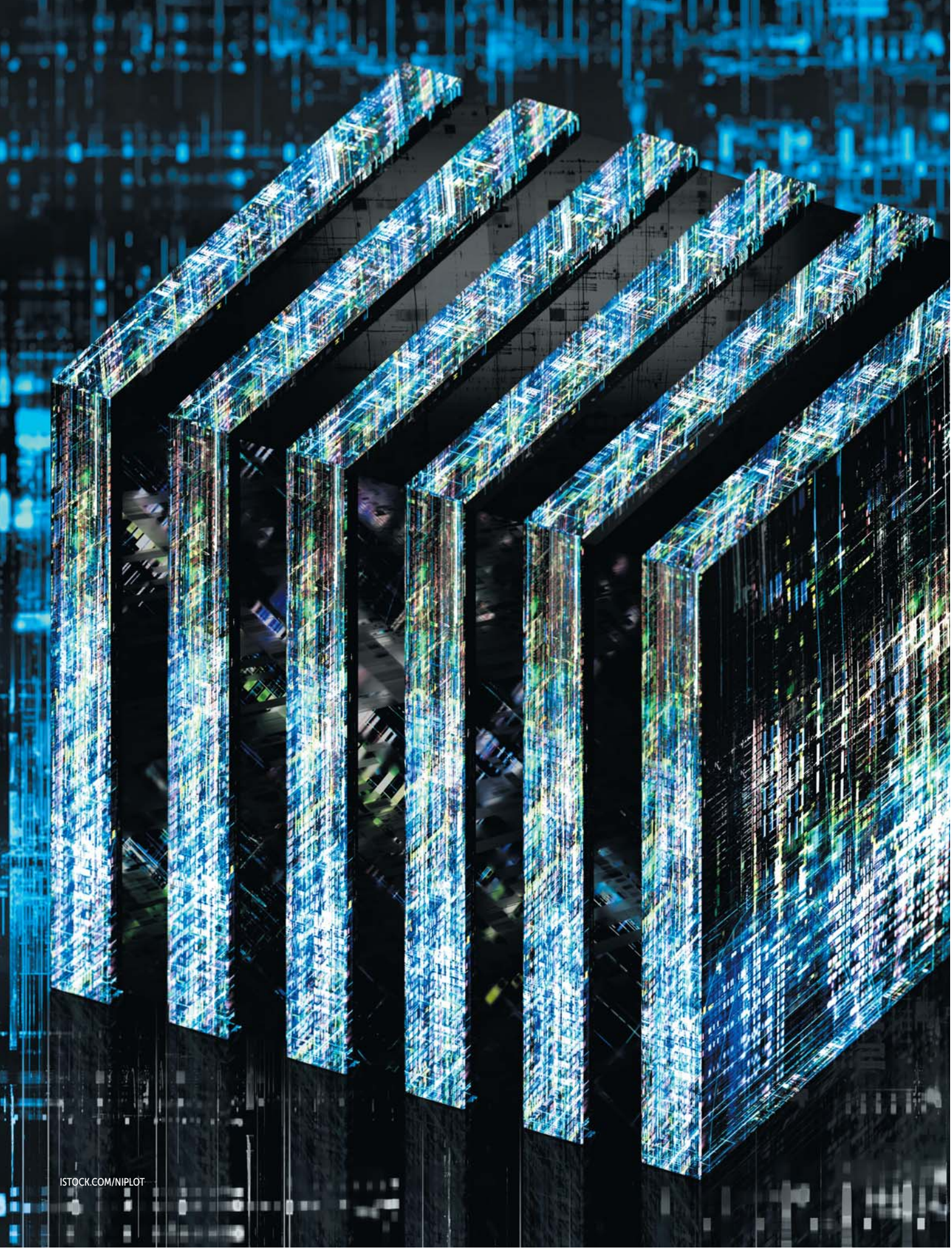


ISTOCK.COM/NIPLOT

Anne Matsuura is the director of quantum applications and architecture, **Sonika Johri** is a quantum algorithms research scientist, and **Justin Hogaboam** is a senior quantum systems architect at Intel Labs in Hillsboro, Oregon.



A systems perspective of QUANTUM COMPUTING

Anne Matsuura, Sonika Johri, and Justin Hogaboam

Quantum architects, with knowledge of both physics and computer engineering, are key to developing scalable quantum computers.

Quantum computing research is coming of age. Recent engineering advances in quantum bit, or qubit, systems have led to a steady stream of successful physics experiments that demonstrate the computational capabilities of small numbers of qubits. Companies are tackling the difficult task of advancing beyond small, lab-scale, proof-of-principle devices to build scalable quantum computing systems that utilize those capabilities. The ultimate goal of a full quantum computing system of commercially relevant scale—thousands to millions of logical qubits—is to harness the power of quantum mechanics inside a technology capable of solving real-world problems that are intractable for classical computers.

Historically, quantum computing research has existed in the realm of physics and mathematics and has concentrated on two fundamental areas: qubits and quantum algorithms. Qubit research has focused on the creation, operation, and performance benchmarking of qubits in experimental devices. Those efforts have been guided by the DiVincenzo criteria, which list the operational conditions necessary to demonstrate quantum computing in a physical system.[1] Quantum algorithm research has focused on developing algorithms[2] that can be implemented on abstract ideal qubit systems and, more recently, on real qubit systems.[3] However, a quantum computing system will be composed of far more than algorithms and qubits, just as a classical computer is made of more than software programs and transistors.

How does one create a quantum computing system that takes a quantum algorithm as input and automatically performs a computation on qubits? Researchers have now begun to define the essential functionalities of the architectural layers[4,5] without specifying how to build the remaining components of a fully operating and

reproducible quantum computing system. Examples of a few of those functionalities are illustrated in figure 1.

Opportunities are available for those willing to acquire the cross-disciplinary skills needed to help build a fully scalable quantum computer. Designers of such a system are asking thought-provoking questions that cross the boundaries between physics, engineering, and computer architecture.

Application-driven design

How does one program and run a quantum algorithm on real qubits? First it is decomposed into a quantum circuit (see box 1) comprising a series of logical operations, or gates. Designers find symmetries in the physics that simplify the circuits and enable the algorithms to run on the few physical qubits available today. However, quantum algorithms cannot be optimized completely by computers yet, and so those painstaking calculations must be done entirely by hand. Aided by software tools, the designer then estimates the number of qubits and gates required by analyzing how the accuracy of the result is affected by parameters such as coherence times, gate fidelities, and approximations in the algorithm. Optimization tools should be developed that eliminate the need to simplify algorithms by hand. For example, they should be able to quickly generate various circuits that achieve the same targeted unitary transformation and then allow the circuit designer to pick the best one.

Today a number of programming environments allow compilers to perform rudimentary optimization of algorithms and prepare them for resource analysis, simulation, or execution on a particular type of qubit. Basically, the compilers take a quantum circuit and translate it into a sequence of logical quantum instructions known as quantum assembly language (QASM). The text-format language represents the quantum circuit as a series of operations. The instruction sequence describes what needs to happen to each qubit in order to run the given algorithm.

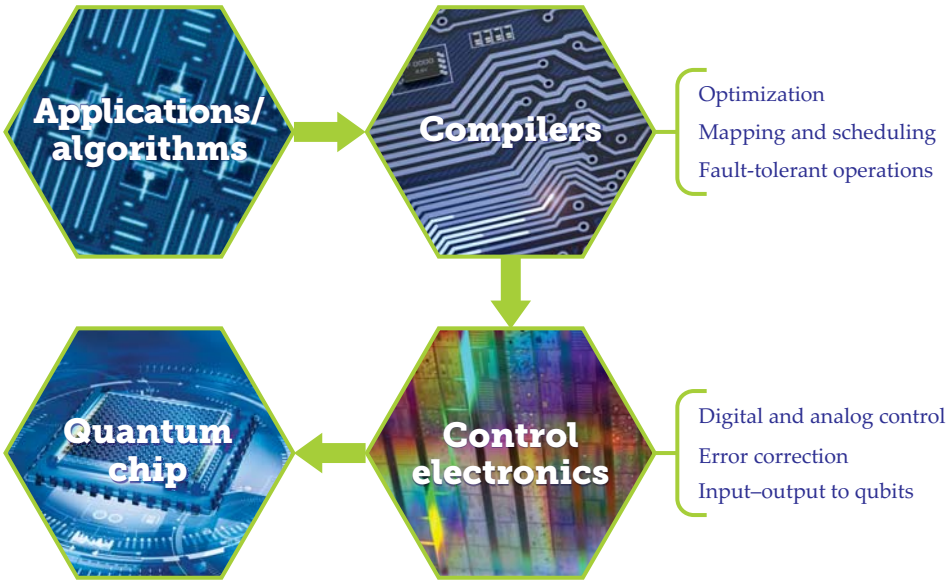


FIGURE 1. FUNCTIONALITIES NECESSARY FOR A QUANTUM COMPUTER are shown here. An application-driven design will help determine how to create a system that will accelerate algorithms from a particular application area. (Applications/algorithms: Jay M. Gambetta, Jerry M. Chow, and Matthias Steffen, CC BY 4.0; compilers: iStock.com/Bet_Noire; control electronics: Steve Jurvetson, CC BY 2.0; quantum chip: Yurchanka Siarhei/Shutterstock.com.)

From the algorithm designer's perspective, the QASM code sequence is now ready to execute on a real qubit system. Unfortunately, all QASM-defined quantum logic gates are not necessarily available on every type of qubit system. Qubits can be created out of many materials, and the choice determines which physical quantum gates are available. Such gates, called native gates, are specific to the underlying qubit system. The logical QASM operations must therefore be translated into the corresponding sets of native gates. However, in doing so, researchers need to consider factors that influence system design and architecture, such as whether the logical qubit operations used in the algorithm can run directly on the qubit system. If they can't, what equivalent gates or gate sequences can be run? What gates can be run in parallel? How many gates can be run in series within the fidelity, or error, budget?

After choosing the appropriate logical operations and type of qubit, the researcher applies the sequence of native gates to real physical qubits. From an algorithm designer's perspective, a two-qubit gate can be applied between any pair of available qubits. However, in many types of systems, those two qubits must be physically near one another to perform the operation. Thus the algorithm needs to be mapped to the parts of the qubit grid capable of executing operations and be sequenced into a schedule of operations that can be optimally implemented, sometimes in parallel, within the limitations of the quantum device.

Mapping and scheduling the algorithm operations onto the qubits is difficult because most qubit devices have limited, often nearest-neighbor, connectivity, which places constraints on where particular qubit operations can be implemented. Operations between more distant qubits may require shuttling or quantum teleportation, which rapidly depletes the quantum resources available to run the algorithm.[6] Mapping and scheduling protocols are made even more challenging because physical qubits are notoriously fragile and noisy, so protocols need to be created that can adapt to those imperfections.

Ensuring that a quantum computer is initialized to a state that is as similar as possible for all algorithmic runs is also important because quantum algorithms must be executed repeatedly to achieve a statistically meaningful result. The process must be performed in a rapid and accurate manner so that it does not introduce an additional source of error into potentially thousands of runs of the quantum circuit.

As in classical computing, error correction will be critical for the reliable functioning of a quantum machine, but classical

error-correcting protocols do not work directly for quantum computing. Like bits in classical computers, the information encoded in individual qubits can be destroyed by environmental noise. However, quantum computing error sources are more numerous, and error rates are much higher. Quantum error correction involves encoding a logical qubit state in multiple physical qubits and using measurements and classical computing resources to detect errors quickly enough to correct them during computation. In today's Noisy Intermediate-Scale Quantum (NISQ) era,[7] the short qubit coherence times and low gate fidelities require a high ratio of physical to logical qubits. The lack of active feedback in most current qubit devices means that state-of-the-art ones still cannot apply a nondestructive error measurement to a series of qubits, identify the source of errors, determine a correction, and implement the correction within the lifetime of the physical qubits.

Since the number of physical qubits is currently limited, error-mitigation techniques, as opposed to full, rigorous error correction, need to be designed for the NISQ-era devices. One such technique is the removal of errors using additional or postselected measurements.[8] The hope is to eventually build a fully fault-tolerant quantum computer that is robust to physical errors. Such a computer requires qubits with long lifetimes, high-fidelity gates, and fast feedback—all challenges for the research field.

The first quantum computers probably will be coprocessors coupled with a classical central processing unit. In such a model, the quantum computer acts as an accelerator for a particular part of the algorithm. In the short term, when error correction is not available, the most promising algorithms are quantum–classical hybrids, which require an intimate coupling between the classical and quantum processors. Thus architectural design

BOX 1. CODESIGN: QUANTUM FOURIER TRANSFORM

In a quantum circuit, as shown on the left below, each line corresponds to a particular qubit. Each box represents an operation, and the chronological order proceeds from left to right. Quantum gates represent the unitary operations that need to be applied on the qubits and are denoted by specific symbols. Quantum gates are the building blocks of a quantum circuit, just as classical logic gates are the building blocks of digital circuits in classical computing. This particular diagram describes a quantum Fourier-transform (QFT) algorithm. Here H is the Hadamard gate, a one-qubit rotation that maps the qubit basis states |0> and |1> to two superposition states: |0> maps to |0>+|1> and |1> maps to |0>−|1>, where

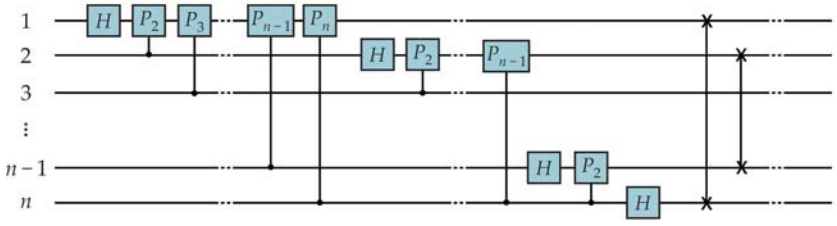
$$H = 1/\sqrt{2}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}.$$

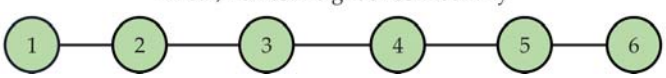
A controlled phase gate, $C\text{–}P_n$, acts on two qubits and applies a phase change when they are both in the |1> state:

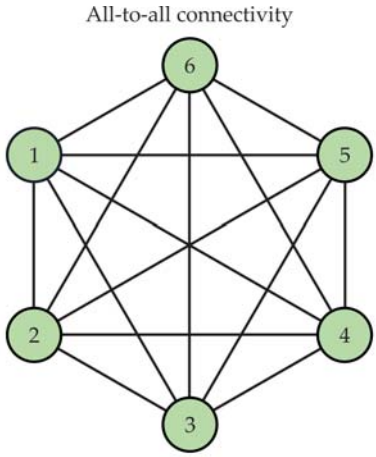
$$C\text{–}P_n\,|j\rangle \otimes |k\rangle = \exp\left(\frac{2\pi i(kj)}{2^n}\right)|j\rangle \otimes |k\rangle.$$

Each SWAP gate, denoted by a vertical line at the far right of the circuit, exchanges the quantum states of two qubits.

The quantum circuit is executed on n qubits, and the total number of gates scales as $O(n^2)$. The O notation classifies algorithms according to how their running time or space requirements grow with the input size. However, the circuit depth—the longest path between algorithm input to output—can be calculated as $O(n)$ in the optimal, theoretical case, when every qubit is connected to every other qubit, as shown on the right. The green circles represent individual qubits, and the black lines indicate where two-qubit gates are possible. It would be much easier to manufacture a linear array of qubits with nearest-neighbor connectivity as shown below on the left. A surprising result is that the QFT can be scheduled to run even on a linear array of qubits with a circuit depth that also has $O(n)$ scaling and only a small constant-factor overhead of 1.25. (For details, see A. Holmes et al., https://arxiv.org/abs/1811.02125.) The pseudocode for the algorithm at the bottom right has an outer loop that executes n times. Within the outer loop are two loops that can each be executed in parallel, which implies O(1) time steps. That algorithm construction also eliminates the need in the end for an extra time step with SWAP gates.







Algorithm 1 Linear Quantum Fourier Transform

logical qubit x_i mapped to physical qubit q_i

```
H (q1)
for i in {2, 3, 4, ..., n − 1, n, n − 1, ..., 2} do
    j ← i
    while j ≥ 2 do Parallel
        CPj (qj, qj−1)
        j ← j − 2
        if j == 1 then
            H (q1)
        end if
    end while
    j ← i
    while j ≥ 2 do Parallel
        SWAP (qj, qj−1)
        j ← j − 2
    end while
end for
H (q1)
```

features throughout the computing system will likely be a mix of classical and quantum pieces. In the design of the instruction sequence, for instance, some researchers advocate for a hybrid of quantum and classical instructions,[9,10] while others propose that the instruction sequence should be purely quantum.[11] Whereas the efficient use of coprocessors in computer architecture is a well understood benefit of modern design, how to implement hybrid architecture for quantum computing remains an open research question.

Classical electronics for controlling qubits will be an integral part of any quantum computing system. Currently, even small qubit systems require racks of laboratory electronics and numerous wires for controlling and operating qubits in a cryogenic refrigerator or ultrahigh vacuum chamber. As the number of qubits scales up, the increase in on-chip and input–output wiring interconnects will introduce more heat and noise to the qubit system. Noise is also easily added during the often long delay times necessary to send electrical signals to the qubits in the chamber. Current qubit hardware uses between five and seven input–output cables for each qubit. However, that arrangement does not scale beyond a few tens of qubits before manufacturers would need to build larger, custom dilution refrigerators. The problem of interconnect scalability for qubit control will be critical to any quantum computing system of useful size.

Different qubit connectivity layouts and engineering constraints, such as the number of control lines relative to the number of qubits and the parallelization and selectivity for qubit control operations, introduce further restrictions. However, such restrictions offer opportunities for optimization. At Intel Labs, for example, research has shown that the quantum Fourier-transform algorithm can be scheduled to execute on a linear array of qubits with nearest-neighbor connectivity almost as efficiently as a fully connected qubit system, as shown in box 1.

The key to designing a quantum computing system is to develop a library of algorithms that are small building blocks of larger, real-world applications in areas such as quantum chemistry and condensed-matter physics. Those algorithms may then be used to drive the design of the full quantum computer system down to the physical qubits. Such an approach will help researchers understand the appropriate layers of functionality required to run the algorithm on real qubits and to design a scalable quantum computing system that incorporates those layers. Even with system noise, the knowledge gained by running small algorithmic building blocks can help researchers improve the system organization and architecture, including the optimal connectivity of the qubits and the appropriate qubit grid organization. By keeping the number of gates low, researchers can run small algorithms on as few as five to seven qubits[12,13] without using quantum error correction. If all the commonly used components of algorithms from a particular application area can be run separately, the hope is that full-scale quantum algorithms can be run on the larger-scale qubit system as the number of qubits and executable circuit depth are scaled up. An application-driven design will thus result in a system architecture that will serve as an accelerator for the particular application area. Although difficulties will arise as successively larger qubit systems are created, running algorithms on each generation of quantum hardware will allow researchers to learn from each new system and solve scaling problems in a systematic manner.

System versus qubit performance

Traditionally, quantum computing performance has been primarily focused only on qubits. The most important quantification metrics have been the physical characteristics of the qubit technologies themselves, such as whether they satisfy the DiVincenzo criteria by concentrating on longer qubit coherence times and on qubit gate performance as measured by gate error rates, gate execution speed, and interconnectivity. However, for quantum computing to move beyond physics research to a computer technology, researchers need to start thinking in terms of overall system performance, which is ultimately what end users will care about most.

Research that compares the hardware architectures of two systems built from different qubit technologies[14] illustrates issues from a hardware perspective. However, that perspective leaves out many components that are critical to a complete quantum computing system beyond individual qubit performance. The field is beginning to move toward approaches such as IBM's quantum volume metric that defines a family of quan-

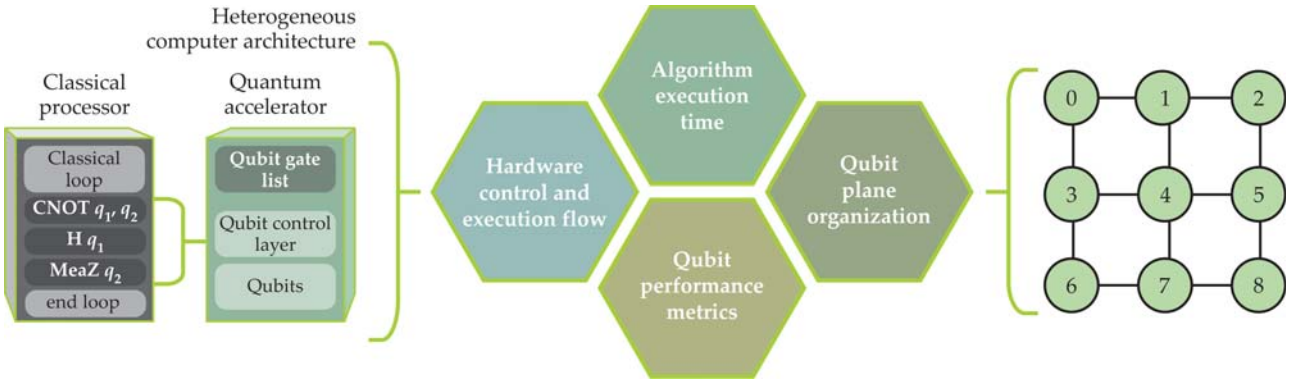
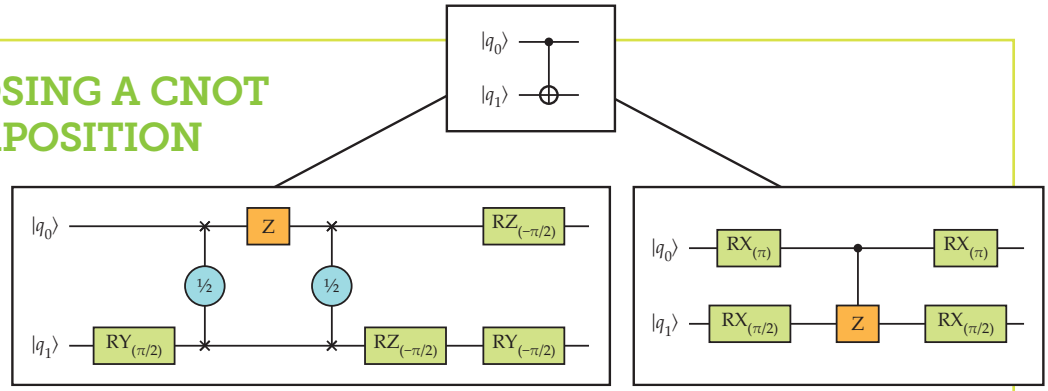


FIGURE 2. RESEARCHERS CONSIDER FOUR INTERACTING CATEGORIES OF FUNCTIONALITIES when designing a system capable of running a quantum algorithm automatically on real qubit devices. For example, the connectivity of the qubits, as shown on the right, may influence the number of control lines that connect to the qubits in the cryogenic refrigerator. The left side shows how the quantum coprocessor may interact with the classical processor.

BOX 2. CHOOSING A CNOT GATE DECOMPOSITION



A logical controlled-NOT, or CNOT, gate is an entangling operation that flips the target qubit between 1 and 0 if the control qubit is in the 1 state. It must be decomposed into a sequence of native quantum gates for the qubit technology to perform the gate operation on the specific qubit system. Two possible decompositions are shown, where RX, RY, and RZ denote rotations around the *x*-, *y*- and *z*-axes, respectively. A system performance simulation could provide metrics to help choose which CNOT to incorporate into the specific design.

Depending on the fidelity of the single- and two-qubit gates in the circuits and on their speed, researchers may want to choose only one to implement the logical CNOT in the system. Depending on the performance of the qubits available at a particular point in the execution of the algorithm, it is also possible to choose a different logical CNOT sequence.

tum circuits for measuring system performance.[15] Challenges in the field remain, such as defining a set of broader system performance metrics appropriate for a larger-scale computing technology and building on the lessons learned from running realistic algorithms for useful application areas on small qubit systems.

As quantum computing systems scale beyond small numbers of qubits running small algorithms, researchers may be able to leverage well-established engineering techniques from classical computing to learn how to design and scale quantum computers to sizes that are useful for more complex quantum algorithms. A few of the trade-offs architects consider when designing a quantum computing system are shown in figure 2. **Algorithm execution time** refers to the amount of resources that must be devoted to compiling and scheduling a sequence of quantum gates to run on the specific qubit device technology and qubit plane organization provided by the hardware. **Hardware control and execution flow** consist of the set of available quantum gates and the parallelism and degree of individual qubit control afforded by classical control electronics that will be key to determining how a system performs. **Qubit performance metrics**, such as one- and two-qubit gate fidelity, state-preparation and measurement errors, and coherence times, influence the maximal executable circuit depth, fault tolerance approach, and other system performance characteristics. **Qubit plane organization** considers how connected the physical qubits are to each other. If the qubit plane provides nearest-neighbor, two-dimensional planar connectivity between qubits, it will be possible to implement topological error-correction codes. If that degree of connectivity cannot be provided, then a repetition code or another error-correcting approach must be taken, which will affect the amount of fault tolerance that the system can provide.

The new multidisciplinary field of quantum architecture

In classical computing, software tools are used to model and simulate the functioning of all the components in the system, with its limitations and constraints, to enable better hardware design. Numerous alternate designs are first modeled in a system performance simulator before any hardware is built. In the case of quantum computing, creating a system performance simulator is a more computationally expensive task because of the superposition of states and the entanglement of qubits. In addition, the physics of the qubits themselves affects the functioning of the quantum computing system. The challenge is to construct a simplified Hamiltonian for a grid of qubits that when incorporated into the system performance simulator provides predictive insights for system design decisions without exceeding the running time and memory constraints of the simulation.

A system performance simulator for quantum computing involves two simulator classes. The first, a system simulator, models the software architecture, hardware architecture, and control electronics from the compiler down to the classical control pulses and interacts with the second, a quantum device simulator that mimics the Hamiltonians of few-qubit systems and the interface with classical control.

The first kind of simulator captures basically everything not quantum about the quantum computer, such as programming languages, the compiler, control schemes, and qubit connectivity. Such a simulator may help address quantum architecture questions such as the following: How many qubits are needed to enable useful applications? What number of qubits should the quantum architecture be able to handle in the next 10–20 years? Where should the division between room temperature and cryogenic control be? Should different elements of the architecture be constructed of different qubit types?

The second kind of simulator will consist of the Hamiltonian of a small number of qubits. It takes as inputs device-level metrics, such as the coherence times, one- and two-qubit gate fidelities, electromagnetic cross talk between qubits, qubit connectivity, and the electronics that are used to control the gates. Since a qubit is often an approximation for a multilevel quantum system, a low-level simulator may involve extra levels that have perturbative but nonvanishing effects on the system. The simulator may also incorporate noise such as charge-trapping defects for semiconductor dots and will allow researchers to perform functions like optimizing the implementation of

common quantum computing algorithms, pinpointing the most damaging sources of noise, and debugging quantum hardware.

It would be more powerful to combine the two simulator approaches into one overall quantum computer system performance simulator. The joint simulator could run small algorithms on a model with, for instance, a specific compiler, qubit control system, type of qubit, and qubit connectivity. It could also analyze modifications to any of those components and reveal how the changes would affect overall computation. Computer architects then could make better design choices at both classical and quantum levels of the quantum computing system. Furthermore, experimentalists and qubit designers could analyze the impact of their choices regarding qubit type, connectivity, and control constraints and better understand how their qubits might be used in a full system environment.

Box 2 details a particular system-level design choice that could be facilitated by a system performance simulation. It shows two ways to decompose a common logical operation, the CNOT gate—a two-qubit gate that performs a NOT operation on the second qubit only when the first qubit is in state $|1\rangle$—into physical qubit operations and rotations. For a specific quantum system, the best choice for the physical gate decomposition will depend on many factors, such as qubit coherence times, gate fidelity, the execution time of the gate operation, and qubit connectivity. A simulator that models the entire quantum computing system will help with those design decisions.

The classical–quantum design choices that need to be made throughout the quantum computing system require researchers with knowledge of both physics and computer architecture design. The new field of quantum architecture stands poised to be an exciting career choice for a new generation of physicists. It bridges the boundary between quantum and classical computing and will be key to building truly useful quantum computers in the future.

The authors gratefully acknowledge contributions from Jim Held and Xiang Zou.

REFERENCES

1. D. P. DiVincenzo, *Fortschr. Phys.* **48**, 771 (2000).
2. A. Montanaro, *npj Quantum Inf.* **2**, 15023 (2016).
3. R. Van Meter, C. Horsman, *Commun. ACM* **56**, 84 (2013).
4. F. Chong, D. Franklin, M. Martonosi, *Nature* **549**, 180 (2017).
5. N. C. Jones et al., *Phys. Rev. X* **2**, 031007 (2012).
6. M. Oskin et al., in *Proceedings of the 30th Annual International Symposium of Computer Architecture*, ACM (2003), p. 374.
7. J. Preskill, *Quantum* **2**, 79 (2018).
8. J. I. Colless et al., *Phys. Rev. X* **8**, 011021 (2018).
9. R. S. Smith, M. J. Curtis, W. J. Zeng, https://arxiv.org/abs/1608.03355.
10. A. W. Cross et al., https://arxiv.org/abs/1707.03429.
11. A. JavadiAbhari et al., in *Proceedings of the 11th ACM Conference on Computing Frontiers,* ACM (2014), art. no. 1.
12. N. M. Linke et al., *Phys. Rev. A* **98**, 052334 (2018).
13. J. S. Otterbach et al., https://arxiv.org/abs/1712.05771.
14. N. M. Linke et al., *Proc. Natl. Acad. Sci. USA* **114**, 3305 (2017).
15. L. S. Bishop et al., *Quantum Volume*, IBM technical report (4 March 2017), available at https://pdfs.semanticscholar.org/650c/3fa2a231cd77cf3d882e1659ee14175c01d5.pdf.

PT