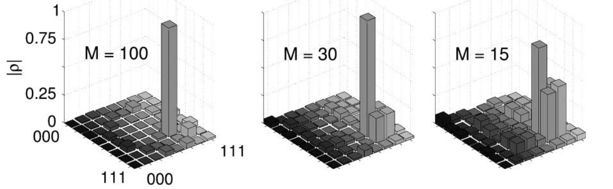


FIG. 3. Plot of the absolute value of the deviation density matrix for $M = 100$ ($T = 374$ ms), $M = 30$ ($T = 115$ ms), and $M = 15$ ($T = 59.2$ ms), adjusted by an identity portion such that the minimum diagonal value equals zero. The scale is arbitrary but the same for each plot.

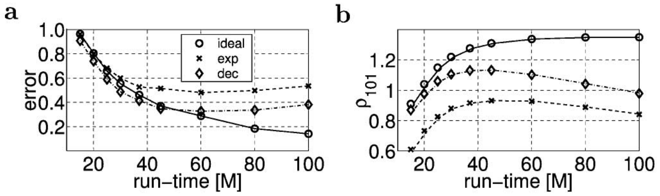


FIG. 4. Experimental performance of the adiabatic algorithm. (a) Plot of the error as a function of M. The error measure is the trace distance $D(\rho, \sigma) = |\rho - \sigma|/2$, where σ is the traceless deviation density matrix for $M = 400$, approximating $M \rightarrow \infty$, and ρ equals the ideal expected (○), the experimentally obtained (×), or the ideal expected traceless deviation density matrix with decoherence effects (◇) [6]. The minimum error occurs at about $M = 60$ indicating an optimal run time of the algorithm. (b) A similar observation can be made when plotting $|101\rangle\langle 101|$ as a function of M.

In our experiments, we actually implemented $0.5H_p$ and $0.5887H_b$ instead of H_p and H_b. This ensures that the error due to the second order Trotter approximation is sufficiently small. We also choose g so the applied rf field does not heat the sample, and $g \gg h$ so J_{ij} can be ignored when applying H_b. All of these choices result in a total experimental time that is within the shortest T_2 decoherence time [10]. We reconstructed the traceless deviation density matrices upon completion of the experiments using quantum state tomography [10].

We executed this algorithm for several M [with w_i and w_{ij} as listed above Eq. (4)]. Since we chose Δt to be constant, this meant increasing the run time T of the algorithm. The reconstructed deviation density matrices are shown in Fig. 3. The plots clearly display the expected pure state $|101\rangle$. The local maximum at $s = 110$ has a decreasingly small probability of being measured for increasing M. Simulations using Eq. (2) show that this optimization algorithm performs better for increasing M. We wanted to verify whether this is indeed true experimentally.

For this purpose, we estimate the error of our obtained deviation density matrices compared with the ideal case of $M = \infty$. Figure 4(a) plots the trace distance as a function of M, using the same arbitrary scale as in Fig. 3. From the plot, we observe there exists an optimal run time of the algorithm, corresponding to 0.226 s in our experiment. This optimal run time is in good agreement with the prediction of a previously developed simple decoherence model [6]. Predicting the impact of decoherence has already provided invaluable insight into estimating errors in previous experiments [6], and we believe continued effort towards understanding decoherence will greatly benefit experimental investigations of quantum systems.

In conclusion, we have provided the first experimental demonstration of an adiabatic quantum optimization algorithm. We show a concrete procedure turning a continuous time adiabatic quantum algorithm into a discrete time version, even when certain restrictions apply to the accessible Hamiltonians. Our results indicate that there exists an optimal run time of the algorithm which can be roughly predicted using a simple decoherence model. We believe this implementation opens the door to a variety of interesting experimental demonstrations and investigations of adiabatic quantum algorithms.

We wish to thank A. Childs, A. Landahl, and E. for useful discussions. This work was supported by the NSF Grant No. CCR-0122419, the DARPA QuIST program, and the HP/MSRI grant.

*Electronic address: msteffen@snowmass.stanford.edu

[1] P. Shor, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, CA, 1994), p. 124.

[2] L. K. Grover, Phys. Rev. Lett. **79**, 325 (1997).

[3] E. Farhi *et al.*, quant-ph/0001106.

[4] T. Hogg, Phys. Rev. A **61**, 052311 (2000).

[5] D. G. Cory *et al.*, Fort. Phys. **48**, 875 (2000); L. M. K. Vandersypen, Ph.D. thesis, Stanford University, 2001, and references therein.

[6] L. M. K. Vandersypen *et al.*, Nature (London) **414**, 883 (2001).

[7] X. Peng *et al.*, Phys. Rev. A **65**, 042315 (2002).

[8] N. Gershenfeld and I. L. Chuang, Science **275**, 350 (1997); D. Cory, A. F. Fahmy, and T. F. Havel, Proc. Natl. Acad. Sci. U.S.A. **94**, 1634 (1997).

[9] E. Farhi *et al.*, Science **292**, 472 (2001).

[10] L. M. K. Vandersypen *et al.*, Appl. Phys. Lett. **76**, 646 (2000).

[11] W.W. Rhim, A. Pines, and J. S. Waugh, Phys. Rev. Lett. **25**, 218 (1970).

[12] M. R. Garey, D. S. Johnson, and L. Stockmeyer, Theor. Comput. Sci. **1**, 237 (1976).

[13] A. Messiah, *Quantum Mechanics* (Wiley, New York, 1976).

[14] W. van Dam, M. Mosca, and U. Vazirani, in *Proceedings of the 42nd Annual Symposium on FOCS* (IEEE, Las Vegas, NV, 2001), p. 279.

[15] A. Aharonov and M. Ben-Or, in *Proceedings of the 29th Annual ACM STOC* (ACM, New York, 1997), p. 176.

[16] H. F. Trotter, Pacific J. Math. **8**, 887 (1958).

[17] T. Hogg, quant-ph/0206059.

[18] F. Barahona, J. Phys. A Math. Gen. **18**, L673 (1985).

[19] G. Ausiello *et al.*, *Complexity and Approximation. Combinatorial Optimization Problems and Their Approximability Properties* (Springer-Verlag, Berlin, 1999).

[20] A greedy search is done by first choosing a random node configuration s, and then repeatedly moving to a new configuration s' which differs from the previous configuration by only one node value s'_i and which also has the highest payoff, until the payoff is maximized.

[21] R. Ernst, *Principle of Nuclear Magnetic Resonance in One and Two Dimensions* (Oxford University Press, New York, 1994).

[22] D.W. Leung *et al.*, Phys. Rev. A **61**, 042310 (2000).

A.9 SPAA'03 Paper

The Effect of Communication Costs in Solid-State Quantum Computing Architectures

Dean Copsey[‡], Mark Oskin[†], Tzvetan Metodiev[‡],
Frederic T. Chong[‡], Isaac Chuang[◇], and John Kubiatowicz[○]
[‡] University of California at Davis, [†] University of Washington,
[◇] Massachusetts Institute of Technology, [○] University of California, Berkeley

ABSTRACT

Quantum computation has become an intriguing technology with which to attack difficult problems and to enhance system security. Quantum algorithms, however, have been analyzed under idealized assumptions without important physical constraints in mind. In this paper, we analyze two key constraints: the short spatial distance of quantum interactions and the short temporal life of quantum data.

In particular, quantum computations must make use of extremely robust error correction techniques to extend the life of quantum data. We present optimized spatial layouts of quantum error correction circuits for quantum bits embedded in silicon. We analyze the complexity of error correction under the constraint that interaction between these bits is near neighbor and data must be propagated via swap operations from one part of the circuit to another.

We discover two interesting results from our quantum layouts. First, the recursive nature of quantum error correction circuits requires a additional communication technique more powerful than near-neighbor swaps – too much error accumulates if we attempt to swap over long distances. We show that quantum teleportation can be used to implement recursive structures. We also show that the reliability of the quantum swap operation is the limiting factor in solid-state quantum computation.

Categories and Subject Descriptors

C.1 [**Processor Architectures**]: Miscellaneous; C.4 [**Performance of Systems**]: Fault Tolerance; C.5 [**Computer System Implementation**]: Miscellaneous; E.4 [**Coding and Information Theory**]: Error control codes

General Terms

Performance, Design, Algorithms, Reliability

Keywords

quantum computing, quantum architecture, silicon-based quantum computing

SPAA'03, June 7–9, 2003, San Diego, California, USA.
Copyright 2003 ACM 1-58113-661-7/03/0006 ...$5.00.

1. INTRODUCTION

Physical systems that behave quantum-mechanically have dynamics which can be exploited to speed up certain computational tasks. This is the essential thought behind the field of quantum computation and quantum information. A significant challenge arises in implementing quantum computation, however, because quantum systems are unstable: their quantum state is easily altered by omnipresent extraneous noise. This problem of *decoherence* was once thought to be a fundamental problem for quantum information processing [8], but the discovery of fault-tolerant constructions [1, 23, 27, 10] changed this; it is now known that an arbitrarily reliable quantum computer can be constructed from unreliable quantum wires and gates, as long as certain conditions are met. These constructions are made possible by recursive application of quantum error correction, generalizing the classical version of von Neumann's early constructions for reliable automata [37, 39].

The conditions for fault-tolerant quantum computation are as follows: First, the probability of failure of each elementary component must be less than some threshold value p_{th}, currently estimated to be around 10^{-4}. Second, current fault models assume that errors are independent and uniformly distributed (although other error models can also be dealt with by changing the scheme appropriately). Third, and most interesting, a variety of assumptions are made about both the quantum circuit and the necessary classical controller. In particular, it is essential that the quantum circuit employ maximum parallelism – executing as many quantum gates simultaneously as possible – and that the classical circuitry controlling the quantum operations run at a much higher clock speed than the quantum circuitry. Without these properties, p_{th} decreases significantly [1, 10].

Here, we take this study one step further, and consider the impact of *physical layout* on the requirements for fault-tolerant quantum computation. Do realistic physical implementations of these machines allow achievable fault-tolerance thresholds? In particular, what constraints must be satisfied in the architectural design of a quantum computer in order to allow a reliable machine to be realized?

Such questions can now be seriously considered in light of recent progress in the physical implementation of quantum computers, with a wide variety of systems ranging from spins in molecules [9] and single photons [18], to spins in semiconductors [16], trapped ions [19, **?**], and superconducting systems [36], among others. These systems have led to successful demonstrations of a wide variety of quantum information processing tasks, including quantum teleportation [4], creation of multiple quantum-bit entangled states [24], fast quantum search [7, 14], and recently, Shor's fast quantum factoring algorithm [35], in factoring the number fifteen, using a seven quantum bit (qubit) machine.

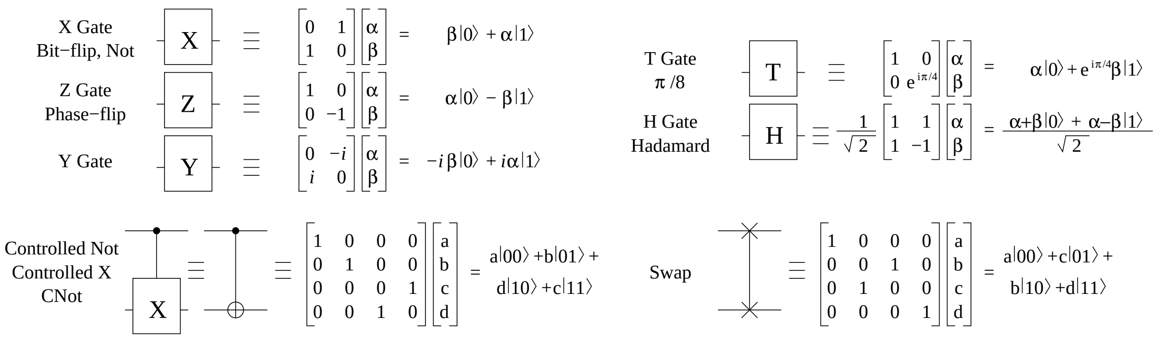


Figure 1: Basic quantum gates and their matrix representations

Among these implementations, the solid state systems are perhaps the most intriguing, because of the extensive investment that has been made in semiconductor technology for conventional classical computing, and the potential for scaling to large numbers of qubits. One such scheme, proposed by Kane, is particularly well suited for architectural study; it captures common elements from the whole range of implementations, using the nuclear spins of dopant atoms in silicon as qubits, classically controlled metal electrodes for control of quantum gates, and near neighbor, planar spin-spin interactions for multi-qubit gates. This scheme is also suitable for VLSI style CAD layout and modeling, and reveals an interesting constraint arising from pitch-matching large classical wires to small qubits, which forces computation units to be distributed in clusters rather than a single sea-of-qubits structure [22].

Our study of the architectural constraints on fault-tolerant quantum computation builds on the scenario posed by the Kane solid-state implementation proposal, and within this framework we obtain several interesting results. We first present complete layouts of qubits and gate sequences required to implement a concatenated seven-qubit Steane code for recursive quantum error correction. These layouts give us analytic expressions for the circuit's space and time resource requirements as a function of desired system reliability. We also consider the impact of planar near neighbor interactions on p_{th} and find that a huge limiting role will be played by a single gate, the SWAP gate, in determining achievable reliabilities.

We begin our study in the next two sections with a brief overview of quantum computation and error correction in quantum systems. In Section 4, we discuss the model we will be using for the rest of the paper, and the limitations it and similar models impose. Section 5 discusses implementations for error correction codes, while section 6 discusses the impact of communication on error correction algorithms. Finally, Section 7 discusses future work, while Section 8 concludes.

2. QUANTUM COMPUTATION

We begin with a brief overview of the basic terminology and constructs of quantum computation. Our purpose is to introduce the language necessary for subsequent sections; in-depth treatments of these subjects are available in the literature [21].

2.1 Quantum States: Qubits

The state of a classical digital system X can be specified by a binary string $\mathbf{x}$ composed of a number of bits x_i, each of which uniquely characterizes one elementary piece of the system. For n bits, there are 2^n possible states. The state of an analogous quantum system ψ is described by a complex-valued vector $|\psi\rangle = \sum_x c_x|\mathbf{x}\rangle$, a weighted combination (a "superposition") of the basis vectors $|\mathbf{x}\rangle$, where the *probability amplitudes* c_x are complex numbers whose modulus squared sums to one, $\sum_x |c_x|^2 = 1$.

A single quantum bit is commonly referred to as a *qubit* and is described by the equation $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$, where the c_i are complex valued. Legal qubit states include pure states, such as $|0\rangle$ and $|1\rangle$, and states in superposition, such as $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, or $\frac{1}{2}|0\rangle - i\frac{\sqrt{3}}{2}|1\rangle$. Larger quantum systems can be composed from multiple qubits, for example, $|00\rangle$, or $\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle - \frac{1}{\sqrt{2}}|11\rangle$. An n-qubit state is described by 2^n basis vectors, each with its own complex probability amplitude, so an n-qubit system can exist in an arbitrary superposition of the possible 2^n classical states of the system.

Unlike the classical case, however, where the total can be completely characterized by its parts, the state of larger quantum systems cannot always be described as the product of its parts. This property, known as *entanglement*, is best illustrated with an example: there exist no single qubit states $|\psi_A\rangle$ and $|\psi_B\rangle$ such that the two-qubit state $|\Psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ can be expressed as the composite state[1] $|\psi_A\rangle \otimes |\psi_B\rangle$. Entanglement and superposition have no classical analogues: they give quantum computers their computational powers.

Although a quantum system may exist in a superposition of states, only one of those states can be observed, or measured. After measurement, the system is no longer in superposition: the quantum state collapses into the one state measured, and probability amplitude of all other states goes to 0. For example, when the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ is measured, the result is either 00 or 11, with equal probability; the outcomes $|01\rangle$ or $|10\rangle$ never occur. Furthermore, if a subset of the qubits in a system is measured, the remaining qubits are left in a state consistent with the measurement.

Since measurement of a quantum system only produces a single result, quantum algorithms must maximize the probability that the result measured is the result desired. This may be accomplished by iteratively amplifying the desired result, as in Grover's fast database search, $O(\sqrt{n})$ for a dataset of size n [11]. Another option is to arrange the computation such that it does not matter which of many random results is measured from a qubit vector. This method is used in Shor's algorithm for factoring the product of two large primes [26], which is built upon modular exponentiation and a quantum Fourier transform. For the interested reader, quantum algorithms for a variety of problems other than search and factoring have been developed: adiabatic solution of optimization problems (the quantum analogue of simulated annealing; complex-

[1] The composition operator for quantum systems is the tensor product, $\otimes$: $|\mathbf{x}\rangle \otimes |\mathbf{y}\rangle = \sum_x c_x|x\rangle \otimes \sum_y c_y|y\rangle = \sum_{x,y} c_x c_y |x \otimes y\rangle$, where $x \otimes y$ is simply the string formed by concatenating x and y.

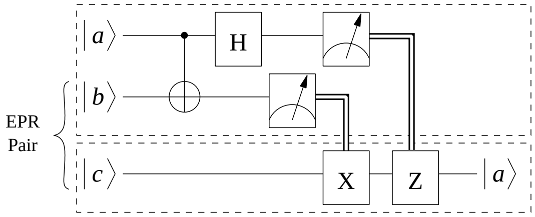


Figure 2: Quantum Teleportation: Quantum Teleportation of state $|a\rangle$. First, *entangled* qubits $|b\rangle$ and $|c\rangle$ are exchanged. Then, $|a\rangle$ is combined with $|b\rangle$ after which two *classical* bits of information (double lines) are produced via measurement ("meter" boxes). After transport, these bits are used to manipulate $|c\rangle$ to regenerate state $|a\rangle$ at destination.

ity unknown) [5], precise clock synchronization (using EPR pairs to synchronize GPS satellites) [15, 6], quantum key distribution (provably secure distribution of classical cryptographic keys) [3], and very recently, Gauss sums [33], testing of matrix multiplication (in $O(n^{1.75})$ steps versus the $O(n^2)$ required classically) [13], and Pell's equation [12].

2.2 Quantum Gates and Circuits

Just as classical bits are manipulated using gates such as NOT, AND, and XOR, qubits are manipulated with quantum gates such as those shown in Figure 1. A quantum gate is described by a unitary operator U. The output state vector is the operator applied to the input vector; that is, $|\psi_{\text{out}}\rangle = U|\psi_{\text{in}}\rangle$. The classical NOT has the quantum analogue X which inverts the probabilities of measuring 0 and 1. The quantum analogue of XOR is the two-qubit CNOT gate: the *target* qubit is inverted for those states where the *source* qubit is 1. Most quantum gates, however, have no classical analogue. The Z gate flips the relative phase of the $|1\rangle$ state, thus exchanging $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$. The Hadamard gate H turns $|0\rangle$ into $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ and $|1\rangle$ into $\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$; it can be thought of as performing a radix-2 Fourier transform. Another important single-qubit gate, T, leaves $|0\rangle$ unchanged but multiplies $|1\rangle$ by $\sqrt{i}$. Single qubit gates are characterized by a rotation around an axis: X rotates the qubit by π around the $\hat{x}$-axis; Z rotates by π around the $\hat{z}$-axis; and T rotates by $\pi/4$ around the $\hat{z}$ axis. By composing the T and H gates, any single-qubit gate can be approximated to arbitrary precision. The combination of T, H, and CNOT provide a *universal set*: just as any Boolean circuit can be composed from AND, OR, and NOT gates, any polynomially describable multi-qubit quantum transform U can be efficiently approximated by composing just these three quantum gates into a circuit.

One additional important operator is the SWAP gate. Just as two classical values can be swapped using three XOR's, a quantum SWAP can be implemented as three CNOTs. However, SWAP is often available natively for a given technology, which is valuable, given its importance to quantum communication.

Figure 2 shows a *quantum circuit* for teleportation (described in the next section). In quantum circuits, time goes from left to right, where single lines represent qubits, and double lines represent classical bits. A meter represents measurement. By convention, black dots represent control terminals for quantum-controlled gates. The symbol $\oplus$ is shorthand for the target qubit of the CNOT gate.

2.3 Quantum Teleportation

Quantum teleportation is the re-creation of a quantum state at a distance, using only classical communication. It accomplishes this feat by using a pair of entangled qubits, $|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$, called an EPR pair[2].

Figure 2 gives an overview of the teleportation process. We start by generating an EPR pair. We separate the pair, keeping one qubit, $|b\rangle$, at the source and transporting the other, $|c\rangle$, to the destination. When we want to send a qubit, $|a\rangle$, we first interact $|a\rangle$ with $|b\rangle$ using a CNOT gate. We then measure the phase and the amplitude of $|a\rangle$, send the two one-bit classical results to the destination, and use those results to re-create the correct phase and amplitude in $|c\rangle$ such that it takes on the original state of $|a\rangle$. The re-creation of phase and amplitude is done with X and Z gates, whose application is contingent on the outcome of the measurements of $|a\rangle$ and $|b\rangle$. Intuitively, since $|c\rangle$ has a special relationship with $|b\rangle$, interacting $|a\rangle$ with $|b\rangle$ makes $|c\rangle$ resemble $|a\rangle$, modulo a phase and/or amplitude error. The two measurements allow us to correct these errors and re-create $|a\rangle$ at the destination. Note that the original state of $|a\rangle$ is destroyed when we take our two measurements[3].

Why bother with teleportation when we end up transporting $|c\rangle$ anyway? Why not just transport $|a\rangle$ directly? First, we can precommunicate EPR pairs with extensive pipelining without stalling computations. Second, it is easier to transport EPR pairs than real data. Since $|b\rangle$ and $|c\rangle$ have known properties, we can employ a specialized procedure known as *purification* to turn a collection of pairs partially damaged from transport into a smaller collection of asymptotically perfect pairs. Third, transmitting the two classical bits resulting from the measurements is more reliable than transmitting quantum data.

3. FAULT-TOLERANT COMPUTATION

We turn now to an outline of the basic constructions of fault-tolerant quantum computation. This is a rather involved subject (for which the reader is referred to the literature [21, 10]), but three essential ideas are covered here. The main result we build upon is the following: *A quantum circuit containing N error-free gates can be simulated with a probability of failure of at most ε using $O(\text{poly}(\log(N/\varepsilon))N)$ imperfect gates which fail with probability p as long as $p < p_{th}$, where p_{th} is a constant threshold that is independent of N.* This remarkable result, the *Threshold Theorem* [1], is achieved by three steps: (1) using quantum error-correction codes (Section 3.1), (2) performing all computations on encoded data, using *fault tolerant procedures* (Section 3.2), and (3) recursively encoding until the desired reliability is obtained (Section 3.3). All of these results are from prior literature [1, 28, 31, 21, 10], but we describe them here to make our contributions clearer in future sections.

3.1 Quantum Error Correction

The only error which can occur to a classical bit is a bit-flip, which can be modeled as a random NOT gate. Quantum bits suffer more kinds of error, because of the greater degree of freedom in their state representation; surprisingly, however, there are general strategies for reducing the universe of possible quantum errors to only two kinds: bit-flips (random X gates), and phase-flips (random Z gates). Classical error correction codes only take into account bit flip errors, and thus are insufficient for correcting quantum data; furthermore, quantum states collapse upon measurement, so strategies must be employed for determining errors without actually measuring encoded data.

[2] An EPR or Einstein-Podolsky-Rosen pair is a special instance of entanglement noted in the Einstein-Podolsky-Rosen paradox [2].

[3] This is consistent with the *no-cloning* theorem, which states that a quantum state cannot be copied.

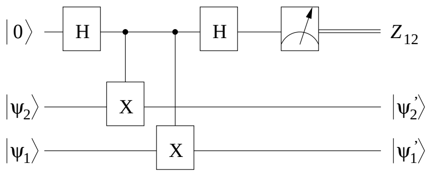


Figure 3: Quantum circuit for measuring Z_{12}, the phase difference between ψ_2 and ψ_1. The meter box indicates measurement and double lines indicate classical information.

Z_{12}	Z_{23}	Error Type	Action
0	0	no error	no action
0	1	qubit 3 flipped	flip qubit 3
1	0	qubit 1 flipped	flip qubit 1
1	1	qubit 2 flipped	flip qubit 2

Table 1: Phase correction for a 3-qubit code

Classical error correction relies upon distributing k bits of information across n bits, $n > k$, and ensuring enough redundancy to recreate the original information. Because of the no-cloning theorem, quantum information cannot be simply duplicated. Instead, redundancy is achieved through entangled states with known properties. For example, a single logical qubit, $c_0|0_L\rangle + c_1|1_L\rangle$ can be represented using three physical qubits, as the state $c_0|000\rangle + c_1|111\rangle$. A bit flip error on the first (left-most) qubit would turn this into $c_0|100\rangle + c_1|011\rangle$; this error can be detected by computing the *parity* of each pair of qubits, and leaving the result in an extra qubit called an *ancilla*. The three parities give the *error syndrome*, uniquely locating any single bit-flip error. Crucially, this strategy reveals nothing about the coefficients c_0 and c_1, since the parities cannot distinguish between $|000\rangle$ and $|111\rangle$ or any single bit-flip version of the two three-qubit strings. By measuring parities, errors can be detected without collapsing encoded data.

Correcting phase flips is achieved by measuring differences in phase, using a circuit like the one in Figure 3. This works by using a Hadamard gate to transform phase flips into bit flips; parities are then measured as before, the results stored in ancilla qubits, and then the qubits are transformed back into their original basis. Figure 4 shows how a phase error syndrome can be computed and a corresponding correction procedure applied to correct the error, following the specification of Table 1.

A quantum code which encodes one qubit and allows any single bit-flip or phase-flip error to be corrected uses the encoding $c_0|0_L\rangle + c_1|1_L\rangle$, where the logical zero and one qubits are

$$|0_L\rangle = \frac{(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)}{2\sqrt{2}}$$
$$|1_L\rangle = \frac{(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle)}{2\sqrt{2}}$$

This nine qubit code, discovered by Peter Shor [28], is also known as the $[\![9,1,3]\!]$ code, in the notation $[\![n,k,d]\!]$, where n is the number of physical qubits, k is the number of logical qubits encoded, and d is the quantum Hamming distance of the code. A code with distance d is able to correct $(d-1)/2$ errors.

3.2 Computing on Encoded Data

The nine qubit code has a remarkable property that illustrates a key requirement for fault tolerance: applying a Z gate to each of the nine qubits takes $|0_L\rangle$ to $|1_L\rangle$ and vice versa. It is the same as applying a logical $\overline{X}$ operator[4] to the encoded qubit! Similarly, $\overline{Z}$ can be performed by applying an X operator to each qubit, and $\overline{H}$ by applying an H operator to each qubit.

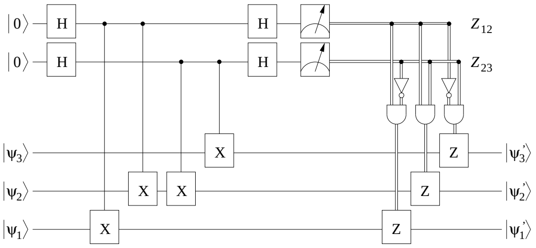


Figure 4: Syndrome Measurement for a 3-qubit Code. The classical results of measurement (double lines) control application of the Z operator.

In this paper, we employ Steane's $[\![7,1,3]\!]$ code [30], which also allows simple computation on encoded data, but requires two fewer physical qubits. In addition, a CNOT gate on two encoded qubits can be accomplished using seven CNOT gates, between each pair of corresponding physical qubits. The last remaining gate necessary to achieve the universal set from Section 2.2, the T gate, can also be performed, albeit with some extra effort [21]. Thus, universal computation is possible without requiring that the data be decoded.

Merely computing on encoded data is not sufficient, however; one additional step is required, which is frequent, periodic error correction. Because all gates used in this task are assumed to be subject to failure, this must be done in a careful manner, such that no single gate failure can lead to more than one error in each encoded qubit block. Such constructions are known as *fault tolerant procedures*, and the impact of this requirement on our study is twofold: (1) no single operation may cause multiple failures, and (2) measurement errors must not be allowed to propagate excessively. To achieve (1), no two encoding qubits are allowed to both interact directly with a third qubit. Instead, the "third" qubit is replaced with a *cat state* (a generalization of an EPR pair), $\frac{1}{\sqrt{2}}|00\ldots0\rangle + \frac{1}{\sqrt{2}}|11\ldots1\rangle$, that has itself been verified. Cat states are used because they do not transmit errors through CNOT gates. To achieve (2), measurements are performed in a multiple fashion. While it is not possible to copy a value before measuring, it is possible to form a three-qubit state, similar to the three-qubit bit-flip encoding (Section 3.1), where all of the qubits should measure to the same value; if one of the measurements differs, it is assumed to be in error. These impacts are explained in detail in later examples.

Any logical operator may be applied as a fault tolerant procedure, as long as the probability, p, of an error for a physical operator is below a certain threshold, $1/c$, where c is determined by the implementation of the error correction code. For the Steane $[\![7,1,3]\!]$ code, c is about 10^4. The overall probability of error for the logical operator is cp^2. That is, at some step in the application of the operator, and subsequent error correction, two errors would have to occur in order for the logical operator to fail.

3.3 Recursive Error Correction

A very simple construction allows us to tolerate additional errors. If a logical qubit is encoded in a block of n qubits, it is possible to encode each of those n qubits with an m-qubit code to produce an mn encoding. Such recursion, or *concatenation*, of codes can re-

[4] The overscore denotes an operator on a logical qubit: a logical operator.

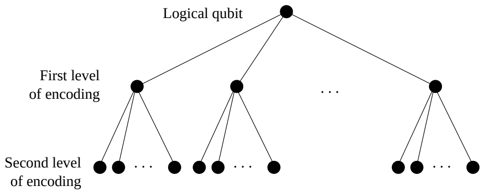


Figure 5: Tree structure of concatenated codes

duce the overall probability of error even further. For example, concatenating the $[\![7,1,3]\!]$ with itself gives a $[[49,1,7]]$ code with an overall probability of error of $c(cp^2)^2$ (see Figure 5). Concatenating it $k-1$ times gives $(cp)^{2^k}/c$, while the size of the circuit increases by d^k and the time complexity increases by t^k, where d is the increase in circuit complexity for a single encoding, and t is the increase in operation time for a single encoding. For a circuit of size $p(n)$, to achieve a desired probability of success of $1-\varepsilon$, k must be chosen such that [21]:

$$\frac{(cp)^{2^k}}{c} \leq \frac{\varepsilon}{p(n)}$$

The number of operators required to achieve this result is

$$O(\text{poly}(\log p(n)/\varepsilon)p(n))\,.$$

4. TECHNOLOGY MODEL

With some basics of quantum operations in mind, we turn our attention to the technologies available to implement these operations. Experimentalists have examined several technologies for quantum computation, including trapped ions [19], photons [32], bulk spin NMR [34], Josephson junctions [20, 36], electron spin resonance transistors [38], and phosphorus nuclei in silicon (the "Kane" model) [16] [29]. The last three of these proposals, which are built on a solid-state silicon substrate, share the following key aspects:

1 Qubits are laid out in silicon in a 2-D fashion, similar to traditional CMOS VLSI.

2 Quantum interactions are near-neighbor between qubits.

3 Qubits are stored at fixed locations, but quantum data may be swapped between nearest neighbors.

4 The control structures necessary to manipulate the bits prevent a dense 2-D grid of bits. Instead, we have linear structures of bits that can cross, but that have a minimum distance between such intersections [22]. This restriction is similar to a "design rule" in traditional CMOS VLSI.

These four assumptions apply to several solid-state technologies, but for concreteness, we will focus upon an updated version of Kane's phosphorus-in-silicon nuclear-spin proposal [29]. This scheme will serve as an example for the remainder of the paper, although we will generalize our results when appropriate.

Figure 6 illustrates the Kane scheme. Quantum states are stored in relatively stable electron-donor (e^-–${}^{31}\mathrm{P}^+$) spin pairs, where the electron (e) and the phosphorous donor nucleus (n) have opposite spins. The basis states, $|0\rangle$ and $|1\rangle$ are defined as the phase difference $|0\rangle \equiv |\uparrow_e\downarrow_n\rangle + |\downarrow_e\uparrow_n\rangle$ and $|1\rangle \equiv |\uparrow_e\downarrow_n\rangle - |\downarrow_e\uparrow_n\rangle$, respectively. Twenty nanometers above the phosphorus atoms lie three classical gates, one A gate and two S gates. Precisely timed pulses on these gates provide arbitrary one- and two-qubit quantum gates.

Single qubit operators are composed of pulses on the A-gates, modulating the hyperfine interaction between the electron and nucleus to provide rotations around the $\hat{z}$-axis. A globally applied, static magnetic field provides rotations around the $\hat{x}$-axis. By changing the pulse widths, any desired rotational operator may be applied. including the identity operator[5]. Two-qubit interactions are mediated by S-gates, which move an electron from one nucleus to the next. Exact details of the pulses and quantum mechanics of this technique are beyond the scope of this paper and are described in [29].

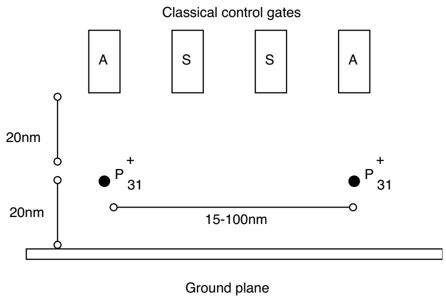


Figure 6: The basic quantum bit technology proposed by Kane. Qubits are embodied by the nuclear spin of a phosphorus atom coupled with an electron embedded in silicon under high magnetic field at low temperature.

The Kane proposal, like all quantum computing proposals, uses classical signals to control the timing and sequence of operations. All known quantum algorithms, including basic error correction for quantum data, require the determinism and reliability of classical control. Without efficient classical control, fundamental results demonstrating the feasibility of quantum computation do not apply (such as the Threshold Theorem used in Section 3).

The scale required by the Kane model, on the other hand, is at odds with efficient classical control. In order to provide the fine-grained control necessary, the control lines need to operate in a classical manner. That is, there need to be enough quantum states in the control lines so that electron movement is bulk, not ballistic, and voltage transitions are smooth rather than stair-stepped. Because of this, the control lines need to be physically much larger than the qubits they are controlling [22]. Conceptually, the control lines need to be of *classical* size and pitch, and packed closely to control quantum bits placed on a *quantum* scale. This imposes a constraint that qubits be laid out in straight lines, with a certain minimum number of qubits between junctions.

Given the constraint of linearity with infrequent junctions, there are several ways to lay out physical and logical qubits. Optimally, qubits should be arranged to minimize communication overhead.

In a fault tolerant design, the main activity of a quantum computer is error correction. To minimize communication costs, qubits in an encoding block should be in close proximity. Assuming that the distance between junctions is greater than the number of qubits in an encoding, the closest the qubits can be is in a straight line. But in order to avoid interacting two qubits in an encoding with a third, a two-rail approach is used–one rail for data qubits, and one for communication.

A concatenated code requires a slightly different layout (see Figure 7). Error correction is still the important operation, but the logical qubits at all but the bottom level of the code are more complicated. For the second level, the qubits are themselves simple encodings, laid out using the two-rail construction. However, to minimize communication costs, we want these logical qubits in as close proximity to each other as possible, just like the bottom level.

[5] One impact of the external magnetic field is the state of the qubit is in constant flux. The identity operator must be applied on every "cycle" in order to keep the current state.

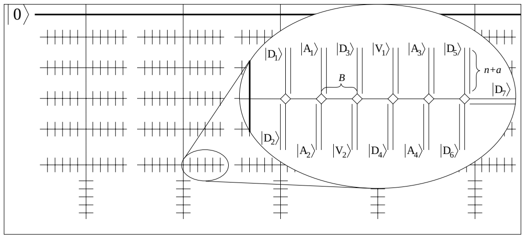


Figure 7: Schematic layout of the H-tree structure of a concatenated code. The branches in the inset represent the singly-encoded qubits. The $|D_i\rangle$ are data qubits, and the $|A_i\rangle$ are ancillae. The $|V_i\rangle$ are for verification.

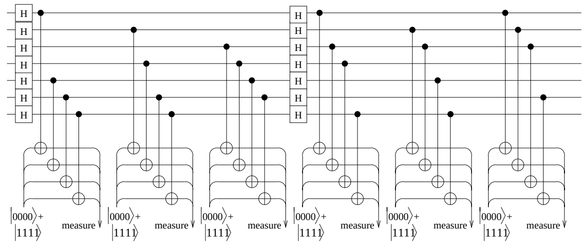


Figure 8: Measuring the error syndrome for the $[\![7,1,3]\!]$ error-correction code.

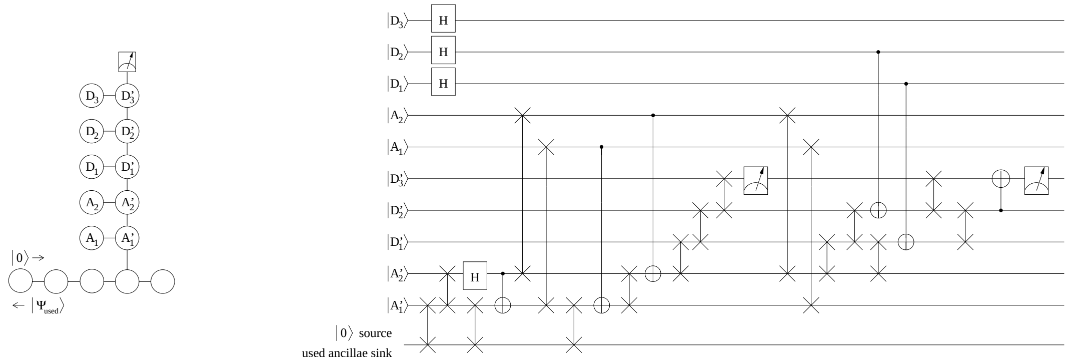


Figure 9: "Two-rail" layout for the three-qubit phase-correction code. The schematic on the left shows qubit placement and communication, where D's indicate data qubits, A's are ancillae, and D''s and A''s are for communication. The open qubits at the bottom swap in fresh ancillae, and remove used ancillae. The same layout is shown as a quantum circuit on the right, with the operations required to create and verify an ancillary cat state, and to measure the parity of a pair of data qubits.

Hence, we need to arrange the bottom level as branches coming off of a main bus. Similarly, the third level would have second-level branches coming off of a main trunk, and so on for higher levels, forming an H-tree.

5. ERROR CORRECTION ALGORITHMS

We've discussed error correction in a general sense, and how the need for recursive error correction influences the architectural design. In addition, we have introduced several error-correction codes, such as Shor's 3-qubit phase-flip code, Shor's 9-qubit code, and Steane's 7-qubit code. The constructions in Figures 4 and 9 deal with the simplest of these codes, the 3-qubit code, which only corrects phase flips. In order to correct both bit and phase flips, a more complicated code is needed. For the remainder of this paper, we will focus on the 7-qubit code, $[\![7,1,3]\!]$, which corrects up to a single error, and recursive codes based on $[\![7,1,3]\!]$ which can correct many errors. We choose $[\![7,1,3]\!]$ because of the ease with which logical operators may be applied. In particular, remember that the logical operators $\overline{X}$, $\overline{Z}$, $\overline{H}$, and $\overline{\text{CNOT}}$ are applied by applying the simple operator to each qubit in the encoding block.

5.1 The $[\![7,1,3]\!]$ Code

Error correcting using the $[\![7,1,3]\!]$ code consists of measuring the parity of the encoding qubits in various bases. As shown in Figure 8, the qubits are rotated to the measurement basis with Hadamard gates. Parity is then measured in much the same way it is on a classical code, using two-qubit CNOT operators acting as XOR's. Conceptually, the parity can be measured in the same way as the three-qubit code in Section 3.1, gathering the parity on ancilla $|0\rangle$'s. To perform a fault tolerant measurement, however, a cat state is used in place of a $|0\rangle$. Figure 8 shows a schematic for measuring the $[\![7,1,3]\!]$ code. Not shown are cat-state creation and cat-state verification. In addition, each parity measurement must be performed twice to reduce the probability of an error from $O(p)$ to $O(p^2)$; if the measurements disagree, the parity must be measured a third time!

A parity measurement consists of the following:

1 Prepare a cat state from four ancillae, using a Hadamard gate and three CNOT gates.
2 Verify the cat state by taking the parity of each pair of qubits. If any pair has odd parity, return to step 1. (Note that this requires six additional ancillae, one for each pair.)
3 Use the four-ancillae cat state as the CNOT target of the data qubits whose parity is to be measured.
4 Deconstruct the cat state by selecting one of the ancillae, $|A_0\rangle$, and using it as the CNOT target of the remaining three ancillae. $|A_0\rangle$ now has the overall parity of the cat state.
5 Measure this $|A\rangle_0$:
   A With $|A_0\rangle = \alpha|0\rangle + \beta|1\rangle$, create the three-qubit state, $\alpha|000\rangle + \beta|111\rangle$ by using $|A_0\rangle$ as the control for two CNOT gates, and two fresh $|0\rangle$ ancillae as the targets.
   B Measure each of the three qubits.
6 Use the majority measured value as the parity of the cat state.

The resulting syndrome determines which, if any, qubit has an error, and which X, Z, or Y operator will correct the error.

For the Steane $[\![7,1,3]\!]$ code, each parity measurement requires twelve ancillae–four for the cat state to capture the parity, six to verify the cat state, and two additional qubits to measure the cat state. The six parity measurements are each performed at least twice, for a minimum of 144 ancillae to measure the error syndrome! A less complex example is shown in Figure 9.

Each of the twelve parity measurements require:

- One Hadamard and three CNOT's to create the cat state;
- Twelve CNOT's to verify the cat state;
- Four CNOT's, which can be applied in parallel, to collect the parity of the data qubits;
- Three CNOT's and a Hadamard to uncreate the cat state;
- Two CNOT's to create the three-qubit state for measurement; and
- Three qubit measurements, which may be performed in parallel with the next parity measurement.

If the time required to apply a single-qubit operator is S, a CNOT is C, and a measurement is M, then the minimum time required to measure the error syndrome is $2S+12(2S+24C)$.

5.2 Concatenated Codes

The $[\![7,1,3]\!]\times[\![7,1,3]\!]$ two-level concatenated code is measured in the same way as the $[\![7,1,3]\!]$ code, except the qubits and ancillae are encoded. For example, each logical ancilla must be prepared in the following manner[6]

1 Begin with seven ancillae.
2 Measure the error syndrome, and correct, as in Section 5.1. At this point, the seven qubits constitute a valid code word.
3 Measure the value of the logical ancilla:
  A Create a cat state with another seven ancillae to collect the parity of the seven qubits in the logical ancilla.
  B Verify the cat state.
  C Use the cat-state qubits as the CNOT target of the qubits encoding the logical ancilla.
  D Uncreate the cat-state, collecting the parity into a single qubit.
  E With two fresh ancillae, create $\alpha|000\rangle+\beta|111\rangle$
  F Measure each of these three qubits.
4 Use the majority measured value as the value of the logical ancilla.
5 If the measurement is $|1_L\rangle$, apply $\overline{X}$.

The error syndrome measurement is analogous to the singly-encoded $[\![7,1,3]\!]$ case, except that the lower-level encodings must be error corrected between operations:

1 Prepare four logical ancillae in a cat state.
2 Error correct the four ancilla.
3 Verify the cat state.
4 Use the ancillae as the $\overline{\text{CNOT}}$ target of the qubits whose parity is to be measured.
5 Error correct the four qubits in the cat state and the logical data qubits.
6 Measure each of the four logical cat-state qubits. The parity of these measurements is the parity of the four encoding qubits. This step is equivalent to the cat-state deconstruction step for the singly-encoded case.

As in the singly-encoded case, each parity measurement must be performed at least twice. The resulting syndrome determines which, if any, logical qubit has an error. The appropriate $\overline{X}$, $\overline{Z}$, or $\overline{Y}$ operator can be applied to correct the error. Of course, after the operator is applied to a logical qubit, that qubit must be error-corrected.

Higher levels are error-corrected analogously.

6. COMMUNICATION COSTS

In this section, we derive the primary results of this paper. First, we model the communication costs of our error correction algorithms under the near neighbor constraint. We show that there are too many SWAP operations between upper levels of our tree structures and that too much error accumulates to be corrected. Second, we analyze quantum teleportation as an alternative to SWAP operations for long-distance communication. Finally, we show that teleportation is necessary both in terms of distance and in terms of the accumulating probability of correlated errors between redundant qubits in our code words.

6.1 Error Correction Costs

The error correction algorithms in the previous section are in an ideal situation, where any qubit can interact with any other qubit. Usually, qubits can only interact with their near neighbors, so before applying a two-qubit operator, one of the operand qubits must be moved adjacent to the other.

One of the easiest ways to move quantum data is to use the SWAP operator. By applying SWAP's between alternating pairs of qubits, the values of alternating qubits are propagated in one direction, while the remaining qubit values are propagated in the reverse direction. This can be used to supply $|0\rangle$ ancillae for the purpose of error correction. As a side benefit, this also removes "used" ancillae. Figure 9 illustrates this method for the three-qubit example, using two rows of qubits, one for the encoding data qubits and one for the ancillae.

The same method can be applied to the $[\![7,1,3]\!]$ code. The actual communication costs depend on the physical implementation used. The time required for an error correction parity check is

$$t_{ecc}=12(t_{cc}+t_{cv}+t_p+t_{cd}+t_m) \qquad (1)$$

where

t_{cc} is the time for cat state creation;
t_{cv} is the time for cat state verification;
t_p is the time to entangle the cat state with the parity qubits;
t_{cd} is the time to uncreate the cat state; and
t_m is the time to perform a fault-tolerant measurement.

For $[\![7,1,3]\!]$ in the ideal, sea-of-qubits model, t_{cc} is $t_{single}+3t_{cnot}$, t_{cv} is $6(2t_{cnot}+t_{meas})$, t_p is t_{cnot}–four CNOT's performed in parallel, t_{cd} is $3t_{cnot}+t_{single}$ and $t_{overlap}$ is $t_{decat}+t_{meas}$, where

t_{single} is the time required for a single-qubit operator;
t_{cnot} is the time required for a CNOT operator;
t_{swap} is the time required for a SWAP operator; and
t_{meas} is the time required for the measurement operator.

If communication by swapping is used,

$$t_{cc}=\max(t_{single},t_{swap})+4t_{swap}+3\max(t_{cnot},t_{swap}), \qquad (2)$$
$$t_{cv}=\max(t_{single},t_{swap})+18t_{swap}+12\max(t_{cnot},t_{swap}), \qquad (3)$$
$$t_p\leq 4\max(t_{cnot},t_{swap}),\text{ and} \qquad (4)$$
$$t_{cd}\leq 3t_{swap}+2\max(t_{cnot},t_{swap})+t_{single}. \qquad (5)$$

In the Kane model, $t_{single}<t_{swap}<t_{cnot}$, so the overall cost is

$$t_{ecc}\leq 336t_{swap}+168t_{cnot}+t_{meas}.$$

Since measurement is fully parallelizable, these times assume that there are enough measurement functional units to perform measurement in parallel with the other operations in the error-correction cycle.

6.2 Multilevel Error Correction

For the concatenated code, the data movement in the upper levels is more complicated. Although Eq. 1 still holds, each parity

[6] Fault-tolerant algorithms that avoid the overhead of encoded ancilla are a topic of future research.

k	SWAP	CNOT	1-Qubit	Measurement
1	1620	288	12	108
2	$690,000$	$120,000$	4800	$43,000$
3	2.7×10^{08}	4.7×10^{07}	1.9×10^{06}	1.7×10^{07}
4	1.1×10^{11}	1.8×10^{10}	7.5×10^{08}	6.7×10^{09}
5	4.3×10^{13}	7.3×10^{12}	3.0×10^{11}	2.7×10^{12}

Table 2: Operations required for an error-correction cycle at level *k*.

measurement requires the following. Since ancillae are themselves encoded, they each require their own branch, and the first step is to create the encoded ancillae, by error correcting, measuring, and if necessary, inverting. The second step is to create the four-qubit cat state from the logical ancillae, by applying $\overline{H}$ to one of the ancilla, moving a second ancillae through the main branch to the second rail of the first ancilla, applying $\overline{\text{CNOT}}$, moving the second ancilla back, and error-correcting both ancillae. This is repeated for the second and third ancillae, and the third and fourth ancillae. Since the ancillae are error corrected along the way, the cat state need not be verified.

Next, an ancillae is moved through the main branch to the data branch that holds a bottom-level encoding. After applying $\overline{\text{CNOT}}$, the ancilla is moved back to its own branch, and both it and the logical data qubit are error-corrected. Since measuring the ancillae can be performed completely in parallel, all four ancillae are measured, and the parity of the measurements is the parity of the data qubits.

For $[[7,1,3]]$ concatenated with itself k times,

$$t_{anc,k} = t_{ecc,k-1} + t_{m,k-1}; \tag{6}$$

$$t_{cc,k} = 6t_{ecc,k-1} + 6t_{b,k}; \tag{7}$$

$$t_{cv,k} = 18t_{ecc,k-1} + 28t_{b,k}; \tag{8}$$

$$t_{p,k} = 8t_{ecc,k-1} + 10t_{b,k}; \tag{9}$$

$$t_{cd,k} = 4t_{m,k-1} \tag{10}$$

$$t_{m,k} = 4^{k-1} t_{m,1}; \text{ and} \tag{11}$$

$$t_{b,k} = \begin{cases} 1, & k = 1 \\ t_{B,arch}, & k = 2 \\ (n+a)t_{b,k-2} + t_{B,arch}, & k > 2 \end{cases} \tag{12}$$

where the subscript k indicates the level of encoding, $t_{anc,k}$ is the cost of encoding an ancilla, $t_{b,k}$ is the branch distance between logical qubits at level k, $t_{m,1}$ is the time required to measure a singly-encoded qubit, $t_{B,arch}$ is the minimum number of qubits between two branches for a given architectural model, n is the number of physical qubits in the non-concatenated code and a is the number of ancillae per parity measurement. For concatenated codes, parallel operation is determined by the ratio of ancillae delivery to ancillae consumption for a singly-encoded parity check. For $[[7,1,3]]$ and a single-qubit-wide branch this ratio is around 3. Arranging the ancillae as in the inset of Figure 7 minimizes the distance that ancillae must travel.

The recurrence relation given in Eqs. 6 through 12 give an overall time to perform an error-correction cycle at a given level of recursion. A similar recurrence relation gives the total number of operations required. The number of operators required for different levels of encoding are summarized in Table 2, which shows that the SWAP operator is very important in a realistic model, compared to the sea-of-qubits model, where SWAP's are not required. In this realistic model, SWAP's account for over 80% of all operations.

k	Teleportation	Swapping, $t_{B,arch} = 22$	Swapping, $t_{B,arch} = 61$	Swapping, $t_{B,arch} = 285$
1	864	1	1	1
2	864	22	61	285
3	864	77	194	866
4	864	330	876	4,012
5	864	913	2,317	10,381
6	864	3,696	9,819	44,987

Table 3: Comparison of the cost of swapping an encoded qubit to the cost of teleporting it. The "swapping" values are b_k, the distance between adjacent qubits.

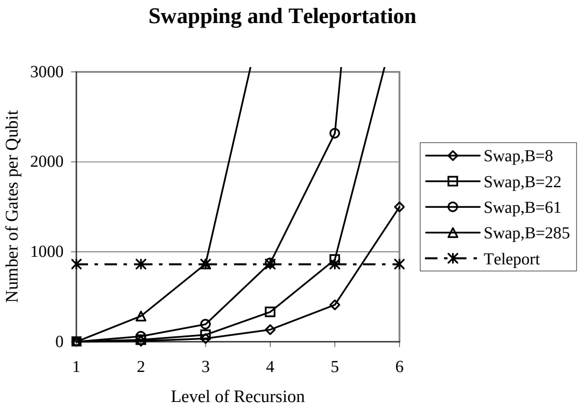


Figure 10: Cost of teleportation compared to swapping. The values chosen illustrate the break-even point for different levels of recursion.

6.3 Teleportation

Fortunately, we can use quantum teleportation as an alternative to swapping for communication over long distances. To use teleportation for our circuit, we must evaluate the number of computation and communication operations within the teleportation circuit. By comparing this number of operations with the swapping costs from the previous section, we can decide at what level k of the tree to start using teleportation instead of swapping for communication.

6.4 Distance Tradeoff

By calculating the number of basic computation and communication operations necessary to use teleportation for long-distance communication, we can quantify when we should switch from swapping to teleportation in our tree structure. Figure 10 illustrates this tradeoff. We can see that for $t_{B,arch} = 22$, teleportation should be used when $k \geq 5$.

6.5 Avoiding Correlated Errors

An important assumption in quantum error correction is that errors in the redundant qubits of a codeword are uncorrelated. That is, we do not want one error in a codeword to make a second error more likely. To avoid such correlation, it is important to try not to interact qubits in a codeword with each other.

Unfortunately, we find that a 2D layout cannot avoid indirect interaction of qubits in a codeword. At some point, all the qubits in a codeword must be brought to the same physical location in order to calculate error syndromes. In order to do this, they must pass through the same line of physical locations. Although we can avoid swapping the codeword qubits with each other, we cannot avoid swapping them with some of the same qubits that flow in the other direction.

For concreteness, if two qubits of codeword d_0 and d_1 both swap with an ancilla a_0 going in the opposite direction, there is some probability that d_0 and d_1 will become correlated with each other through the ancilla. This occurs if both SWAPs experience a partial failure. In general, if p is the probability of a failure of a SWAP gate, the probability of an error from swapping a logical qubit is

$$n^k b_k p + \binom{n^k}{2} b_k p^2 + \binom{n^k}{3} b_k p^3 + \cdots,$$

where b_k is the number of qubits between branches at level k, and the higher order terms are due to correlation between the qubits. From this form, it is clear that correlated errors are dominated by uncorrelated errors, when $n^k p \ll 1$.

7. FUTURE WORK

Our results have interesting implications for the Threshold Theorem, effectively increasing the reliability requirements for quantum operators, particularly SWAP operators. In addition to the teleportation solution to long-distance communication, it may be possible to modify the straightforward recursive structure used in quantum error correction codes to include intermediate error correction steps in the middle of long chains of SWAP operators. There are, however, serious challenges of getting the ancillae to all of these intermediate points in such a layout.

At the lowest level, the largest consumer of ancillae for error correction is cat-state verification. However, at higher levels, the cat states themselves are constructed from logical ancillae, each of which must be error corrected, measured, and the whole cat state verified. This approach is a straightforward analog to the lowest level, but there may be more efficient algorithms from the standpoint of ancilla use.

The proposed teleportation solution assumes that the distribution of reliable EPR pairs is significantly easier than transporting arbitrary quantum data. EPR pairs are precommunicated in a pipelined fashion, then "purified" using an entanglement-concentrating algorithm that eliminates bad EPR pairs [25]. Quantifying the reliability and bandwidth of this mechanism is the subject of future study.

Finally, this paper has focused on solid-state implementations with static qubits. There is a proposal for scalable ion-trap quantum computers, built using conventional microfabrication techniques, where the qubits are mobile [17]. How the mobility constraints of such a system compare to swapping with static qubits is a subject of future study.

8. CONCLUSION

Quantum computation is in its infancy, but now is the time to evaluate quantum algorithms under realistic constraints and derive the architectural mechanisms and reliability targets that are needed in order to scale quantum computers to their full potential. This paper has focused upon the spatial and temporal constraints of solid-state technologies, and has shown that the recursive construction for quantum error correction codes requires a long-distance communication technology such as quantum teleportation. We derived the tradeoff point between short- and long-distance technologies. Also, the reliability of the quantum SWAP operation used in short-distance communication is the dominant factor in system reliability. These results are a beginning. The next step is moving quantum computation from theory to practice, unlocking an unprecedented tool to attack difficult problems.

9. ACKNOWLEDGMENTS

Thanks to John Owens, Matt Farrens, Paul Sultana, and Mark Whitney for their helpful comments on preliminary material for this paper. This work is supported in part by the DARPA Quantum Information, Science and Technology Program, by NSF CAREER grants to Mark Oskin and Fred Chong, an NSF NER grant and a UC Davis Chancellor's Fellowship to Fred Chong.

10. REFERENCES

[1] D. Aharonov and M. Ben-Or. Fault tolerant computation with constant error. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 176–188, 1997.

[2] J. S. Bell. On the Einstein-Podolsy-Rosen paradox. *Physics*, 1:195–200, 1964. Reprinted in J. S. Bell, *Speakable and Unspeakable in Quantum Mechanics*, Cambridge University Press, Cambridge, 1987.

[3] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing*, pages 175–179, 1984.

[4] D. Bouwmeester, J. W. Pan, K. Mattle, M. Eibl, H. Weinfurter, and A. Zeilinger. Experimental quantum teleportation. *Nature*, 390(6660):575–579, 1997.

[5] A. M. Childs, E. Farhi, and J. Preskill. Robustness of adiabatic quantum computation. *Phys. Rev. A*, (65), 2002.

[6] I. L. Chuang. Quantum algorithm for clock synchronization. *Phys. Rev. Lett.*, 85:2006, Aug 2000.

[7] I. L. Chuang, N. Gershenfeld, and M. Kubinec. Experimental implementation of fast quantum searching. *Phys. Rev. Lett.*, 18(15):3408–3411, 1998.

[8] I. L. Chuang, R. Laflamme, P. Shor, and W. H. Zurek. Quantum computers, factoring, and decoherence. *Science*, 270:1633, Dec 1995. arXive e-print quant-ph/9503007.

[9] N. Gershenfeld and I. Chuang. Quantum computing with molecules. *Scientific American*, June 1998.

[10] D. Gottesman. Theory of fault-tolerant quantum computation. *Phys. Rev. A*, 57(1):127–137, 1998. arXive e-print quant-ph/9702029.

[11] L. Grover. In *Proc. 28th Annual ACM Symposium on the Theory of Computation*, pages 212–219, New York, 1996. ACM Press.

[12] S. Hallgren. *Quantum Information Processing '02 Workshop*, 2002.

[13] P. Hoyer. *Banff workshop on quantum algorithms*, 2002.

[14] J. A. Jones, M. Mosca, and R. H. Hansen. Implementation of a quantum search algorithm on a nuclear magnetic resonance quantum computer. *Nature*, 393(6683):344, 1998. arXive e-print quant-ph/9805069.

[15] R. Jozsa, D. Abrams, J. Dowling, and C. Williams. Quantum atomic clock synchronization based on shared prior entanglement. *Phys. Rev. Lett.*, pages 2010–2013, August 2000.

[16] B. Kane. A silicon-based nuclear spin quantum computer. *Nature*, 393:133–137, 1998.

[17] D. Kielpinski, C. Monroe, and D. J. Wineland. Architecture for a large-scale ion-trap quantum computer. *Nature*, 417:709–711, 2002.

[18] S. Lloyd. Quantum-mechanical computers. *Scientific American*, 273(4):44, Oct. 1995.

[19] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland. Demonstration of a fundamental quantum logic gate. *Phys. Rev. Lett.*, 75:4714, 1995.

[20] Y. Nakamura, Y. A. Pashkin, and J. S. Tsai. Coherent control of macroscopic quantum states in a single-cooper-pair box. *Nature*, 398:786–788, 1999.

[21] M. Nielsen and I. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge, England, 2000.

[22] M. Oskin, F. Chong, I. Chuang, and J. Kubiatowicz. Building quantum wires: The long and the short of it. In *Proc. International Symposium on Computer Architecture (ISCA 2003)*, New York, 2003. ACM Press.

[23] J. Preskill. Reliable quantum computers. *Proc. R. Soc. London A*, 454(1969):385–410, 1998.

[24] C. Sackett, D. Kielpinsky, B. King, C. Langer, V. Meyer, C. Myatt, M. Rowe, Q. Turchette, W. Itano, D. Wineland, and C. Monroe. Experimental entanglement of four particles. *Nature*, 404:256–258, 2000.

[25] L. Schulman and U. Vazirani. Molecular scale heat engines and scalable quantum computation. In *31st STOC*, 1999.

[26] P. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. 35th Annual Symposium on Foundations of Computer Science*, page 124, Los Alamitos, CA, 1994. IEEE Press.

[27] P. Shor. Fault-tolerant quantum computation. In *37th FOCS*, 1994.

[28] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 54:2493, 1995.

[29] A. Skinner et al. Hydrogenic spin quantum computing in silicon: a digital approach. *quant-ph/0206159*, 2002.

[30] A. Steane. Error correcting codes in quantum theory. *Phys. Rev. Lett.*, 77, 1996.

[31] A. Steane. Simple quantum error correcting codes. *Phys. Rev. Lett.*, 77:793–797, 1996.

[32] Q. A. Turchette, C. J. Hood, W. Lange, H. Mabuchi, and H. J. Kimble. Measurement of conditional phase shifts for quantum logic. *Phys. Rev. Lett.*, 75:4710, 1995.

[33] W. van Dam and G. Seroussi. Efficient quantum algorithms for estimating gauss sums. *quant-ph*, page 0207131, 2002.

[34] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, R. Cleve, and I. L. Chuang. Experimental realization of order-finding with a quantum computer. *Phys. Rev. Lett.*, to appear, 2000.

[35] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414:883, 2001.

[36] D. Vion, A. Aassime, A. Cottet, P. Joyez, H. Pothier, C. Urbina, D. Esteve, and M. H. Devoret. Maniplating the quantum state of an electrical circuit. *Science*, 296:886, 2002.

[37] J. von Neuman. *Automata Studies*. Princeton University Press, Princeton, NJ, 1956.

[38] R. Vrijen, E. Yablonovitch, K. Wang, H. W. Jiang, A. Balandin, V. Roychowdhury, T. Mor, and D. DiVincenzo. Electron spin resonance transistors for quantum computing in silicon-germanium heterostructures. *arXive e-print quant-ph/9905096*, 1999.

[39] S. Winograd and J. D. Cowan. *Reliable Computation in the Presence of Noise*. MIT Press, Cambridge, Mass., 1963.