Virtual Logical Qubits: A Compact Architecture for Fault-Tolerant Quantum Computing

Jonathan M. Baker, Casey Duckering, David I. Schuster, and Frederic T. Chong, *University of Chicago, Chicago, IL, 60637, USA*

Fault-tolerant quantum computing is required to execute many of the most promising quantum applications. In recent years, numerous error correcting codes, such as the surface code, have emerged which are well suited for current and future limited connectivity 2-D devices. We find quantum memory, particularly resonant cavities with transmon qubits arranged in a 2.5-D architecture, can efficiently implement surface codes with around 20× fewer transmons via this work. We virtualize 2-D memory addresses by storing the code in layers of qubit memories connected to each transmon. Distributing logical qubits across many memories has minimal impact on fault tolerance and results in substantially more efficient logical operations. Virtualized logical qubit (VLQ) systems can achieve fault tolerance comparable to conventional 2-D transmon-only architectures while putting within reach a proof-of-concept experimental demonstration of around ten logical qubits, requiring only 11 transmons and 9 attached cavities.

Quantum devices have improved significantly in the last several years both in terms of physical error rates and number of usable quantum bits (qubits). Concurrently, great progress has been made at the software level such as improved compilation procedures reducing required overhead for program execution. These efforts are directed at enabling noisy intermediate-scale quantum (NISQ)[1] algorithms to demonstrate the power of quantum computing and are expected to run some important programs.

Despite this, these machines will be too small for error correction and unable to run large-scale programs due to unreliable qubits. The ultimate goal is to construct fault-tolerant machines capable of executing thousands of gates and in the long term to execute large-scale algorithms with speed-ups over classical algorithms. There are a number of promising error correction schemes which have been proposed such as the surface code.[2,3] The surface code is a particularly appealing candidate because of its low overhead, high error threshold, and its reliance on few nearest-neighbor interactions in a 2-D array of qubits, a common feature of currently popular hardware like superconducting transmon qubits.

Current architectures for both NISQ and fault-tolerant quantum computers make no distinction between the memory and processing of quantum information. While currently viable, as larger devices are built, the engineering challenges of scaling to hundreds of qubits become readily apparent. For transmon technology, some of these issues include fabrication consistency and crosstalk during parallel operations. Every qubit needs dedicated control wires and signal generators, which fill the refrigerator the device runs in. To scale to the millions of qubits needed for useful fault-tolerant machines,[4] we need to adopt a memory-based architecture to decouple qubit count from transmon count.

We use a recently realized qubit memory technology, which stores qubits in a superconducting cavity.[5] Stored in cavity, qubits have a significantly longer lifetime (coherence time), but must be loaded into a transmon for computation. We design and evaluate a system-level organization of these components within

0272-1732 © 2021 IEEE
Digital Object Identifier 10.1109/MM.2021.3072789
Date of publication 13 April 2021; date of current version 25 May 2021.

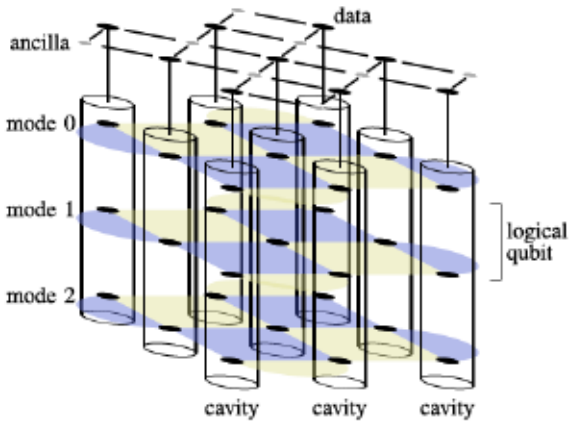

FIGURE 1. Our fault-tolerant architecture with random-access memory local to each transmon. On top is the typical 2-D grid of transmon qubits. Attached below each data transmon is a resonant cavity storing error-prone data qubits (shown as black circles). This pattern is tiled in 2-D to obtain a 2.5-D array of logical qubits. Our key innovation here is storing the qubits that make up each logical qubit (shown as checkerboards) spread across many cavities to enable efficient computation.

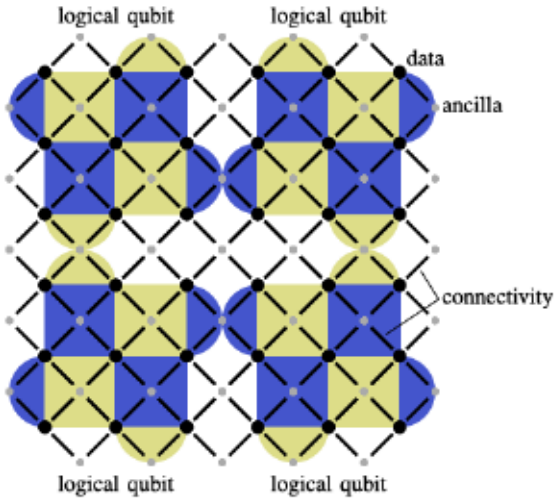

FIGURE 2. Typical 2-D superconducting qubit architecture. The dots are transmon qubits where black are used as data and gray are used as ancilla for error correction. The lines indicate physical connections between qubits that allow operations between them. Four logical qubits, each consisting of nine error-prone data qubits are shown here in the rotated surface code with distance 3. Z parity checks are shaded yellow (light) and X parity checks are shaded blue (dark) where checks on only two data are drawn as half circles.

the context of a novel surface code embedding and fault-tolerant quantum operations.

Our proposed 2.5-D memory-based design is a typical 2-D grid of transmons with memory added, Figure 1. This can be compared with the traditional 2-D error correction implementation in Figure 2, where the checkerboards represent error-corrected logical qubits. The logical qubits in this system are stored at unique virtual addresses in memory cavities when not in use. They are loaded to a physical address in the transmons and made accessible for computation on request and are periodically loaded to correct errors, similar to DRAM refresh. This design allows for more efficient operations such as the transversal CNOT between logical qubits sharing the same physical address, i.e., colocated in the same cavities. This is not possible on the surface code in 2-D, which requires methods such as braiding or lattice surgery for a CNOT operation.

We develop an embedding from the standard representation to this new architecture, which reduces the required number of physical transmon qubits by a factor of approximately k, the number of resonant modes per cavity which is expected to be at least 10 and is expected to improve over time. We also develop a Compact variant saving an additional $2\times$. This means we can obtain a code distance $\sqrt{2k}$ times greater or use hardware with only $1/(2k)$ the required physical transmons for a given algorithm. In the near-to-intermediate term, when qubits are a highly constrained resource, this will accelerate a path toward fault-tolerant computation. In fact, the smallest instance of Compact requires only 11 transmons and 9 cavities for about ten logical qubits. Via simulation, we determine the error correction threshold rates for each and find they are all close to the baseline threshold meaning the additional error sources do not significantly impact the performance.

BACKGROUND

Superconducting Qubit Architectures

In contrast to other leading qubit technologies such as trapped ion devices with one or more fully connected qubit chains, superconducting qubits are typically connected in nearest neighbor topologies, often a 2-D mesh on a regular square grid. This limitation makes engineering these devices easier but results in high communication costs, increasing the chance of errors on NISQ devices, and communication congestion for error corrected operations. More background on superconducting hardware can be found by Krantz *et al.*[6]

Qubit Memory Technology

Recently, studies have demonstrated random access memory for quantum information.[5] Qubit states can be stored in the resonant modes of physical superconducting cavities attached to a transmon qubit and depicted as the individual cylinders in Figure 1. Currently demonstrated error rates are promising, and there is nothing fundamental preventing this technology from becoming competitive with other transmon devices as it matures. We expect operation error rates to improve, cavity sizes and coherence times to increase and in general expect performance to improve as it has with other quantum technologies.

Local memory is not free. Stored qubits cannot be operated directly. Instead, operations are mediated through the transmon. To operate on qubits stored in memory, we first load the qubit from memory. Then, we perform the desired operation on the transmons, and store the qubit back in its original location. A two-qubit operation such as a CNOT can also be performed directly between the transmon and a qubit in its connected cavity by manipulating higher states of the transmon. Qubits stored in the same cavity cannot be operated on in parallel. There are two primary benefits of this technology. First, we are able to quickly perform two-qubit interactions between any pair of qubits stored in the same cavity. Second, qubits stored in the cavity are expected to have longer coherence times by about one order of magnitude.

Surface Codes

The surface code[2] is one of the most promising quantum error correction protocols because it requires only nearest neighbor connectivity between physical qubits and improvements continue to be made.[7] The surface code is implemented on a 2-D array of physical qubits shown in Figure 2. These qubits are either data, where the state of the logical qubit is stored, or ancilla used for syndrome extraction (parity checks). These ancilla qubits are measured to stabilize the entangled state of the data. These ancilla fall into two categories, measure-Z and measure-X for Z syndromes and X syndromes designed to detect bit and phase errors, respectively.

Each X (Z) plaquette corresponds to a single measure-X (Z) qubit and the four data, which it interacts with. The corners of each plaquette are the data qubits. For the baseline, we use standard Z and X syndrome extraction (parity measurement) circuits where the qubits of this circuit are physical qubits. The Z-syndrome measures the bit-parity of its corner qubits and the X-syndrome measures their phase parity. By repeatedly performing syndrome extraction and detecting parity changes, we are able to locate errors. This repeated syndrome extraction collapses any error to a correctable Pauli error and forces the data to remain in what is called the code state. We may detect errors which occur as changes in measurement outcomes of the parity checks.

There are two primary ways to manipulate the logical qubits of the surface code to perform desired logical operations–braiding and lattice surgery. In this article, we will primarily consider lattice surgery, which has been shown to have some advantages over braiding like using fewer physical qubits. For a more thorough introduction to lattice surgery, we refer the reader to Horsman *et al.*[3,8,9] In our proposed scheme, all primitive lattice surgery operations can be used such as split and merge, which together perform a logical CNOT. For universal quantum computation in surface codes, we allow for the creation and use of magic states such as $|T\rangle$ or $|CCZ\rangle$.

VIRTUALIZED LOGICAL QUBITS

Our proposed architecture is an embedding of the surface code, which virtualizes logical qubits, saving in required number of transmons. This takes advantage of quantum resonant cavity memory technology to store *logical* qubits, in the form of surface code patches, in memory local to the computational transmons.

Natural Surface Code Embedding

Our embedding slices the plane of surface code tiles into many pieces, storing them flat in memory to enable them to stitch together on-demand. This embedding enables the fast transversal CNOT and high connectivity.

For every transmon in this architecture (the compute qubits in the top layer of Figure 1), there is a cavity attached with a fixed number of resonant modes k. Each cavity can store k qubits, one per mode. Each transmon can load and store qubits from its attached cavity. All transmons can be operated on in parallel as is the case in most superconducting hardware. We expect this technology to allow cavity size k on the order of 10 to 100 qubits.

Consider the rotated surface code of Figure 2 and the high level view of this architecture in Figure 1. We map each of the physical qubits of this logical qubit to the same mode z of each cavity in this memory architecture. Another logical qubit can be mapped to a different mode of the same set of cavities. We view this as stacking the surface code patches, the logical

qubits, together under the same set of transmon qubits. The transmons themselves are only used for logical operations and error correction cycles performed on the patches.

In this memory architecture, we are unable to operate on qubits stored in the same cavity in parallel, however we are permitted to operate on qubits stored in different cavities in parallel. In order to detect measurement errors, we require d, the distance of the code, rounds of syndrome extraction before we perform our decoding algorithm and correct errors. We can load a logical qubit (meaning load all data in parallel to each transmon), perform all d rounds of extraction, then store the qubit, this is our All-at-once strategy. Alternatively, we can interleave the extraction cycles by loading the logical qubit in index 0, performing one syndrome extraction step, then storing. We execute this same procedure for every logical qubit in the stack and repeat d times.

Up to k logical qubits share the same set of transmons thereby more efficiently storing these qubits than on a single large surface. To interact logical qubits in different stacks, we load them in parallel to the transmons then interact them via lattice surgery operation. In these cases, all of the other stacks' transmons between the interacting logical qubits act as a single logical ancilla. Furthermore, physical operations between qubits in the same cavity enable our system to perform fast transversal two-qubit interactions if the logical qubits are colocated in the same stack.

Transversal CNOT

A major advantage of this 2.5-D architecture is the ability to do two-qubit operations transversely using the third dimension. The logical operation is performed directly by doing the same physical gate to every data qubit and correcting any resulting errors. For typical 2-D error correcting codes like the surface code, transversal two-qubits operations are not possible because the corresponding data qubits of two logical patches cannot be made adjacent. However, with memory, it is possible to load one patch into the transmons and apply two-qubit gates mediated by each transmon onto the data qubits for a second qubit stored in one mode of the cavities. The transversal CNOT can be performed in a single round of d error correction cycles while the lattice surgery CNOT takes six rounds. This can translate to major savings in runtime for algorithms.

The transversal CNOT is not limited to logical qubits currently stored in the same 2-D address. With an extra step, it is possible to transversely interact any two logical qubits. To do this, one of the qubits must be *moved* to the same 2-D address as the other using a move operation described by Litinski.[8] Once the two qubits are in the same 2-D address, the transversal CNOT can be applied.

Compact Surface Code Embedding

In the previous scheme, half of the transmons did not have attached cavities (or they did not make use of them). An ancilla and data qubit could share a transmon because the data are stored in the cavity the majority of the time and the ancilla are reset every cycle. This leads to a more efficient, Compact embedding which halves the required number of transmons. This comes at the cost of additional loads and stores from memory due to contention during error correction, effectively trading some error and time for significant space savings.

This mapping results in plaquettes, which resemble triangles rather than squares, where the center of the hypotenuse of each triangle corresponds to both the ancilla qubit and the data qubit, stored "beneath" in its cavity. Every data qubit is still mapped to the *same* index. We illustrate this transformation from our undistorted Natural surface code patch to Compact in Figure 3.

This new mapping also requires a new syndrome extraction procedure because data cannot be loaded while a transmon is in use as an ancilla. A single round of syndrome extraction can be executed by dividing the plaquettes into four groups, with each group containing noninterfering plaquettes. Two plaquettes are noninterfering if they do not share their ancilla with any data qubits of the other plaquette. It is imperative this process use both the minimum number of loads and stores and keep data qubits loaded for as short a time as possible as the error incurred during this circuit directly impacts the error threshold for the code. Error correction can be performed interleaved or all-at-once just as with natural.

Beyond the Surface Code

The surface code is an appealing choice for currently available superconducting devices because of its relatively low overhead and it requires only limited nearest neighbor connectivity. For this new memory-based architecture, there is a fortuitous match with the surface code, making it an even more appealing candidate. However, there are many other candidate error correction codes such as the color code.

One particularly relevant class of codes for this architecture are Bosonic codes.[10] The surface and

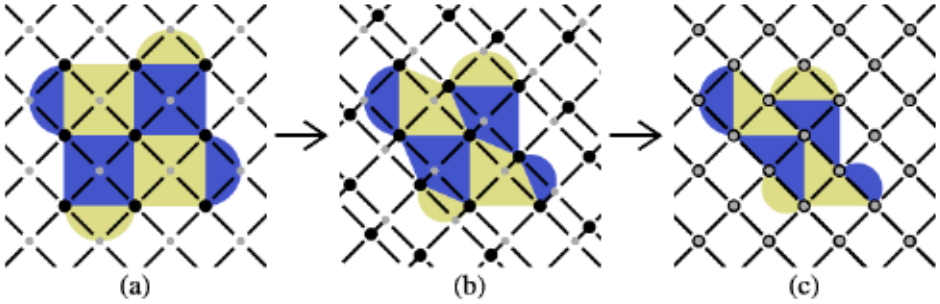

FIGURE 3. Transformation from natural to compact. (a) Natural embedding: Only data have attached cavities (not shown). (b) The transformation: Z ancilla (over yellow/light areas) merge with the upper-right data transmon and X ancilla (over blue/dark areas) merge with the lower left data transmon. The opposite parings are key to keeping 4-way grid connectivity. (c) Compact embedding: All ancilla transmons without attached cavities have been removed. All remaining transmons have cavities and are used as both data and ancilla.

color codes protect quantum information by using many physical qubits to construct a single logical qubit. For Bosonic codes, instead we can create redundancy by using many modes of a single physical system. The modes of the cavities in the underlying hardware of our systems can be used to implement Bosonic codes directly. Unfortunately, Bosonic codes in practice only approximately correct errors and do not have the property that as you scale the size of the code you can obtain an exponential reduction in the logical error rate of the system. Bosonic codes can be used effectively for error mitigation.

It is unclear, given an architecture, what the best error correction scheme is. Ideally, we want a code which takes full advantage of the high connectivity between information stored in the same cavity. For practical demonstrations, we also want a code which requires a small number of transmons. Bosonic codes are viable options for this architecture and have a somewhat natural fit. It is yet to be seen what is the best choice, however we have shown the surface code is a fortuitous match with this new memory-based architecture.

EVALUATION

Error Threshold Results

We detail our threshold results in Figure 4. We study five different code distances in order to obtain the physical error threshold value. The threshold value indicates at which point increasing the code distance, d improves the logical error rate instead of hurting it. This threshold is a function of both the physical system model, the chosen syndrome extraction circuit, and the specific decoding procedure. The major difference in each procedure is the additional error sources and different syndrome extraction procedures. The slopes for each code distance compared across the various schemes is stable, indicating each scheme improves at a similar rate, post error threshold, and

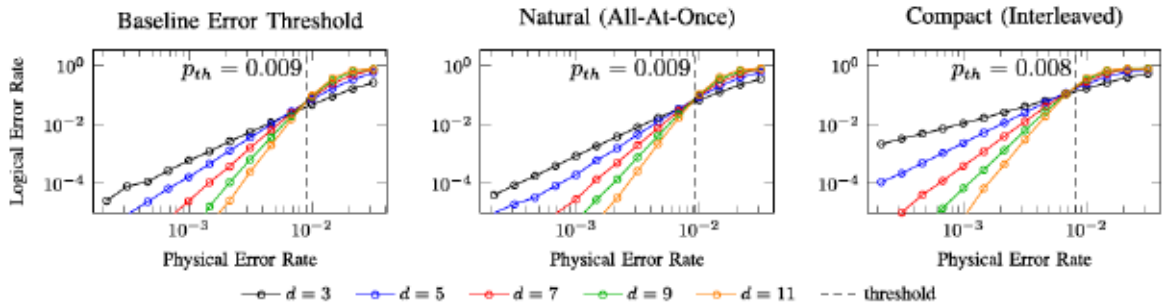

FIGURE 4. Error thresholds for the baseline 2-D architecture and natural and compact variants of our 2.5-D architecture. The thresholds are comparable to the baseline indicating the space savings obtained in our system does not substantially reduce the error thresholds. The slopes of the lines in this figure indicate, postthreshold, how much improvement in physical error rates improve logical error rate. Except for the baseline, all use a cavity size of 10.

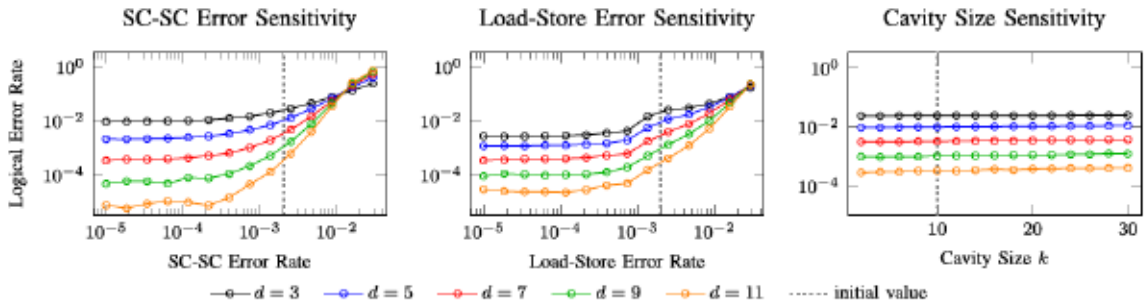

FIGURE 5. Sensitivity of logical error rate to various error sources in compact (Interleaved). The logical error rates are most sensitive to physical error of loads/stores and SC-SC (transmon–transmon) gates. The logical error rate is less sensitive to transmon and cavity coherence times (not shown) and mostly insensitive to effects of cavity size.

showing that the logical error rate decays exponentially with d as desired. This is significant because it means we will be able to save on total number of transmons without major degradation of the error threshold.

Error Sensitivity Results

Different system-level details affect the threshold of the code. Here we focus on compact, interleaved as the most efficient physical qubit mapping and subject to a wide variety of errors. The results of these sensitivity studies are found in Figure 5. The logical error rate is sensitive to a particular error source's probability if the slope of the line is pronounced at the marked reference value. The logical error rate for compact, interleaved is sensitive to all changes in system-level details to some degree. The gate error rates show the highest sensitivity, indicating improvement in these will give the greatest benefit. Coherence times (plots not shown) are not quite as sensitive but the slightly over 10× offset between the cavity and transmon plots shows that there is no benefit in transmon T_1 being longer than 1/10 cavity T_1 when the cavity size is 10. The lines taper off, indicating other errors sources eventually dominate. Initially, we expected the cavity size to have a large impact on the logical error rate. However, when coherence times are high and gate error rates are fairly low below the threshold, the logical error rate does increase proportional to the length of the cavity but the effect is very minor. Given cavities with good coherence times, this indicates our proposed system will be able to scale smoothly into the future as cavity sizes increase.

While larger cavity sizes will make this architecture even more advantageous, there will be a point at which it has a vanishing benefit because the delay between error correction becomes too long and decoherence error dominates. For the error rates used in the evaluation, we find that cavity decoherence error starts dominating after cavity size $k \approx 150$. After this point, it would be more beneficial to improve the cavity coherence time.

CONCLUSION

Current NISQ machines are powerful demonstrations, but fall short of many serious applications without error correction. This article makes an error-corrected machine 20× easier to build by exploiting quantum memory with a codesigned architecture to enable medium-scale quantum machines (100–1000 transmons) and allow the industry to realize the long-term potential of scalable quantum machines.

Current quantum computers are noisy and they are incapable of running sizable programs accurately. There is currently a major gap between what is available on the market and what is required to execute the famed quantum algorithms with quantum error correction. Currently, approaches toward bridging this gap fall into a few categories: either push the error rates of current devices down below the error threshold of known codes, or design new codes. Our work bridges this gap in a completely different way by exploring the use of new technology, resonant cavities, for the design of new architectures which better support error correction codes already available.

Locally accessible quantum memory can be used to create a 2.5-D architecture better suited for the code than traditional approaches. Our architecture enables the execution of the surface codes with lattice surgery operations with significantly lower physical requirements resulting in higher distance codes with fewer total transmon qubits. What does this mean? Given a fixed physical error rate below the threshold, we can use larger codes to obtain strictly

better logical error rates. This enables the execution of longer input programs. Conversely, for a fixed desired logical error rate, determined by the application, we can run on hardware with worse error rates which will be available years sooner. This demonstrates the benefit of codesigning quantum architectures alongside the applications and technologies.

REFERENCES

1. J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2 , 2018, Art. no. 79.
2. A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, no. 3, 2012, Art. no. 032324.
3. 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.
4. C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, p. 433, Apr. 2021.
5. R. Naik *et al.*, "Random access quantum information processors using multimode circuit quantum electrodynamics," *Nature Commun.*, vol. 8, no. 1, 2017, Art. no. 1904.
6. P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Appl. Phys. Rev.*, vol. 6, no. 2, 2019, Art. no. 021318.
7. J. P. Bonilla-Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, "The XZZX surface code," 2020, *arXiv:2009.07851*.
8. D. Litinski, "A game of surface codes: Large-scale quantum computing with lattice surgery," *Quantum*, vol. 3, 2019, Art. no. 128.
9. L. Lao *et al.*, "Mapping of lattice surgery-based quantum circuits on surface code architectures," *Quantum Sci. Technol.*, vol. 4, no. 1, 2018, Art. no. 015005.
10. W. Cai, Y. Ma, W. Wang, C.-L. Zou, and L. Sun, "Bosonic quantum error correction codes in superconducting quantum circuits," *Fundam. Res.*, vol. 1, pp. 50–67, 2021.

JONATHAN M. BAKER is currently working toward a Ph.D. degree with the University of Chicago, IL, USA. His research is primarily focused on vertical integration of the quantum computing hardware-software stack. Contact him at jmbaker@uchicago.edu.

CASEY DUCKERING is currently working toward a Ph.D. degree with the University of Chicago, Chicago, IL, USA, aiming to efficiently bring together quantum algorithms and error correction with their physical implementations on quantum computers. Contact him at cduck@uchicago.edu.

DAVID I. SCHUSTER is an Associate Professor with the Department of Physics, University of Chicago, Chicago, IL, USA. His work is currently centered on superconducting quantum circuits to make quantum memories, perform error correction, and create topologically protected qubits. Schuster received a Ph.D. degree from Yale University in 2007. Contact him at david.schuster@uchicago.edu.

FREDERIC T. CHONG is the Seymour Goodman Professor with the Department of Computer Science, University of Chicago, 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 UC Davis from 1997 to 2005. He was also a Professor of Computer Science, the Director of Computer Engineering, and the Director of the Greenscale Center for Energy-Efficient Computing at UCSB from 2005 to 2015. Chong received a Ph.D. degree from MIT in 1996. He is a recipient of the NSF CAREER Award, the Intel Outstanding Researcher Award, and ten best paper awards. He currently serves on the National Quantum Initiative Advisory Committee and is the Chief Scientist and Co-Founder of Super.tech, a quantum software company. Contact him at chong@cs.uchicago.edu.