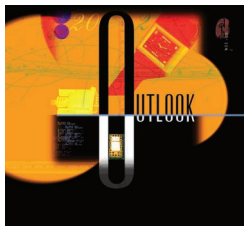
A Practical Architecture for Reliable Quantum Computers



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-

0018-9162/02/$17.00 © 2002 IEEE

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

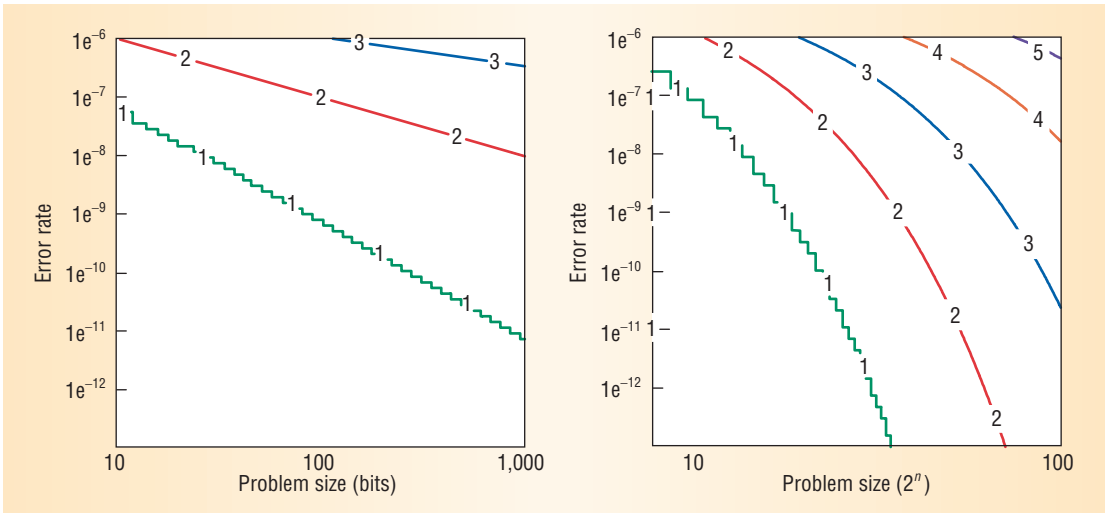


Figure 1. Recursion level k with a varied problem size and underlying qubit error probability for Shor's quantum factorization algorithm (left) and Grover's quantum search algorithm (right).

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 $\pi/4$ (S),
- rotation by $\pi/8$ (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

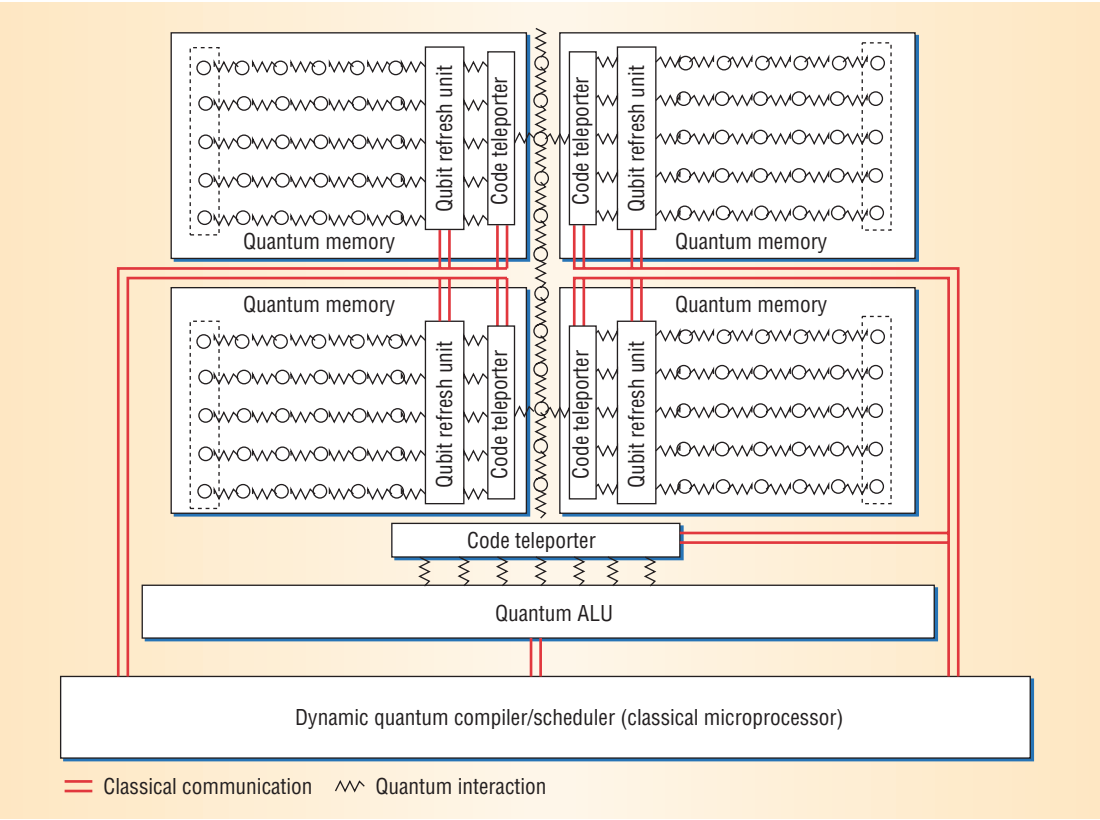


Figure 2. Fault-tolerant quantum computer architecture. The quantum arithmetic logic unit (ALU) performs all quantum operations, quantum memory banks support efficient code conversion, teleportation transmits quantum states without sending quantum data, and the dynamic scheduler controls all processes.

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 sim-

ply 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.

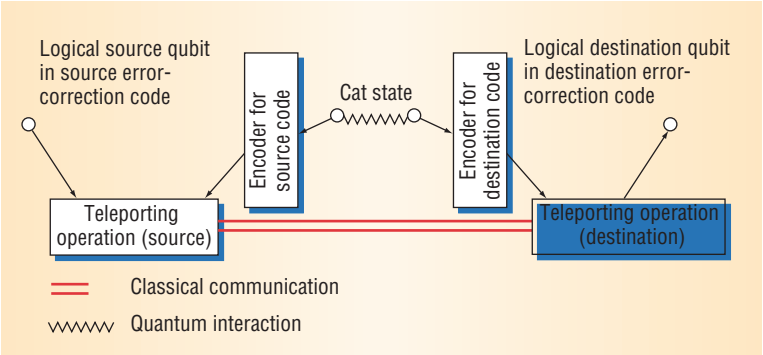


Figure 3. Code teleportation. In a slight modification of standard quantum teleportation, an encoder at the destination recreates quantum data encoded in another form at the sender.

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

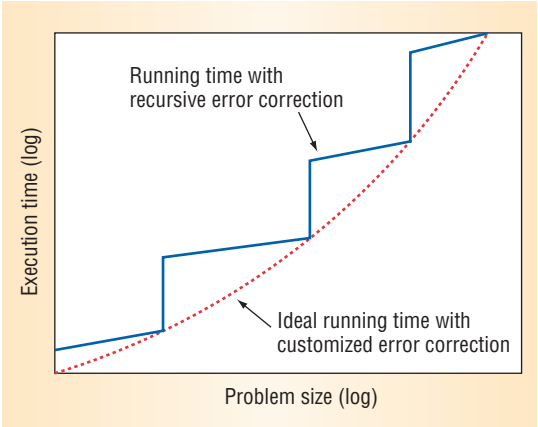


Figure 4. Quantum computing performance with recursive error correction. The recursive approach to stronger error corrections results in a stairstep curve.

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.*