

## Quantum computer architecture holds the key to building commercially viable systems.

BY RODNEY VAN METER AND CLARE HORSMAN

# A Blueprint for Building a Quantum Computer

SMALL-SCALE QUANTUM COMPUTING devices built on a variety of underlying physical implementations exist in the laboratory, where they have been evolving for over a decade, and have demonstrated the fundamental characteristics necessary for building systems. The challenge lies in extending these systems to be large enough, fast enough, and accurate enough to solve problems that are intractable for classical systems, such as the factoring of large numbers and the exact simulation of other quantum mechanical systems. The architecture of such a computer will be key to its performance. Structurally, when built, a “quantum computer” will in fact be a hybrid device, with quantum computing units serving as coprocessors to classical systems. The program, much control circuitry, and substantial pre- and post-processing functions will reside on the classical side of the system. The organization of the quantum

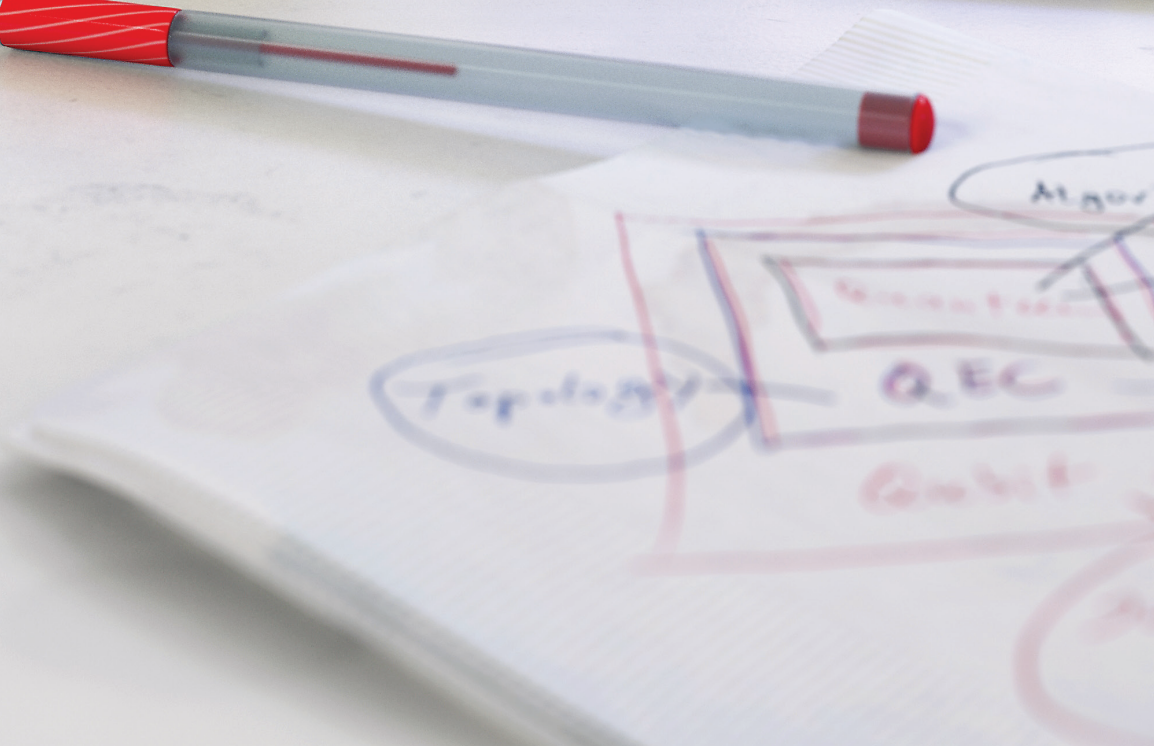
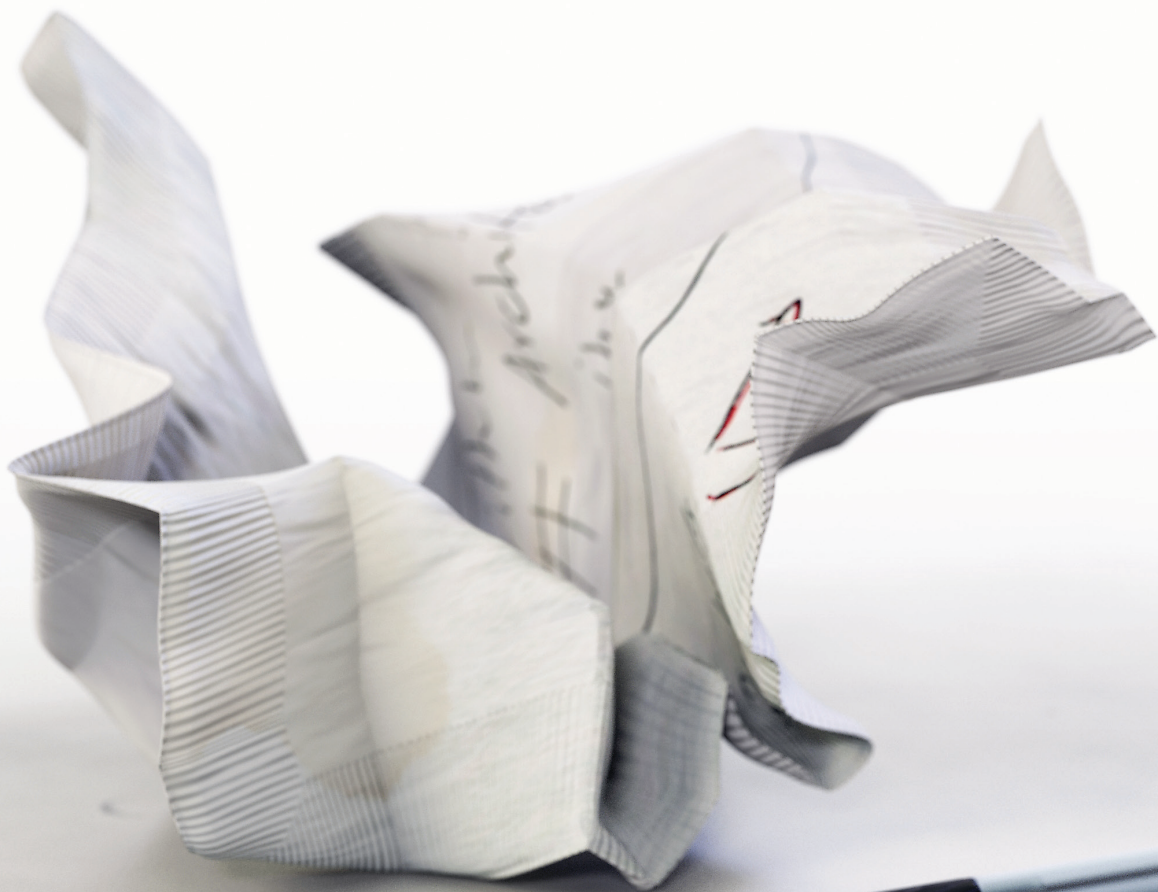
system itself, the algorithmic workloads for which it is designed, its speed and capabilities in meeting those goals, its interfaces to the classical control logic, and the design of the classical control systems are all the responsibility of quantum computer architects.

In this article, we review the progress that has been made in developing architectures for full-scale quantum computers. We highlight the process of integrating the basic elements that have already been developed, and introduce the challenges that remain in delivering on the promise of quantum computing.

The most famous development to date in quantum algorithms is Shor’s algorithm for factoring large numbers in polynomial time.<sup>33</sup> While the vernacular press often talks of factoring large numbers “in seconds” using a quantum computer, in reality it is not even possible to discuss the prospective performance of a system without knowing the physical and logical clock speed, the topology of the interconnect among the elements, the number of logical quantum bits (qubits) available in the system, and the details of the algorithmic implementation—in short, without specifying the architecture. Figure 1 illustrates the impact that architecture can have on the bottom-line viability of a quantum computer; here,

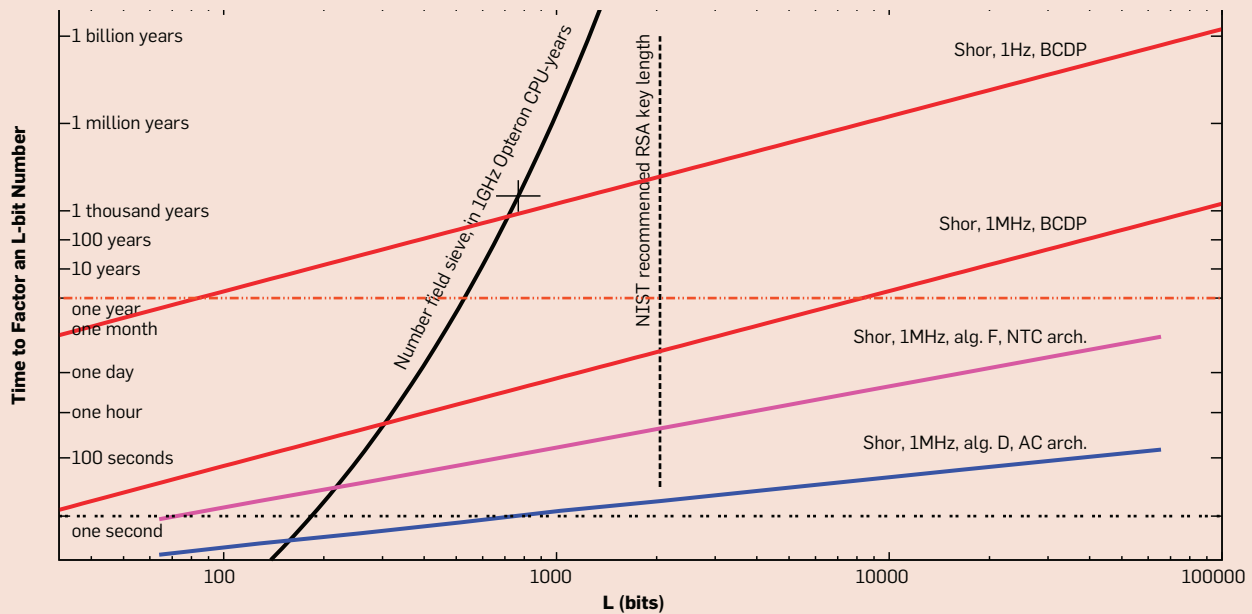
### » key insights

- General-purpose quantum computers capable of efficiently solving difficult problems will be physically large, comprising millions or possibly billions of quantum bits in distributed systems.
- Quantum computer architecture matches available physical qubit technologies to applications.
- The driving force of an architecture is quantum error correction, guarding against loss of fragile quantum data.
- Even with quantum computers, constant factors matter as much as asymptotic computational class; researchers worldwide are working to bring error-corrected clock speeds up to operationally attractive levels.



**Figure 1. Scaling the classical number field sieve (NFS) vs. Shor's quantum algorithm for factoring.<sup>37</sup>**

The horizontal axis is the length of the number to be factored. The steep curve is NFS, with the marked point at  $L = 768$  requiring 3,300 CPU-years. The vertical line at  $L = 2048$  is NIST's 2007 recommendation for RSA key length for data intended to remain secure until 2030. The other lines are various combinations of quantum computer logical clock speed for a three-qubit operation known as a Toffoli gate (1Hz and 1MHz), method of implementing the arithmetic portion of Shor's algorithm (BCDP, D, and F), and quantum computer architecture (NTC and AC, with the primary difference being whether or not long-distance operations are supported). The assumed capacity of a machine in this graph is  $2L^2$  logical qubits. This figure illustrates the difficulty of making pronouncements about the speed of quantum computers.



the architecture used can make the difference between an interesting proof-of-concept device and an immediate threat to all RSA encryption.

In developing a quantum computer architecture we have much to learn from classical computer architecture, but with a few important caveats. Foremost among these caveats is that the delicate nature of quantum information demands that memory elements be very active. Second, long wires or long-distance connections inside a quantum computer are either nonexistent, requiring nearest neighbor, cellular automaton-like transfer of data, or are at best poor quality, requiring much effort to transfer even a single qubit from place to place using quantum teleportation and error management techniques. Thus, the principles of classical computer architecture can be applied, but the answers arrived at likely will differ substantially from classical architectures.

Quantum computer architecture as a field remains in its infancy, but carries much promise for producing machines that vastly exceed cur-

rent classical capabilities, for certain systems designed to solve certain problems. As we begin to design larger quantum computers, it must be recognized that large systems are not simply larger versions of small systems. The conceptual stack of subfields that must all contribute to a scalable, real-world machine is shown in Figure 2, divided into a set of layers. In this article, we discuss the elements of this structure in turn, all leading to the central core of quantum computer architecture.

At the bottom of the stack we have the technologies for storing individual qubits, and processing or transporting them to take part in a larger computation. How small groups of qubits will interconnect is the first problem quantum computer architecture must solve. Given the fragility of quantum data, how can many qubits be kept "alive" long enough to complete a complex computation? The solution to this problem, the field of *quantum error correction* (QEC), began with studies demonstrating that arbitrarily accurate computation is theoretically

possible even with imperfect systems, but our concern here is the design of subsystems for executing QEC, which can be called the quantum computer microarchitecture.<sup>26</sup> Recent progress in experimentally demonstrated building blocks and the implementation of QEC are the first two topics addressed in this article.

At the top of the stack, machines will be designed for specific workloads, to run certain algorithms (such as factoring or simulation) that exist in computational complexity classes believed to be inaccessible to classical computers. Without these algorithms, there will be no economic incentive to build and deploy machines.

With context established at both the top and bottom of the stack, we present progress that has been made toward integrated architectures, and finish with a detailed example of the immense scale-up in size and slowdown in speed arising from the error correction needs of a full-scale, digital quantum computer. We note that the process of writing this article has been made substantially easier by the



appearance in the last few years of excellent reviews on many architecture-relevant subfields.<sup>1,3,4,9,13,19,28</sup>

### Qubit Technologies

At the lowest level of our Figure 2, we have the technological building blocks for the quantum computer. The first significant attempt to characterize the technology needed to build a computer came in the mid-1990s, when DiVincenzo listed criteria that a viable quantum computing technology must have: (1) a two-level physical system to function as a qubit;<sup>a</sup> (2) a means to initialize the qubits into a known state; (3) a universal set of gates between qubits; (4) measurement; and (5) long memory lifetime.<sup>10</sup> These criteria were later augmented with two communication criteria, the ability to convert between stationary and “flying” qubits, and the ability to transmit the latter between two locations.

In any qubit technology, the first criterion is the most vital: What is the state variable? Equivalent to the electrical charge in a classical computer, what aspect of the physical system encodes the basic “0” or “1” state? The initialization, gates, and measurement process then follow from this basic step. Many groups worldwide are currently working with a range of state variables, from the direction of

quantum spin of an electron, atom, nucleus, or quantum dot, through the magnetic flux of a micron-scale current loop, to the state of a photon or photon group (its polarization, position or timing).

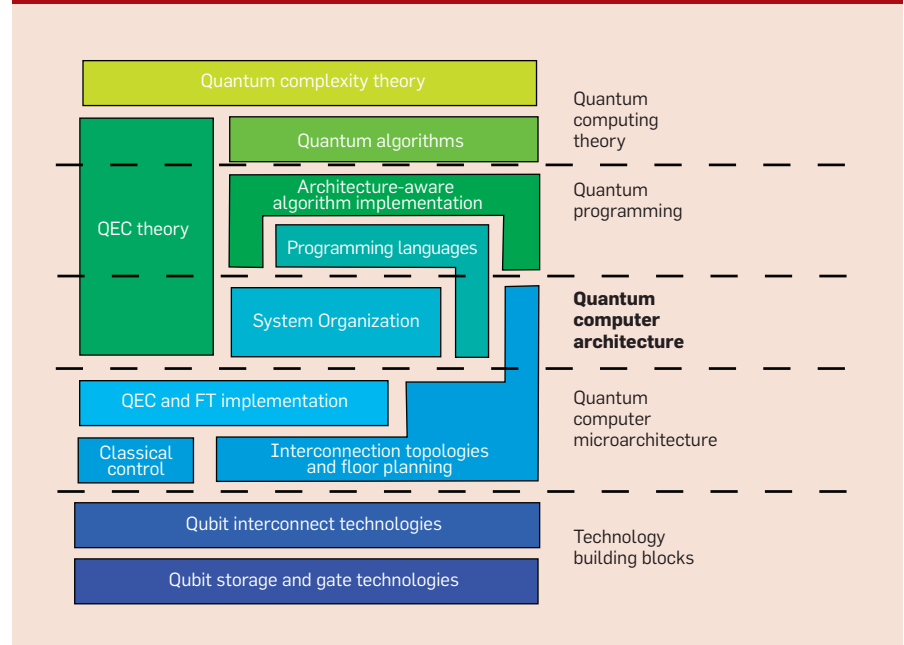
The accompanying table lists a selection of qubit technologies that have been demonstrated in the laboratory, with examples of the material and the final device given for each state variable.

Controlling any kind of physical system all the way down to the quantum level is difficult, interacting qubits with each other but not anything else is even harder, and control of systems

for quantum computing over useful lifetimes is an immense experimental challenge. While experimental progress has been impressive in the last decade, moving away from one- and two-qubit demonstrations, we are still a very long way away from entangling, storing, and manipulating qubits on anything like the scale of classical computing and bits. Here, we are able to discuss only selected examples of the various technologies on offer; for more information, we recommend the recent survey by Ladd et al.<sup>19</sup>

Ion trap devices and optical systems currently lead in the number of qubits that can be held in a device,

**Figure 2. Quantum computer architecture among some sub-fields of quantum computation. QEC is quantum error correction; FT is fault tolerant.**



a DiVincenzo's original criteria was written with qubits, on which we concentrate in this article, but tri-level qutrits or higher-dimension qudit data elements are also possible.

**A few of the many types of qubit technologies available. Different technologies rely on the same or different state variables to hold quantum data, implemented in different materials and devices. The equivalent technology for standard classical computing is given.**

Type	Scaled CMOS (classical)	Ion trap	Quantum dot	Optical circuit	Gate-based superconducting circuit	Superconducting circuit (adiabatic computation)
State variable	Electrical charge	Ion spin	Electron spin, energy level, or position	Photon polarization, time, or position	Magnetic flux, charge, or current phase	Magnetic flux
Material	Doped silicon	Atoms in free-space electromagnetic field	Solid-state semi-conductor at cryogenic temperatures	Optical waveguides, for example, etched in silicon	Superconducting Josephson junction at cryogenic temperatures	Superconducting Josephson junction at cryogenic temperatures
Device gate	MOSFET	Laser- or vibrational-mediated interaction	Laser- or electrically-driven exchanges and rotations	Beam splitters and photon detectors	Electrically-driven exchanges and rotations	Electrically-controlled couplers
Maximum demonstrated variables	> 10 <sup>9</sup> transistors per chip, ≈ 10 <sup>18</sup> per supercomputing system	14	3	8	2 full qubits + 2 special-purpose memories	8 coupled, 50 functional?

and controlled and entangled. Ions are trapped in a vacuum by electromagnetic potentials, and 14-qubit entanglement has been demonstrated (the largest entangled state in any form shown to date).<sup>27</sup> Complex bench-top linear optical circuits are capable of entangling eight-photon qubit states, and have been shown to perform nontrivial computation over short timescales.<sup>40</sup> Both of these approaches do not scale in those forms, but scalable approaches are also under development. Groups headed by Wineland, Monroe, Chuang, and others have demonstrated the necessary building blocks for ion traps.<sup>21</sup> Integrated nanophotonics (using photons as qubits) made on silicon chips provides the route to getting optics off of the lab bench and into easily scalable systems, and is making substantial progress in groups such as O'Brien's.<sup>19</sup>

Solid-state electronic devices for gate-based computation, while currently trailing in terms of size of entangled state demonstrated, hold out great promise for mass fabrication of qubits.<sup>19</sup> By trapping a single extra electron in a 3D block of semiconducting material, a quantum dot has a quantum spin relative to its surrounding that can hold a qubit of data. For flux qubits, the state variable is the quantum state of the magnetic flux generated by a micron-scale ring of

current in a loop of superconducting wire (Figure 3).

In solid-state technologies, the experimental focus has been on improving the control of individual qubits rather than growing their numbers, but that focus has begun to shift. Jaw-Shen Tsai has noted that superconducting qubit memory lifetime has more than doubled every year since 1999, and has now reached the point where quantum error correction becomes effective.

Overall, the prospects are very good for systems consisting of tens of qubits to appear in multiple technologies over the next few years, allowing experimental confirmation of the lower reaches of the scaling behavior of quantum algorithms and the effectiveness of quantum error correction.

However, one factor that is often unappreciated when looking at these qubit technologies is the physical scale of the devices, particularly in comparison with classical digital technologies. Many people associate quantum effects with tiny objects, but most of these technologies use devices that are enormous compared to the transistors in modern computer chips. Transistors really are reaching down to atomic scales, with vendors having shipped chips fabricated with a 22-nanometer process at the end of 2011. In these chips, the smallest features will be only about 40 times the

silicon crystal lattice cell size. In contrast, although the atoms themselves are of course tiny, ion traps are limited to inter-atom spacing of perhaps tens of microns for RF and optical control. Nanophotonic systems will require components tens of microns across, to accommodate the 1.5 $\mu$ m wavelength light that is desirable for telecommunications and silicon optics. Superconducting flux qubits require a current ring microns across. All of these technologies result in qubit devices that are macroscopic, or nearly so, with areal densities a million times less than computer chips. This fact will have enormous impact on large-scale architectures, as we will see.

**First steps in quantum architecture.** At this lowest level, the job of quantum architecture is to determine how individual qubits or qubit blocks interconnect and communicate in order to process data. There are three main areas where experimental groups have begun to consider architectural implications in designing their systems.

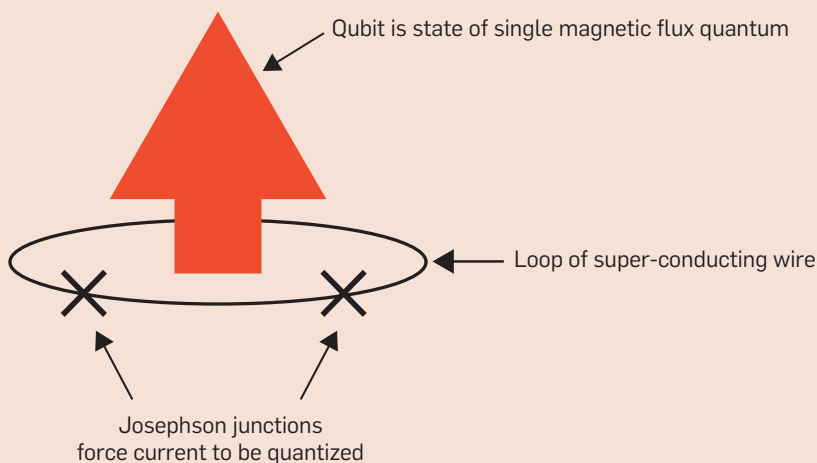
*Heterogeneity.* Some researchers have been investigating technological heterogeneity by using combinations of electron spin, nuclear spin, magnetic flux, and photon polarization in a single system.<sup>5</sup> It is, however, equally important to consider structural heterogeneity, both in operational capability and in interconnect speed or fidelity. Martinis's group has recently demonstrated a chip with a functional distinction between two flux qubits and memory storage elements, leading them to refer to it as a quantum von Neumann architecture.<sup>24</sup> In many technologies, including some forms of quantum dots and Josephson junction qubits, measurement of a qubit requires an additional physical device, consuming die space and making layout of both classical and quantum components more complex.

*Integration and classical control.* Increasing the number of on-chip devices will require improving on-chip integration of control circuits and multiplexing of I/O pins to get away from multiple rack-mount units for controlling each individual qubit. Early efforts at considering the on-chip classical control overhead as systems grow include Oskin's design,<sup>30</sup> and

**Figure 3. The Josephson junction superconducting flux qubit.**

Here, tiny gaps in a superconducting wire (marked with Xs), known as Josephson junctions, force the current in the loop to quantum tunnel across the gap, causing the current to be quantized.

We can use a counterclockwise current (referred to as spin "up," as shown by the large arrow) to be a zero, and a clockwise current (spin "down") to be a one.



recent work by Levy et al. details the on-chip hardware's impact on error correction.<sup>22</sup> Kim uses his expertise in micro-mirrors to focus on the classical control systems for systems requiring switched optical control.<sup>16,18</sup> Controlling meso-scale systems—tens to low hundreds of physical qubits—will require substantial investment in systems engineering. Experimental laboratories likely will have to contract or hire professional staff with mixed-signal circuit design experience to do extensive control circuits, and for many technologies cryogenic circuits are required.

**Interconnects.** Even within individual chips, we are already seeing the beginning of designs with multiple types of interconnects. Integration levels will likely reach hundreds to low thousands of qubits in a single chip or subsystem, but reaching millions to billions of devices in a system will require interconnects that remain only on the drawing board. In some proposals, the inter-subsystem connections are physically similar to intra-subsystem connections, while in others they can be radically different. One ion trap proposal, for example, uses shared, quantum, vibrational modes for intra-node operations and optical connections between separate ion traps.<sup>29</sup>

## Error Correction

The heroic efforts of experimentalists have brought us to the point where approximately 10 qubits can be controlled and entangled. Getting to that stage has been a monumental task as the fragile quantum system must be isolated from the environment, and its state protected from drifting. What will happen, then, when we push to hundreds, thousands, or millions of qubits?

This brings us to the next level of Figure 2, the microarchitecture for quantum computers. If a quantum architecture were designed where gates of an algorithm were run directly on the types of individual physical qubits that we have been describing, it would not work. Both the accuracy of the gate operations and the degree of isolation from the environment required to perform robust computation of any significant size lie far outside the realm of feasibility. To make matters worse, quantum data is subject to a restric-

# Approaches to Quantum Information Processing

Data within a quantum computer is typically stored as *quantum bits* or *qubits*. Like a classical bit, a qubit has two states 0 and 1 but, unlike a classical bit, a qubit may be in a superposition of the two states, being in a certain sense in both 0 and 1 at the same time. A quantum gate operation that changes the state of this qubit can act on both values simultaneously. Each element in the superposition has a (complex) weight, say  $\alpha$  for 0 and  $\beta$  for 1. When measuring a superposed state, only a single result (0 or 1) is returned, but the probability of measuring 0 is  $|\alpha|^2$  and of measuring 1 is  $|\beta|^2$ . We cannot predict which outcome we will see, only its relative likelihood.

The power of superposition extends when we consider qubit registers: by analogy with a classical register, many qubits act together to store data as bit strings. In the quantum case, the register can be in a superposition of all possible register values. For example, a 3-qubit register can be in the superposed state of all eight values 000 to 111, all with different weights. In some such cases, when the superposition contains more than a single qubit, the qubits can be entangled with each other. The individual qubits no longer act independently, and exhibit much more strongly correlated behavior than is possible for classical systems. As with a single qubit, when quantum gates are performed on a register, operations are performed on all values simultaneously.

Extracting the relevant data is the difficult part of quantum computing. Only one element of a superposition can ever be measured, and we cannot choose which one it is. Algorithm designers aim to manipulate the weights so that, when the time comes to measure the result, the majority of the weight is given to a state that is a solution to the input problem.

Several different methods have been developed to use these fundamental elements of quantum computing. The most widely considered is circuit-based computation. Directly analogous with classical digital computation, data is stored in qubits and manipulated by the application of gate operations. In general, the first step of a circuit-based computation is to create an equal superposition of all register states. Gate operations between qubits then change the weights in the superposition, usually creating entanglement in the process.

A separate approach is *adiabatic quantum computation*. As with the circuit model, the output state is measured to give the final answer. In this case, however, the state is designed to be the low energy ground state of a quantum system in the quantum computer. The key to the computation is to adjust the coupling between quantum systems in the device to allow it to relax into that specific ground state.

Other approaches include *measurement-based quantum computation*, in which a large entangled state is reduced to the desired output state simply by carefully choosing how to measure the qubits, and *direct simulation*, in which the quantum states are designed to model a different physical system, rather than calculate a value numerically.

tion known as the “no-cloning theorem” that means standard, classical, methods of controlling error cannot be implemented.<sup>39</sup> Quantum data cannot be “backed-up,” or copied for simple repetition code processing to detect and correct for errors.

The possibility of performing quantum computation is saved, however, by quantum error correction. Some techniques are based on classical error correction and erasure correction, while others are based on uniquely quantum approaches.<sup>9</sup> In all cases, a number of physical qubits are combined to form one or more logical qubits. The number of physical qubits per logical qubit is determined by the quantum operation error rates, the physical memory lifetime, and the accuracy required of the algorithm, and can vary from tens to possibly thou-

sands. A key property of a code is its threshold, the accuracy to which all physical qubit operations must perform in order for the code to work. Once enough physical qubits can be interacted to make interacting logical qubits, with all physical device operations accurate to below the threshold, then error correction will keep the quantum computer error-free for the runtime of the algorithm.

When assessing a classical error correcting code, an important aspect is the code rate, the ratio of delivered, corrected symbols to the number of raw symbols used. High rates are achieved in part by using very large blocks that encode many bits. Block-based codes have been explored in quantum codes, but suffer from the drawback that logical operations on logical qubits within the block are dif-

difficult to execute fault tolerantly. When selecting a quantum code, the rate is important, but the demands made on the implementing hardware and the ease of logical operations are critical.

Performing error correction is not a simple task; in fact, the vast majority of the processing power of a universal quantum computer will be used to correct errors in the quantum state. Application algorithms and data processing make their appearance only at a level well above the real-time, (physical) qubit-by-qubit work of error correction. An architecture for a universal quantum computer will therefore have as its primary goal the execution of specific types of error correction.

The earliest ideas for QEC naturally took advantage of classical error correction techniques. After solving the problems of measuring error syndromes without destroying the quantum state and computing on encoded states without inadvertently spreading errors (known in QC literature as fault tolerance, referring to runtime errors rather than mid-computation failure of hardware components), application of classical error correction became relatively straightforward.<sup>9</sup>

A promising form of error correction is surface code computation, which grew out of work by Kitaev and others on topological quantum computing. Raussendorf and collaborators created the 2D and 3D versions suitable for solid-state and photonic systems, respectively.<sup>11,31</sup> Fowler, Devitt, and others have extended the practicality of these results, including implementing the real-time error processing necessary to determine that the classical half of the machine is a tractable engineering problem.<sup>8</sup> The code rate of surface codes is poor, but their requirement only for nearest-neighbor connections will allow them to work at a higher physical error rate than other methods on some attractive hardware platforms.

Beyond digital quantum error correction for arbitrary states, other approaches can be used to (partially) isolate qubits from undesirable interactions. Decoherence-free subspaces encode a logical qubit in the phase difference of two physical qubits, suppressing the effect of certain

types of noise. Techniques known as spin echo and dynamic decoupling similarly can be used to partially reverse the impact of systematic effects on memory, going with the error for a while and against it for a while, canceling out the effect. Purification—error detection for specific states—will be especially useful for communication, either system internal or external.

The implementation of error correction is perhaps the key near-term experimental goal. As experimental capabilities have grown, groups have begun competing to demonstrate quantum error correction in increasingly complete forms. Blatt's group performed multiple rounds of an error correction-like circuit that detects and corrects certain types of errors using certain simplifications.<sup>32</sup> Pan's group has recently shown an eight-photon entangled state related to the unit cell of 3D surface error correction.<sup>40</sup>

**Microarchitectures for error correction.** As in classical computer architecture, microarchitecture is the bridge between physical device capabilities and the architecture. Microarchitecture in the quantum case can be understood to be the level dedicated to efficient execution of quantum error management, while the system architecture is the organization of microarchitecture blocks into a complete system. There are several specifically quantum elements that must be considered for this microarchitecture.

**Clock speed.** The conversion factor from physical gate cycle to logical gate cycle has a strong, underappreciated impact on the performance of an algorithm. It depends on a number of architectural features, as well as the error correction code itself. For the ion trap-based system analyzed by Clark et al.,<sup>6,26</sup> a 10μsec physical gate results in a 1.6msec error correction cycle time using the basic form of Steane's error correcting code, which encodes one logical qubit in seven physical ones. The code will need to be applied in recursive fashion, resulting in growth of the physical system by an order of magnitude and an increase in the logical clock cycle to 260msec, not far below the topmost quantum line in Figure 1. This dramatic increase illustrates the effect

of the base physical error rate on the architecture and performance, and will be discussed later.

**Efficient ancilla factories.** Most numeric quantum algorithms depend heavily on a three-qubit gate called a controlled-controlled NOT gate, or Toffoli gate. In most quantum error correction paradigms, direct execution of a Toffoli gate on encoded logical qubits is not possible. Instead, the Toffoli gate is performed using several operations, including one that consumes a specially prepared ancilla (temporary variable) state. The ancilla is created using distillation (a quantum error detection code), which takes noisy physical states and builds more accurate logical states. Creation of these states may dominate the workload of the machine, and recent work has assumed that 75%–90% of the machine is dedicated to their production. Isailovic et al. referred to this need as running quantum applications “at the speed of data,” that is, producing the generic ancilla states rapidly enough that they are not the performance bottleneck.<sup>14</sup>

**Balancing error management mechanisms.** A functioning quantum computer almost certainly will not rely on a single type of error correction, but will incorporate different forms of error correction/detection/suppression at different levels. Different error management techniques have different strengths and weaknesses, and the combinatorial space for integrating multiple types is large.

**Defects.** A quantum architecture must also take into account that fabrication will inevitably be an imperfect process. Qubits may be declared defective because they fail to correctly hold the correct state variable carrier (for example, to trap a single electron), because memory lifetime is short or gate control imprecise, or because they fail to couple properly to other qubits. For gate-based, error-corrected systems, calculations show that a stringent definition of declaring a device to be functional pays for itself in reduced error correction demands.<sup>36</sup> A system's resilience to low yield is very microarchitecture-dependent. Alternatively, the digital quantum error correction itself can be adapted to tolerate loss.<sup>34</sup>



## Workloads

So far we have looked at the lower two levels of Figure 2. Before investigating a complete quantum computer architecture, we need to consider the algorithms and programs that will be run on the physical hardware—the workload for the quantum computer. We therefore skip to the top of the stack: quantum programming and quantum algorithms. We are still in the time of Babbage, trying to figure out what Knuth, Lampson, and Torvalds will do with a quantum computer. It has been widely believed that Shor’s factoring algorithm<sup>33</sup> and quantum simulation<sup>3,4,20</sup> will provide the two driving reasons to build machines. There are, however, a number of other useful and interesting quantum algorithms, seven of which are being investigated by teams involved in IARPA’s Quantum Computer Science Program.<sup>b</sup> Bacon and van Dam<sup>1</sup> and Mosca<sup>28</sup> have published surveys covering quantum random walks, game theory, linear algebra, group theory, and more. Our understanding of how to design new quantum algorithms that asymptotically outperform classical ones continues to grow, though the number of people who can put the concepts into practice remains small.

Given the applications we have, how large a computer is needed to run them, and how should it be structured? Only a few quantum algorithms have been evaluated for suitability for actual implementation. Shor’s algorithm is commonly used as a benchmark, both for its importance and clarity, and because the arithmetic and quantum Fourier transform on which it is founded are valuable building blocks for other algorithms.<sup>7,12,26,36</sup> Unfortunately, the size and speed of a machine needed to run the algorithm has been widely misunderstood. Architects have suggested a physical machine comprised of high millions to billions of qubits to factor a 2,048-bit number, a size that experimental physicists find staggering.<sup>7,16,26,36</sup>

In part because designs for a Shor machine have proven to be intimidatingly large, consensus is building that a Shor machine will not be the first commercially practical system, and interest in designing machines for

quantum chemistry is growing. In a somewhat unexpected result, Brown’s group has shown that certain quantum simulation algorithms expected to be computationally tractable on quantum computers are turning out to have dismayingly large resource requirements.<sup>6</sup> However, the field of quantum simulation is varied, and these simulators remain attractive; they are simply going to take more quantum computational resources (hence, more calendar years to develop and more dollars to deploy) than originally hoped.

A key element of the process of developing applications will be programming tools for quantum computers, and enough language designs were under way by 2005 to warrant a survey with a couple of hundred references.<sup>13</sup> In what are arguably the first examples of true “quantum programming,” as distinct from “quantum algorithm design,” shortly after the publication of Shor’s factoring algorithm, Vedral et al. and Beckman et al. produced detailed descriptions of the circuits (sequences of gate operations) necessary for the modular exponentiation portion of the algorithm, which appeared as a single line in Shor’s original paper.<sup>2,38</sup> The next step in implementation is to adapt such a description for execution on a particular machine, as in the block marked “Architecture-aware algorithm implementation” in Figure 2. Matching implementation choices to the strengths of the machine, including choosing adder circuits that match the application-level system interconnect, and trading off time and space, will be a collaboration between the programmer and the tools.<sup>37</sup> Maslov and others have studied efficient architecture-aware compilation,<sup>25</sup> an important element in the IARPA program. Compiler backends for specific experimental hardware configurations will soon be important, as will methods for debugging quantum programs in situ.

## Quantum System Architectures

Finally we come to the central element in Figure 2. A complete system design will specify everything from the “business-level” requirements for an algorithm and machine capable of outperforming classical computers, through the details of the algorithm’s implementation, the strength of error cor-

rection required and type of error management applied, the corresponding execution time of logical Toffoli gates (including the ancilla distillation discussed earlier), and the microarchitecture of individual areas of the system.

The DiVincenzo criteria are fundamental and necessary, but not sufficient to build a practical large-scale system. Considering instead the issues of quantum computer architecture results in a different focus, highlighting such mundane engineering criteria as being large enough and fast enough to be useful, and small enough and cheap enough to be built. Very loosely, meeting the DiVincenzo criteria can be viewed as the responsibility of experimental physicists, while the latter criteria are the responsibility of computer engineers.

Small-scale quantum architecture can be said to have begun with Lloyd’s 1993 proposal for a molecular chain computer,<sup>23</sup> the first for a potentially buildable device. The word “scalable” attained a permanent position in the lexicon with Kielpinski et al.’s 2002 proposal for an ion trap that can shuffle and manipulate individual atoms,<sup>17</sup> an approach that continues to pay dividends. These ideas and many others for multi-qubit devices, such as the quantum von Neumann approach or scalable ion traps with distinct operation and storage sites, sketch local areas of the system using the technological building blocks, but provide no direct guidance on how to organize large systems that meet the goal of solving one or more problems that are classically intractable.

When considering the macro architecture, certain aspects of the design become clear. Because all of memory is expected to be active, the system will probably not consist of separate CPU and memory banks connected via wide, shared buses. A more uniform array of microarchitecture error correction building blocks is the obvious approach, tempered by issues such as defective devices and the needs of the classical control subsystems. Each of these microarchitecture building blocks may utilize heterogeneous technology with an internal storage/computation distinction.<sup>24</sup> Individual chips or ion traps will not be large enough to execute some algorithms (notably Shor) at scale, likely forcing the adoption of a multicomputer structure.<sup>15,29,36</sup>

<sup>b</sup> <http://www.iarpa.gov/solicitations.qcs.html>.  
The IARP QCS was terminated in April 2013.



Large-scale designs are going to be difficult to create and evaluate without the appropriate tools. Further investment in automated tools for co-design of internally heterogeneous hardware and compilation of software is critical. One good example of this practice is Svore and Cross, working with Chuang, who have developed tool chains with round-trip engineering and error correction in mind.<sup>35</sup>

Architectural analyses exist for ion trap systems using Steane error correction, and multiple, distinct forms of nanophotonic and solid-state systems using the surface code.<sup>7,16,26,36</sup> We next take up one moderately complete architecture as an example.

### An Architecture at Scale

We can use error management and application workloads to determine the broad outlines of a computer that could run a useful algorithm at full scale. The size of a quantum computer grows depending on the algorithm's demand for logical qubits, the quantum error correction scheme, the gate and memory error rate, and other factors such as the yield of functional qubits in the system. Overall, including space for various temporary variables and the ancilla state distillation, the scale-up factor from logical to physical qubits can reach half a million. As a specific example, the resource growth in one architecture<sup>36</sup> can be assigned approximately as follows:

- Establish a substantially post-classical goal of factoring an  $L=2,048$ -bit number using Shor's algorithm, requiring
  - $6L$  logical qubits to run a time-efficient form of the algorithm, growing by
  - $8 \times$  to build "singular factories" for the state distillation process, allowing the algorithm to run at the speed of data,
  - $1.33 \times$  to provide "wiring" room to move logical qubits around within the system,
  - $10,000 \times$  to run the Raussendorf-Harrington-Goyal form of the surface code<sup>31</sup> with an error correcting code distance  $d=56$ , and finally
  - $4 \times$  to work around a yield of functional devices of 40%.

A singular factory is simply a region of the computer assigned by the programmer or compiler to the creation of the ancillae discussed here. The size,

## When will a quantum computer do science, rather than be science?

number, and position of the singular factories is dependent on the physical error rate and the size of the computation to be performed, and the type of interconnects available. The space for wiring is a surface code-dependent factor, not required when using other error correction methods, and is probably a substantial underestimate, though researchers are currently looking for compact compilations of programs on the surface code that will minimize this factor. The chosen code distance  $d$  is *strongly* dependent on the application algorithm itself and on the physical gate error rate. Shor's algorithm for  $L=2,048$  demands that, roughly speaking, we must be able to run  $10^{15}$  logical operations with a high probability of correctly executing them all. This work was done assuming a physical error rate of 0.2%, which is not very far below the surface code threshold of 0.75%. These two factors determine the large distance of 56, and in the case of the surface code required resources grow as  $d^2$ , giving the high scale-up factor.<sup>11</sup> The final factor of four is strongly dependent on the details of the microarchitecture and the yield.

This results in a final system size of six billion physical qubits for the main quantum state itself, each of which must be independently controllable. In this particular architecture, this staggering number must be augmented with additional qubits for communications, on-chip optical switches, delay lines, and many external supporting lasers, optical switches, and measurement devices, all deployed in a large multicomputer configuration.

The performance of the system is determined by the error correction time and the complexity of executing the application gates on top of the error correction code. The surface code cycle time on this architecture for measuring all error syndromes is  $\sim 50\mu\text{sec}$ , far slower than the 100psec planned for physical gates. A Toffoli gate will require  $\sim 50\text{msec}$ —a factor of 1,000 from QEC cycle to logical gate, for this code distance. Demonstrating how system-level issues affect performance, the QEC cycle time is limited by contention for access to on-chip waveguides.

In part to address some of these architectural limitations, the QuDOS architecture was developed.<sup>16</sup> QuDOS, if

built, would be 100 times faster, largely due to increased parallelism in measuring the error syndromes. Overall, coordinated changes of physical technology, error correction mechanism, and architecture may gain 3–4 orders of magnitude in performance, demonstrating the impact of the field of quantum computer architecture.

## Conclusion

A principal lesson learned so far in research on quantum computer architectures is that systems capable of solving classically intractable problems will be large, although the search for the smallest commercially attractive machine continues. Device sizes will limit integration levels, affecting architecture, and determining logical clock speed requires making many design decisions but dramatically affects what can and cannot be effectively computed (as shown in Figure 1). Architectural problems cover a broad range, but have received only modest amounts of attention compared to near-term experimental hurdles, leaving much room for high-impact research that can help guide the focus of experimental work.

To sharpen the community focus on building systems, it seems to be time to begin demanding Moore's Law-like improvements in system capacity. Reviewers of papers and funding proposals should look for realistic estimates of logical Toffoli gate time, incorporating error correction, for some target logical fidelity. Even more ambitiously, we recommend requiring realistic estimates of application performance.


Developing a sense of community is critical. Creating a shared understanding including vocabulary, concepts, and important problems among the physics and CS theory, algorithm design, physics experiment, engineering, and architecture communities has proven to be difficult, and few journals or conferences currently provide good venues for such interdisciplinary endeavors, but we expect the number will grow.

Let us close with a question that provokes answers ranging from, "Already has," (in reference to direct quantum simulation of a specific quantum system<sup>20</sup>) to "Twenty years," to "Never,"—and all these from people actually working in the field:

When will the first paper appear in

*Science* or *Nature* in which the point is the results of a quantum computation, rather than the machine itself? That is, when will a quantum computer do science, rather than be science?

## Acknowledgments

This research is supported by the Cabinet Office, Government of Japan, and the Japan Society for the Promotion of Science (JSPS) through the Funding Program for World-Leading Innovative R&D on Science and Technology (FIRST Program). We thank Simon Devitt, Dave Farber, Joe Touch, Michio Honda, Hajime Tazaki, Shigeya Suzuki, the referees and editors for careful reading of the manuscript. 

## Supplemental Material

Additional information and references for this article are available under supplemental material in the ACM Digital Library DOI: 10.1145.249568.

## References

- Bacon, D. and van Dam, W. Recent progress in quantum algorithms. *Commun. ACM* 53, 2 (Feb. 2010), 84–93.
- Beckman, D., Chari, A.N., Devabhaktuni, S. and Preskill, J. Efficient networks for quantum factoring. *Phys. Rev. A* 54 (1996), 1034–1063; <http://arXiv.org/quant-ph/9602016>.
- Brown, K.L., Munro, W.J. and Kendon, V.M. Using quantum computers for quantum simulation. *Entropy* 12, 11 (2010), 2268–2307.
- Buluta, I. and Nori, F. Quantum Simulators. *Science* 326, 5949 (2009), 108–111.
- Childress, L. et al. Coherent dynamics of coupled electron and nuclear spin qubits in diamond. *Science* 314, 5797 (2006), 281–285.
- Clark, C.R., Metod, T.S., Gasster, S.D. and Brown, K.R. Resource requirements for fault tolerant quantum simulation: The ground state of the transverse Ising model. *Phys. Rev. A* 79, 6 (June 2009).
- Devitt, S.J., Fowler, A.G., Stephens, A.M., Greentree, A.D., Hollenberg, L.C.L., Munro, W.J. and Nemoto, K. Architectural design for a topological cluster state quantum computer. *New Journal of Physics* 11 (2009).
- Devitt, S.J., Fowler, A.G., Tilma, T., Munro, W.J. and Nemoto, K. Classical processing requirements for a topological quantum computing system. *International Journal of Quantum Information* 8 (2010), 1–27.
- Devitt, S.J., Nemoto, K. and Munro, W.J. Quantum error correction for beginners. *Reports on Progress in Physics* 76, 8 (Aug. 2013).
- DiVincenzo, D. The physical implementation of quantum computation. *Fortschritte der Physik* 48, 9–11 (2000), 771–783.
- Fowler, A., Mariantoni, M., Martinis, J. and Cleland, A. A primer on surface codes: Developing a machine language for a quantum computer. Arxiv preprint (2012); [arXiv:1208.0928](http://arXiv:1208.0928).
- Fowler, A.G., Devitt, S.J. and Hollenberg, L.C. Implementation of Shor's algorithm on a linear nearest neighbor qubit array. *Quantum Information and Computation* 4, 4 (2004), 237.
- Gay, S. Quantum programming languages: Survey and bibliography. *Bulletin of the European Association for Theoretical Computer Science* (June 2005).
- Isailovic, N., Whitney, M., Patel, Y. and Kubiatowicz, J. Running a quantum circuit at the speed of data. *International Symposium on Computer Architecture*. IEEE (2008), 177–188.
- Jiang, L., Taylor, J.M., Sørensen, A.S. and Lukin, M.D. Distributed quantum computation based on small quantum registers. *Phys. Rev. A* 76 (Dec 2007).
- Jones, N.C., Van Meter, R., Fowler, A.G., McMahon, P.L., Kim, J., Ladd, T.D. and Yamamoto, Y. Layered architecture for quantum computing. *Phys. Rev. D* 3 (July 2012), 031007.
- Kielinski, D., Monroe, C. and Wineland, D.J. Architecture for a large-scale ion-trap quantum computer. *Nature* 417 (2002), 709–711.
- Kim, J. and Kim, C. Integrated optical approach to trapped ion quantum computation. *Quantum Information and Computation* 9, 2 (2009).
- Ladd, T., Jelezko, F., Laflamme, R., Nakamura, Y., Monroe, C. and O'Brien, J. Quantum computers. *Nature* 464 (Mar. 2010), 45–53.
- Lanyon, B.P. Universal digital quantum simulation with trapped ions. *Science* 334, 6052 (2011), 57–61.
- Leibbrandt, D. et al. Demonstration of a scalable, multiplexed ion trap for quantum information processing. *Quantum Information and Computation* 9, 901 (2009).
- Levy, J.E. et al. Implications of electronics constraints for solid-state quantum error correction and quantum circuit failure probability. *New Journal of Physics* 13, 8 (2011).
- Lloyd, S. A potentially realizable quantum computer. *Science* 261 (1993), 1569–1571.
- Mariantoni, M. et al. Implementing the quantum von Neumann architecture with superconducting circuits. *Science* 334, 6052 (2011), 61–65.
- Maslov, D., Falconer, S. and Mosca, M. Quantum Circuit Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 4 (2008), 752–763.
- Metodi, T.S., Thaker, D.D., Cross, A.W., Chong, F.T. and Chuang, I.L. A quantum logic array microarchitecture: Scalable quantum data movement and computation. In *Proceedings of the 2005 International Symposium on Microarchitecture* (2005).
- Monz, T. et al. 14-qubit entanglement: Creation and coherence. *Phys. Rev. Lett* 106, 13 (Mar. 2011).
- Mosca, M. Quantum algorithms (2008); Arxiv preprint [arXiv:0808.0369](http://arXiv:0808.0369).
- Oi, D.K.L., Devitt, S.J. and Hollenberg, L.C.L. Scalable error correction in distributed ion trap computers. *Physical Review A* 74, 052313 (2006).
- Oskin, M., Chong, F.T., Chuang, I.L., and Kubiatowicz, J. Building quantum wires: The long and short of it. In *Proceedings of the 30th Annual International Symposium on Computer Architecture* (June 2003), ACM, N.Y.
- Raussendorf, R., Harrington, J. and Goyal, K. Topological fault-tolerance in cluster state quantum computation. *New Journal of Physics* 9, 199 (2007).
- Schindler, P., Barreiro, J.T., Monz, T., Nebendahl, V., Nigg, D., Chwalla, M., Hennrich, M. and Blatt, R. Experimental repetitive quantum error correction. *Science* 332, 6033 (2011), 1059–1061.
- Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, CA, 1994, 124–134.
- Stace, T.M., Barrett, S.D. and Doherty, A.C. Thresholds for topological codes in the presence of loss. *Physical Review Letters* 102, 20 (2009).
- Svore, K.M., Aho, A.V., Cross, A.W., Chuang, I. and Markov, I.L. A layered software architecture for quantum computing design tools. *IEEE Computer* (Jan 2006), 74–83.
- Van Meter, R., Ladd, T.D., Fowler, A.G. and Yamamoto, Y. Distributed quantum computation architecture using semiconductor nanophotonics. *International Journal of Quantum Information* 8 (2010), 295–323.
- Van Meter III, R.D. Architecture of a Quantum Multicomputer Optimized for Shor's Factoring Algorithm. Ph.D. thesis, Keio University, 2006; [arXiv:quant-ph/0607065](http://arXiv:quant-ph/0607065).
- Vedral, V., Barenco, A. and Ekert, A. Quantum networks for elementary arithmetic operations. *Phys. Rev. A* 54 (1996), 147–153; <http://arXiv.org/quant-ph/9511018>.
- Wooters, W.K. and Zurek, W.H. A single quantum cannot be cloned. *Nature* 299, 802 (Oct. 1982).
- Yao, X.-C. et al. Experimental demonstration of topological error correction. *Nature* 482 (Feb. 2012), 489–494.

**Rodney Van Meter** ([rdv@sfc.wide.ad.jp](mailto:rdv@sfc.wide.ad.jp)) is an associate professor in the Faculty of Environment and Information Studies at Keio University's Shonan Fujisawa Campus, Kanagawa Prefecture, Japan.

**Clare Horsman** ([clare.horsman@gmail.com](mailto:clare.horsman@gmail.com)) was a project assistant professor at Keio University's SFC. She is currently a research assistant at the University of Oxford's Department of CS, Oxford, U.K.

Copyright held by Owners/Author(s)