


Emerging Technologies for Quantum Computing

Jonathan M. Baker , University of Chicago, Chicago, IL, 60637, USA

Frederic T. Chong , University of Chicago, Chicago, IL, 60637, USA and also Super.tech, Chicago, IL, 60615, USA

Despite promising proof of concept demonstrations, currently available quantum hardware suffers from fundamental scalability limitations. The field is relatively new and it is imperative to consider emerging technologies in the space and evaluate their unique tradeoff spaces to determine how to best close the gap between current devices and target applications. Here, we explore three recent developments on this front. First, we consider extensions to currently available hardware, which allow the use of higher level states, beyond the usual binary, which when used temporarily can confer circuit-level advantage. Second, we consider the use of superconducting resonant cavities to reduce hardware requirements to implement quantum error correction protocols. Finally, we consider the use of neutral atoms, which offer unique strengths and weaknesses. It is valuable to evaluate new technology early and often to determine the best path toward scalability.

Quantum computing is an emerging technology, so it may seem some sort of hyperbole to consider new emerging technologies for this paradigm, but we are already at the horizon of some potentially game-changing capabilities. Although current machines have shown impressive success with devices based upon trapped ions and superconducting transmons, it is unclear what the eventual winning technologies will be. In this article, we will consider neutral atoms and superconducting resonant cavities and extensions to currently available technology, and their implications for quantum architectures.

Quantum hardware is still in its relative infancy, boasting tens of qubits as opposed to the thousands to millions needed to execute important algorithms for unordered search and factoring.^{1,2} Most available systems have struggled to scale beyond their prototypes while simultaneously suppressing gate errors and increasing qubit coherence times. This limits the types of programs which can execute effectively, let alone perform error correction. It is unclear whether

systems composed of superconducting qubits or trapped ions, the major industry players will take the lead.

Device success is predicated on the increase in the number of qubits, reduction of gate errors below error correction thresholds, and increase in qubit coherence times. In the near term, this translates into larger, deeper programs with improved output distributions. But, in the long term, this translates into qubits which are protected from noise inherent in operating quantum systems in the form of encoded logical qubits.

In recent years, there have been a number of improvements throughout the compilation pipeline both close to the hardware and high-level circuit optimization. These optimizations aim to reduce gate counts, circuit depth, and communication costs as proxies for increasing the size and quality of programs executable on currently available hardware.

An alternative approach is to evaluate the viability and tradeoffs of new quantum technology and associated architectures. Despite tremendous efforts at both the hardware and software level to minimize the effects of noise, it is unclear if any of the available hardware will be able to scale as needed and no clear winner on underlying hardware has emerged. Even further, it is unknown whether this hardware is best

suiting to execute the desired programs and while it is often the case that we adapt compilation to the hardware, i.e., transforming applications into the right shape for execution, an alternative approach is to explore how to design new architectures which are better suited for the applications we want to run.

Evaluating new hardware technology at the architectural level is decidedly different, though intrinsically coupled to the development of quantum hardware. Device physicists' goal is often to demonstrate the *existence* of high quality qubits and operations in prototypes, while the architect's goal is to evaluate the systems level ramifications of this technology, exploring the inherent trade-off spaces to determine viability in the near and long term. Perhaps most critically, this architectural design exploration leads to important insights about what is most important for hardware developers to optimize. For example, if some limitations can be effectively mitigated via software, hardware developers can focus on other more fundamental issues. This process of codesign, by evaluating new technology early and often, is central to accelerating scalability.

In this work, we detail three key developments on this front. First, we discuss the temporary use of qudits which can be used to accelerate key circuit components, an example of evaluating a fundamental architectural question of computing radix.^{3,4} Second, we explore the use of local "memory-equipped" superconducting qubits to reduce hardware requirements to implement error correction, an example of application-driven hardware design.⁵ Finally, we address the use of neutral atoms as a competitive alternative to industry focuses, an example of using software to mitigate fundamental hardware limitations to both guide development and accelerate scalability.⁶

EXTENDING THE FRONTIER VIA INTERMEDIATE QUDITS

Quantum computation is typically expressed as a two-level binary abstraction using qubits. However, superconducting and trapped ion qubits based quantum architectures are not intrinsically binary and have access to an infinite spectrum of discrete energy levels. Typical implementations actively suppress higher level states. Some gate implementations have made use of higher level states to implement multiqubit interactions, while others are designed to actively mitigate leakage, a source of error which leaves the qubit state left in higher level states.

Higher radix computation has been explored previously in classical computation, but is usually used in specialized applications and general computation

obtains only limited (constant) advantage. Similar work in quantum computation has demonstrated a constant advantage; by expressing an N qubit circuit as one using $M = N/\log_2(d)$ qudits, where a qudit is a d -level system.⁷

This advantage is still critical. Both qubits and gates between them are error prone. As devices scale with more qubits, the relative connectivity of these qubits decreases requiring an increasing number of error-prone gates to interact arbitrary pairs of qubits. Therefore, reducing the hardware resource requirement enables larger programs to be executed while requiring fewer gates.

Full translation from one radix to another (for example binary to ternary) offers constant advantage, however, more clever usage can offer even more. Many quantum algorithms make use of ancilla, additional free bits which are used to store temporary information. In some cases, they can provide asymptotic improvements in the depth of circuit decompositions, highlighting an important space-time tradeoff in quantum programs. By using additional space, in the form of additional qubits or devices, we can reduce the total execution time of the program. When programs are time-limited, for example, by prohibitive coherence times, it is critical to decompose circuits to minimize depth. However, this requires that we maintain a delicate balance as using ancilla limits the total size of programs, which can be executed if a large portion of qubits must be reserved.

Real quantum hardware will have a limited number of qubits so it is critical to make the most of them to enable computation of larger, more useful programs sooner. An alternative approach to full program translation is to make use of qudit states *temporarily* during binary computation, which extends the number of available computational qubits by reducing the ancilla requirements of key circuit elements while still maintaining low-depth decompositions. By accessing higher level states, the computation is subject to a wider variety of errors. If used properly, the amount gained outweighs this cost. Importantly, most quantum hardware does not currently support the execution of multiqubit gates, those with more than two inputs, and instead they must be decomposed into circuits using only one and two input operations.

Here we detail two specific uses. The first is a generalized Toffoli decomposition, a circuit which computes the reversible AND of many input bits.⁴ The most efficient decompositions use many ancilla to temporarily store the ANDs between pairs of inputs, recursively. Rather than using a full extra qubit, we temporarily allow our qubits to access the $|2\rangle$ state

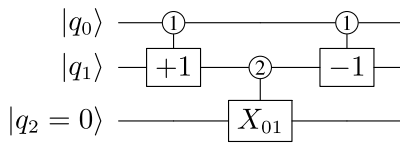


FIGURE 1. Toffoli AND using temporary qutrits (bottom). The value in the circle indicates on which value to execute and the symbol in the box indicates which operation is executed, for example to add 1 modulo 3. Typical Toffoli decompositions use 6 two-qubit gates. Figure from Gokhale *et al.*⁴

and store the temporary AND results *locally*. The simplest version of this is to look at the Toffoli gate. First we can execute a 1-controlled +1 gate. This will cause the second input to be in the state $|2\rangle$ if and only if *both* of the inputs were $|1\rangle$ to begin. Then, executing a 2-controlled X gate onto the target will flip the target if and only if both inputs were $|1\rangle$ which is exactly what a Toffoli gate should do. Following this operation, we want to ensure our inputs return to being only a superposition between the lowest two levels (i.e., return to being a qubit), so we perform some uncomputation. This circuit construction is found in Figure 1. The idea is similar in the general case—store the ANDs of inputs locally as $|2\rangle$ rather than on another ancilla.

It turns out with temporary use of qutrits, we can obtain the same asymptotic depth and gate count as the most efficient decomposition using ancilla without using any ancilla. Specifically, we obtain a logarithmic depth decomposition using no additional space in the form of extra devices. The best known decomposition using only qubits requires linear depth with a large coefficient making it impractical to use. While effective, this strategy has fairly narrow uses requiring hand optimization to be effective.

Second is a decomposition of a reversible adder, which computes the reversible sum of the two inputs onto the second input. With a technique called “compression” we can *generate* the required ancilla in-place to obtain logarithmic depth, the best known depth for decompositions with ancilla.³ The simplest example is to consider the storage of two qubits into a single ququart (four level system). The two qubits and the single ququart can each be written as a superposition of four basis states. Therefore, all of the information of the two qubits can be stored in a single ququart. By allowing one of the qubit’s to access two additional levels, we store all of the information locally leaving the other qubit in the $|0\rangle$ state, effectively an ancilla bit. This technique means we can generate ancilla local to the computation, using unused qubits

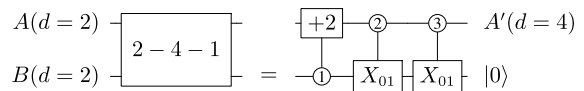


FIGURE 2. The compression of 2 qubits into a single ququart and generating an ancilla, $|0\rangle$. input two qubits, A and B, and produces a single ququart and an ancilla $|0\rangle$. To retrieve the stored information we can do the inverse of this operation using *any* ancilla for the second qubit. Using this type of compression circuit can produce clean ancilla on demand by storing unused data temporarily in higher states. Figure from Baker *et al.*³

nearby. A circuit for this compression is found in Figure 2.

This technique has been shown to be powerful for arithmetic circuits. For example, addition circuits can be decomposed in logarithmic depth without any external ancilla. The circuit is broken in a constant number of blocks, where each block is decomposed individually using the other unused blocks to generate the ancilla needed for an efficient decomposition. This block-based construction along with compression is very powerful. If efficient circuit decompositions are known for qubit circuits requiring ancilla and need only a constant amount of information to move between blocks, we can use this compression technique to obtain the same asymptotic depth as the known decomposition.

Both of these techniques free up more of our limited hardware for computation, rather than dedicating device space as ancilla which inherently limits the maximum size of computation which can be performed. Current hardware is often calibrated for use only with qubits but has higher level states available already. While classically multivalued and mixed-radix computing is niche, it is important to evaluate the architectural ramifications of these strategies for quantum computation. By temporarily accessing already available higher states, we can accelerate common circuit components and reduce space overhead extending the frontier of what can be computed.

VIRTUALIZING LOGICAL QUBITS WITH LOCAL QUANTUM MEMORY

Current quantum architectures do not tend to make a distinction between memory and processing of quantum information. These architectures are viable, however, as more and more qubits are needed the scalability challenges become apparent. For example, many industry players make use of superconducting

transmon qubits, which suffer from fabrication inconsistency and crosstalk during parallel operations. To scale to the millions of qubits needed for error correction, a memory-based architecture can be used to decouple qubit-count from transmon count.

It is imperative to explore the use of technologies outside of what is currently commercialized or made available via cloud services. For example, recently realized qubit memory technology which stores qubits in superconducting resonant cavities may help to realize exactly this memory-based architecture.^{8,9} In these devices, qubit information can be stored in the cavity and when an operation needs to be performed it can be transported to the attached transmon. Local memory is not free. Stored qubits cannot be operated on directly and all operations must be mediated via the transmon by first loading the information. This further prohibits parallel operations in qubits in the same cavity requiring serialization. The realization of this technology alone is not sufficient to understand its viability and instead dedicated architectural studies are needed.

Here we explore a proof-of-concept demonstration of its viability by virtualizing surface code tiles (one example of quantum error correction) in a 2.5-D memory-based architecture.³ Logical qubits can be stored at unique virtual addresses in memory cavities when not in use and are loaded to a physical address in the transmons for computation or to correct errors, similar to DRAM refresh. Figure 3 depicts the proposed architecture with surface tiles mapped to unique virtual addresses. To minimize the use of transmons, the goal is to take logical qubits stored in a plane and find an embedding of that plane in 3-D, where the third dimension is limited to a finite size. The simplest embedding procedure is to slice the plane to form patches, one for each logical qubit and stack them into the layers so that each shares the same physical address and therefore transmons. This procedure is not the most space efficient, since the ancilla needed to detect errors occupy their own unique transmons which are unnecessary. A compact embedding reduces the physical transmon cost by an additional 2× at the cost of some additional time to detect errors.

The initial benefit is clear—this design requires many fewer physical transmons by storing many logical qubits in the same physical location. There are other advantages such as a faster transversal CNOT, which is traditionally executed using a sequence of many primitive surface code operations. Furthermore, this design requires fewer total qubits to distill $|T\rangle$ states, which are commonly used for universal quantum computation. These improvements translate to

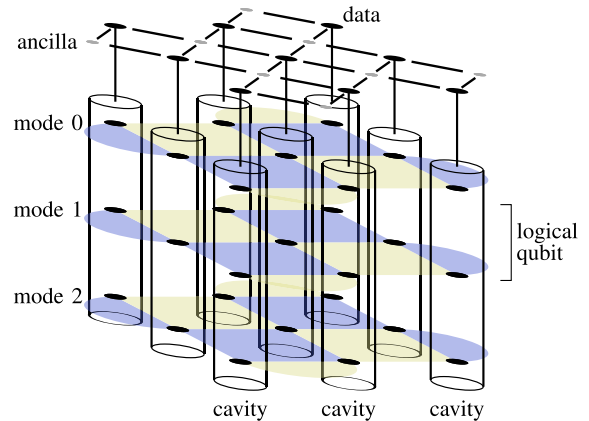


FIGURE 3. A fault-tolerant architecture with random-access memory local to each transmon. On top is the typical 2-D grid of transmon qubits. Attached below each data transmon is a resonant cavity storing error-prone data qubits (shown as black circles). This pattern is tiled in 2-D to obtain a 2.5-D array of logical qubits. The key innovation here is to store the qubits that make up each logical qubit (shown as checkerboards) spread across many cavities to enable efficient computation. Figure from Baker *et al.*⁵

gains in important algorithms by reducing execution time and resource requirements, specifically a 1.22× speedup for Shor’s algorithm or allowing it to run on smaller hardware.

Physical qubit savings alone is not sufficient to guarantee its competitiveness. We must ensure the error thresholds (approximately how good physical operations need to be in order for errors to be efficiently corrected) are as good, if not better, than standard implementations and that all of the necessary operations, such as single qubit gates and an entangling two qubit gate, can be executed. Embedding the surface code in the 2.5-D architecture permits all of the same lattice surgery operations to be executed and error detection, although with some delay between each cycle as only one logical qubit can undergo a round of error correction at a time. Despite this additional delay, the thresholds of the embedded code are comparable to a standard surface code implementation.

Error correction protocols are essential for the execution of large-scale quantum programs. The surface code is one such code, designed with currently available architectures in mind. It is a low threshold code which requires only local operations on a 2-D grid.^{10,11} However, this application is better matched with this 2.5-D architecture, which allows qubits to be virtualized and stored in local memory. This highlights a

distinct role of the computer architect to discover fortuitous matches between application and emerging hardware technology. In this case, we match error correcting codes to an architecture which reduces resource requirements.

ARCHITECTURAL TRADEOFFS IN EMERGING TECHNOLOGY—NEUTRAL ATOMS

Current hardware implementations face unique scalability challenges, such as high gate error rates or high crosstalk error with densely connected qubits, or other fundamental challenges in controllability. While these devices have been useful as proof of concept demonstrations of small-scale near-term algorithms, it is unclear whether any of them in present form will be able to execute the large-scale computation needed for quantum speedup. These limitations are fundamental and current compilation approaches to reduce gate counts and depth are insufficient for long-term scalability.

Evaluating the viability of new hardware is essential.⁶ Architectural studies which fully explore their unique tradeoff spaces is key for finding the best way to accelerate beyond prototypes. One such alternative to superconducting qubits or trapped ions is neutral atoms.¹² These arrays of individually trapped atoms can be arranged in one, two, or even three dimensions.^{13–16}

This technology offers distinct advantages, which make it an appealing choice for scalable quantum computation. Mainly, atoms can interact at long distances, which leads to reduced communication overhead yielding lower gate count and depth programs. Furthermore, these devices may be able to execute high fidelity multiqubit (≥ 3 operands) operations natively. For example, a Toffoli gate can be implemented directly between three qubits without needing an expensive decomposition.¹⁷ These benefits do not come for free. Long range interaction requires proportionately large regions of qubits to be restricted leading to reduced parallelism requiring gates to be serialized with mitigating some of the reduced depth benefits. These unique properties are depicted in Figure 4.

Evaluating new quantum computing technology is especially challenging. Neutral atoms offer some clear advantages over other gate-based models, however, the current demonstrations display gate errors and coherence times which are worse than competitors. With physical properties lagging years behind, the appeal is dampened, but there is no fundamental

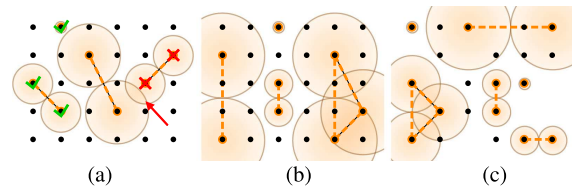


FIGURE 4. Examples of interactions on a neutral atom device. (a) Interactions of various distances are permitted up to a maximum. Gates can occur in parallel if their zones do not intersect. The interaction marked with green checks can occur in parallel with the middle interaction. (b) The maximum interaction distance specifies which physical qubits can interact. Compiler strategies suited for this variable distance are needed for neutral atom architectures. (c) Neutral atom systems are prone to sporadic atom loss. Efficient adaptation to this loss reduces computation overhead. Figure from Baker *et al.*⁶

limitation to these properties. When studied with all else being equal, the unique properties of neutral atom hardware claim a dominant position.

Exploring the use of new hardware also serves a critical role in the development of the platform itself. Neutral atom systems suffer a potentially crippling drawback—atoms can be lost both during and between program execution. When this happens, measurement of the qubits at the end of the run is incomplete and requires the output of the run to be discarded. The standard approach to coping with this loss is simply to run the program again after reloading all of the atoms in the array. Typically programs are run thousands of times and so each run which must be discarded incurs a twofold cost—we have to perform an array reload and we have to execute an additional run. This is especially bad when execution time is limited by atom reload rate, which is significantly longer than the actual program run.

Fortunately, software solutions can effectively mitigate this increased run time overhead. The idea is to simply keep track of when atoms are lost during the execution of a program, which can be detected quickly via fluorescence. If an atom loss occurs, we throw away that run and adjust how the program is compiled to the hardware. Ideally we do not want to recompile the entire program as this will be expensive, often more expensive than the reload itself, which should be treated as the worst case. Better solutions simply adjust the compiled program by adjusting the placements of the qubits and addition in a small number of extra communication operations.

THERE IS A LARGE GAP BETWEEN CURRENTLY AVAILABLE QUANTUM COMPUTING HARDWARE AND THE APPLICATIONS WE WANT TO RUN. HOWEVER, QUANTUM COMPUTING SYSTEMS HAVE MADE TREMENDOUS STRIDES IN RECENT YEARS IN A SIGNIFICANTLY COLLABORATIVE AND INTERDISCIPLINARY EFFORT BETWEEN PHYSICISTS, MATHEMATICIANS, COMPUTER SCIENTISTS, AND MANY MORE.

The best strategies directly take advantage of the neutral atom benefits. For example, these architectures support long range interactions, though most communication reduction does not require the use of the maximum allowed interaction distance. Therefore, by compiling the program to less than the maximum gives us flexibility in the final compiled program as atoms are lost over time requiring only a virtual remapping of program qubits and no extra operations. The best coping mechanisms sustain large numbers of loss by minimizing the total number of reloads, having low compilation time overhead, and adding a small number of gates (minimizing the number of error prone operations added).

Atom loss is a fundamental limitation of neutral atom architectures. Probabilistic loss of atoms is inherent in the trapping process itself and prior hardware studies have focused on *hardware* solutions to reduce this probability of loss. However, software solutions can effectively mitigate problems due to loss. Demonstrating effective mitigation strategies is critical to the overall development of the platform—by solving fundamental problems at the systems level with software, hardware developers can focus on solving and optimizing other problems. Codesign of quantum systems is key to accelerate the advancement of quantum computing technology.

CONCLUDING REMARKS

There is a large gap between currently available quantum computing hardware and the applications we want to run. However, quantum computing systems have made tremendous strides in recent years in a significantly collaborative and interdisciplinary effort between physicists, mathematicians, computer scientists, and

many more. Here we have discussed some of the central roles of the computer architect in the development of scalable hardware. First, we must determine which are the right abstractions for the job. Typically, quantum computation has been expressed exclusively in terms of two-level systems, however, this abstraction is artificial and *temporarily* making use of higher level qudit states yields efficient low depth circuit decompositions freeing up additional space on the hardware allowing us to execute larger programs sooner. New technologies which more easily allow access to these states can be powerful. Second, we must use applications to guide the design architectures, which best implement them. New, memory-equipped transmon technology has a fortuitous match with lattice surgery based surface codes, trading some serialization for an efficient error correction implementation. This architecture reduces hardware requirements enabling demonstrations of error correction sooner. Third, we must evaluate new technology as it is developed at the systems level to determine its viability for long-term scalability and develop software solutions to new technologies' fundamental limitations to guide hardware developers. For example, neutral atoms offer many unique advantages while also suffering from unique disadvantages and it is important to understand if architectures based on this technology are viable in either the near or long term. Furthermore, software can mitigate the downsides of atom loss, lending hardware developers the space to work on other aspects of the hardware. New technology inspired each of these ideas but the techniques developed here are general and could be applied to other similar technology as they emerge.

ACKNOWLEDGMENTS

This work was supported in part by EPIQC, an NSF Expedition in Computing, under Grant CCF-1730449; in part by STAQ under Grant NSF Phy-1818914; in part by DOE Grants DE-SC0020289 and DESC0020331; and in part by NSF OMA-2016136 and the Q-NEXT DOE NQI Center. Disclosure: F. Chong is Chief Scientist at Super.tech and an advisor to Quantum Circuits, Inc.

REFERENCES

1. L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. Annu. ACM Symp. Theory Comput.*, 1996, pp. 212–219.
2. P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.

3. J. M. Baker, C. Duckering, and F. T. Chong, "Efficient quantum circuit decompositions via intermediate qudits," in *Proc. IEEE 50th Int. Symp. Multiple-Valued Logic*, 2020, pp. 303–308.
4. P. Gokhale, J. M. Baker, C. Duckering, N. C. Brown, K. R. Brown, and F. T. Chong, "Asymptotic improvements to quantum circuits via qutrits," in *Proc. 46th Int. Symp. Comput. Archit.*, 2019, pp. 554–566.
5. J. M. Baker, C. Duckering, D. Schuster, and F. Chong, "Virtual logical qubits: A compact architecture for fault-tolerant quantum computing," *IEEE Micro*, vol. 41, no. 3, pp. 95–101, May/Jun. 2021.
6. J. M. Baker, A. Litteken, C. Duckering, H. Hoffmann, H. Bernien, and F. T. Chong, "Exploiting long-distance interactions and tolerating atom loss in neutral atom quantum architectures," in *Proc. ACM/IEEE Annu. Int. Symp. Comput. Archit.*, 2021.
7. A. Pavlidis and E. Floratos, "Arithmetic circuits for multilevel qudits based on quantum Fourier transform," 2017, *arXiv:1707.08834*.
8. C. T. Hann *et al.*, "Hardware-efficient quantum random access memory with hybrid quantum acoustic systems," 2019, *arXiv:1906.11340*.
9. R. Naik *et al.*, "Random access quantum information processors using multimode circuit quantum electrodynamics," *Nature Commun.*, vol. 8, no. 1, 2017, Art. no. 1904.
10. C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, "Surface code quantum computing by lattice surgery," *New J. Phys.*, vol. 14, no. 12, 2012, Art. no. 123011.
11. D. Litinski, "A game of surface codes: Large-scale quantum computing with lattice surgery," *Quantum*, vol. 3, p. 128, 2019.
12. M. Saffman, "Quantum computing with atomic qubits and Rydberg interactions: Progress and challenges," *J. Phys. B, At., Mol. Opt. Phys.*, vol. 49, no. 20, 2016, Art. no. 202001.
13. M. Endres *et al.*, "Atom-by-atom assembly of defect-free one-dimensional cold atom arrays," *Science*, vol. 354, no. 6315, pp. 1024–1027, 2016.
14. D. Barredo, S. De Léséleuc, V. Lienhard, T. Lahaye, and A. Browaeys, "An atom-by-atom assembler of defect-free arbitrary two-dimensional atomic arrays," *Science*, vol. 354, no. 6315, pp. 1021–1023, 2016.
15. H. Kim, W. Lee, H.-G. Lee, H. Jo, Y. Song, and J. Ahn, "In situ single-atom array synthesis using dynamic holographic optical tweezers," *Nature Commun.*, vol. 7, no. 1, pp. 1–8, 2016.
16. D. Barredo, V. Lienhard, S. De Leseleuc, T. Lahaye, and A. Browaeys, "Synthetic three-dimensional atomic structures assembled atom by atom," *Nature*, vol. 561, no. 7721, pp. 79–82, 2018.
17. H. Levine *et al.*, "Parallel implementation of high-fidelity multiqubit gates with neutral atoms," *Phys. Rev. Lett.*, vol. 123, no. 17, 2019, Art. no. 170503.

JONATHAN M. BAKER is currently working toward a Ph.D. degree in computer science at the University of Chicago, Chicago, IL, USA. He is currently interested in many interdisciplinary aspects in the design and evaluation of quantum computer architectures, spanning the software-hardware stack from application to device control. Baker has degrees in computer science, chemistry, and mathematics from the University of Notre Dame, Notre Dame, IN, USA. Contact him at jmbaker@uchicago.edu.

FREDERIC T. CHONG is the Seymour Goodman Professor with the Department of Computer Science, University of Chicago, Chicago, IL, USA and the Chief Scientist at Super.tech, Chicago, IL, USA. He is also Lead Principal Investigator for the EPIQC Project (Enabling Practical-scale Quantum Computing), an NSF Expedition in computing. He was a faculty member and Chancellor's Fellow at the University of California Davis in 1997–2005. He was also a Professor of Computer Science, Director of Computer Engineering, and Director of the Greenscale Center for Energy-Efficient Computing at the University of California Santa Barbara from 2005 to 2015. Chong received a Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1996. He is a recipient of the NSF CAREER Award, the Intel Outstanding Researcher Award, and 10 best paper awards. Contact him at chong@cs.uchicago.edu.