Optimized Surface Code Communication in Superconducting Quantum Computers

Ali Javadi-Abhari
Princeton University
ajavadia@princeton.edu

Pranav Gokhale
University of Chicago
pranavgokhale@uchicago.edu

Adam Holmes
University of Chicago
adholmes@uchicago.edu

Diana Franklin
University of Chicago
dmfranklin@cs.uchicago.edu

Kenneth R. Brown
Georgia Institute of Technology
ken.brown@chemistry.gatech.edu

Margaret Martonosi
Princeton University
mrm@princeton.edu

Frederic T. Chong
University of Chicago
chong@cs.uchicago.edu

ABSTRACT

Quantum computing (QC) is at the cusp of a revolution. Machines with 100 quantum bits (qubits) are anticipated to be operational by 2020 [30, 73], and several-hundred-qubit machines are around the corner. Machines of this scale have the capacity to demonstrate *quantum supremacy*, the tipping point where QC is faster than the fastest classical alternative for a particular problem. Because error correction techniques will be central to QC and will be the most expensive component of quantum computation, choosing the lowest-overhead error correction scheme is critical to overall QC success. This paper evaluates two established quantum error correction codes—planar and double-defect surface codes—using a set of compilation, scheduling and network simulation tools. In considering scalable methods for optimizing both codes, we do so in the context of a full microarchitectural and compiler analysis. Contrary to previous predictions, we find that the simpler planar codes are sometimes more favorable for implementation on superconducting quantum computers, especially under conditions of high communication congestion.

CCS CONCEPTS

• **Hardware** → **Quantum computation**; **Quantum error correction and fault tolerance**;

KEYWORDS

Quantum Computing, ECC, Design-Space Exploration

MICRO-50, October 14–18, 2017, Cambridge, MA, USA

ACM ISBN 978-1-4503-4952-9/17/10. . . $15.00
https://doi.org/10.1145/3123939.3123949

ACM Reference format:
Ali Javadi-Abhari, Pranav Gokhale, Adam Holmes, Diana Franklin, Kenneth R. Brown, Margaret Martonosi, and Frederic T. Chong. 2017. Optimized Surface Code Communication in Superconducting Quantum Computers. In *Proceedings of MICRO-50, Cambridge, MA, USA, October 14–18, 2017,* 14 pages.
https://doi.org/10.1145/3123939.3123949

1 INTRODUCTION

Major academic and industry efforts are underway to build scalable quantum computers, which can have significant applications in solving currently intractable problems in areas as diverse as AI [33], medicine [47] and security [72]. They will also profoundly impact our understanding of the nature of computation itself [1].

Owing to considerable theoretical successes over the past two decades [3, 27, 31, 45] and successful small-scale physical demonstrations [43, 76], the main challenges now relate to scaling. This includes building more and longer-lived qubits, while simultaneously reducing the number of qubits and time required to run a given application. This paper focuses on the latter. We propose optimizations that reduce the number of qubits and clock cycles (the space-time resource usage), especially due to qubit communication requirements. We then discuss criteria that should be considered in choosing the lowest-overhead error correcting code.

Our focus on quantum error correction (QEC) stems from its importance in large-scale QC. Although it is expected that computers with 50-100 qubits will be built in a few years (and surpass classical computing capabilities due to exponentially increasing information state space), they will likely not sustain lengthy computations. This is due to the fragility of quantum states and the imprecisions of quantum control. Thus, in the long term, QEC is a necessity. However, QEC is the most resource-consuming component of QC, predicted to utilize up to 90% of qubits in the system [39, 41]. Any design should therefore aggressively optimize QEC, and also choose the best code among different possible schemes [18, 27, 46]. This paper hints at how this may be achieved.

In this paper, we study the most promising proposal for building large-scale quantum computers: *surface code* QEC on *superconducting* quantum technology [22]. These are being targeted in most industry efforts [30, 67, 73]. Superconducting technology relies on qubits created from electrical circuits cooled to very low temperatures. They have the advantage of being both low error and fast. Surface codes are a family of codes that protect information using a simple 2-dimensional layout of redundant qubits with desirable scaling properties and high error resilience. Our study quantitatively compares two variations of the surface code in the context of a comprehensive microarchitectural and compiler analysis. Considering issues like locality, concurrency, and congestion, our evaluation is informed by classical microarchitectural techniques, while bringing quantitative clarity to the most important QC design decisions.

The key challenge of this paper is to frame quantum computing problems, which look very different from traditional architecture problems, in a manner that allows us to apply computer architecture techniques. This application-system-technology co-design is crucial if the scarce quantum resources are to be used effectively. In particular:

(1) Surface code error correction has traditionally involved statically scheduling "braids" by hand in a 3D space-time volume for very small quantum circuits [19, 26]. Leveraging the property that program inputs are often fully-determined in QC applications, we map the problem from a set of 3D topological transformations to a 2D static routing problem, over-constraining it to allow scalability to very large circuits. We show that using proper heuristics, this approximation can still achieve near-optimal performance.
(2) Surface code "braids" have been greatly favored in the quantum computing community because they appear latency insensitive (an entire braid can be implemented in 1 cycle, regardless of length). Yet we find that highly parallel quantum programs scale poorly because simultaneous braids can neither cross nor be prefetched, causing a form of contention scaling not previously analyzed. We show that in such cases, "teleportation-based" communication is more desirable as it is prefetchable. (Quantum resources can be pre-communicated, allowing any contention to be diffused over time prior to the point of use.)
(3) We incorporate (1) and (2) into an extensive software toolchain that performs end-to-end synthesis from high-level quantum programs to physical layout and circuits. This automated framework allows us to map the design space parameterized by application characteristics, error correction scheme, and device reliability. We show the importance of (2) by plotting optimal space-time design points as device reliability evolves with better technology.

The rest of this paper is organized as follows: Section 2 is an overview of the relevant background, and Section 3 discusses related prior work. Sections 4 and 5 discuss the micro-architectural details of various QEC implementations, and the software stack designed for evaluations of the design space. Section 6 discusses optimizations on braid-based communications, and Section 7 shows our results. Section 8 offers broader discussions including options for other communication optimizations, and Section 9 concludes the paper.

2 BACKGROUND

This section presents a brief overview of the relevant background on quantum computation, theory of error correction, and physical technologies.

2.1 Principles of Quantum Computation

Quantum bits (*qubits*) are the principal units of information in quantum computers. Unlike classical bits, their states are not confined to two deterministic binary values. Instead, a generic quantum state exists in a probabilistic yet simultaneous superposition of the $|0\rangle$ and $|1\rangle$ states. Manipulation of probabilities is equivalent to changing the state, and can be achieved through the application of various quantum *operations*. When observed (measured), a qubit collapses into a classical binary state (either 0 or 1), in accordance with the prior probabilities of the quantum state. Expressed in algebraic terms, quantum states are unit vectors in the Hilbert space. Quantum operations rotate these vectors, while quantum measurements project them onto a lower dimension.

While this means that there are theoretically infinite possible quantum states, and infinite valid operations to transition between those states (any small degree of rotation is permissible), it can be proven that a small set of operations is sufficient to approximate all possible operations [4, 44]. These *universal operations* are akin to a classical instruction set.

The power of quantum computation comes from its exponentially increasing state space: n qubits are represented by a 2^n-dimensional state vector. Furthermore, quantum states can interfere, owing to their dual nature as particles and waves, further allowing manipulations that surpass the power of classical computers. The key idea behind designing quantum applications is to initialize qubit states to encode a given problem, orchestrate interferences to eliminate undesired answers from the state space, and finally measure the qubits, obtaining (with high probability) the answer to the query.

2.2 Theory of Quantum Error Correction

When a quantum application is designed, it assumes noise-free computation. In practice, however, the computation can deviate from the intended path due to noise, interaction of qubits with the environment, and imprecise operations. Fortunately, if the noise level is within specific bounds, this deviation can be corrected through constant monitoring and error detection [3].

As with any type of error resilience, redundant information is used to protect fragile states. Although we cannot simply copy the state of one qubit into multiple qubits and later take a majority vote (due to quantum no-cloning constraints [82]),

we can still use redundant qubits to create a code space where valid code words are safely separated, immune to noise perturbations. The idea is similar to classical error correction: more redundancy will result in a larger *code distance (d)*, increasing error tolerance. Small corruptions of data would place the data's state into an orthogonal error subspace, which can be immediately detected and corrected. The higher-level encoded quantum state is called the *logical qubit*, while its comprising fundamental qubits (including redundant bits) are called *physical qubits*.

One complicating factor in checking logical qubits for errors is that we cannot directly observe their comprising physical qubits—observing a qubit collapses its state superposition, yielding a simple classical bit. This necessitates the use of extra "helper" qubits, known as *ancillas*. Ancillas are interacted with data for *syndrome measurement*—learning just enough about the encoded block to detect and fix errors without measuring the full state of the encoded block. Even though quantum states are analog, and errors can occur in arbitrary amounts, syndrome measurement has the effect of projecting the qubit state onto the basis vectors of the error subspace, thereby quantizing errors. Such quantized errors can then be corrected using simple corrective operations: a bit-flip operation, a phase-flip operation, or both.

After encoding, any computation must be performed directly on encoded data, since decoding and encoding again would be too error prone. Performing encoded versions of some operations are harder than others. In particular, most proposals for performing the T operation require the use of even more extra qubits. The extra qubits must first be prepared into one specially encoded quantum state (called "magic state" due to its distinctive properties [9, 29, 54]), and then interacted with the original data to perform the T operation.

Unfortunately, the work required to achieve proper error resilience is quite costly, since there is a wide gap between reliability requirements of non-trivial quantum applications and reliability rates supported by quantum technologies (called *logical error rates* (p_L) and *physical error rates* (p_P) respectively). It is typical to have $10-15$ orders of magnitude difference between p_L and p_P. The application imposes a far more stringent error-per-operation expectation than the technology can support. p_L and p_P are dictated by the *size of computation* (i.e. total number of pre-QEC logical operations that must be executed) and the noise level in the physical technology, respectively. Longer computation has a higher chance of corruption, thus requiring more reliable operations. For example, an application that executes a total of 10^{12} logical operations must ensure that per-operation errors do not exceed 0.5×10^{-12}, if it wants to accurately perform its computation at least half the times it is executed. 50% is a typical correctness target, which we also assume. On the other hand, current superconducting technology is able to perform operations with reliabilities of $99.99-99.999\%$—equivalent to physical error rates of $10^{-2}-10^{-3}$. A main goal of this paper is to optimize QEC in a way that we do not lose more resources to it than necessary. In addition, we also suggest the least resource-intensive QEC code for a range of logical-to-physical error gaps, as this parameter looks very different for various applications and technologies.

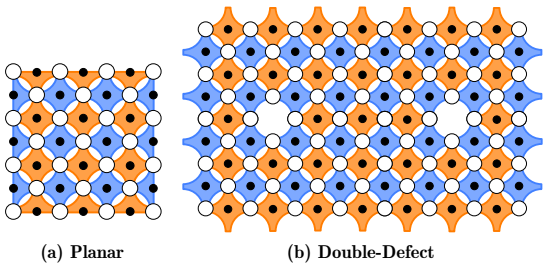


Figure 1: One encoded logical qubit, using the (a) planar and (b) double-defect surface code variations. Ancilla qubits (small black) are used to continuously monitor data qubits (large white). The planar encoding uses fewer physical qubits for the same encoding strength.

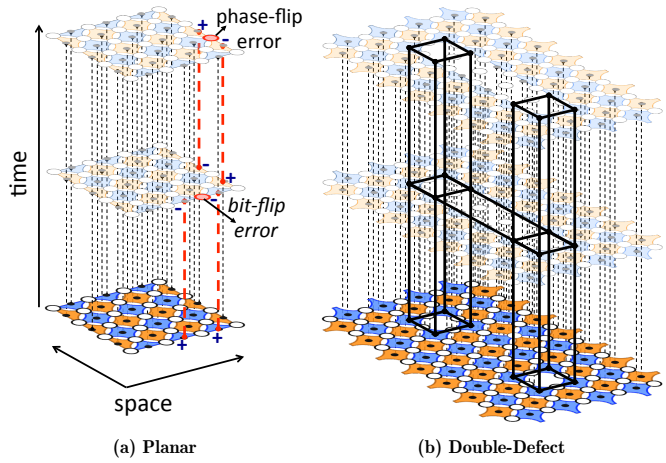
2.3 Surface Codes

Surface codes are a family of promising, low-overhead QEC codes [27], where logical qubit information is encoded in the topology of a two-dimensional lattice of physical qubits [64]. A surface code lattice alternates data and ancilla qubits (white and black circles in Figure 1). Data qubits collectively encode the logical state, while ancillas continuously collect syndrome measurements from their neighboring data. It can be shown that the lattice as a whole acts as an encoded qubit. A larger lattice is a stronger encoding (larger code distance d).

Intuitively, surface codes are so fault tolerant because they employ a "soft decoding" scheme in which error syndromes are measured and recorded over an extended time period, and then a minimum-weight perfect-matching algorithm [25] is used to identify errors after the fact (and then the errors are post-corrected due to commutativity of the correction operations).

2.3.1 Planar vs. Double-Defect Encodings. In this paper we investigate the two main flavors of surface codes: *planar* [10, 18] and *double-defect* [27] encodings. In the planar encoding (Figure 1a), a single lattice (or "plane") represents a single logical qubit. To introduce extra logical qubits to the system, it suffices to build separate lattices. In this scenario, the system can be seen as a collection of planar tiles. In contrast, the double-defect implementation (Figure 1b) introduces holes (or "defects") into the lattice, which are locations where the syndrome measurements are turned off. This creates a standalone qubit between the two defects (the lattice boundary is no longer needed), and thus a monolithic lattice is now able to accommodate multiple connected double-defect tiles of this type.

As discussed previously, detecting bit-flips and phase-flips is sufficient to correct arbitrary qubit errors. Therefore, ancillas are alternately designated as those that measure bit-flip



(a) Planar (b) Double-Defect

Figure 2: (a) Surface code error detection. A matching algorithm over anomalous syndromes can determine the location of faulty qubits. (b) Defects that are stretched through space-time create "braids" — 3D pipes used to operate on the encoded logical qubit.

syndromes and those that measure phase-flip syndromes from their surrounding data (shown as orange and blue interactions in Figure 1). By recording a history of syndrome measurements, we are able to detect (in classical software) anomalous syndromes and trace their cause to specific qubit corruptions. This collection of syndrome histories can be thought of as a 3D space-time volume; the bottom area being the lattice, and the vertical height the progress in time. This is shown in Figure 2.

2.4 Superconducting Technology

A number of technologies have been proposed for implementing qubits. Ongoing experimental physics research seeks to create a technology that can accommodate large numbers of stable qubits, operations with high fidelity, and the ability to easily address and control the system's many qubits. In this work we focus on superconducting technology as one promising step towards those goals. Owing to fast clock speeds and low error rates [5, 15, 66], superconducting qubits have become the leading candidate for large-scale QC. This technology has a good prospect of scaling up to a large number of qubits [21], and operation clock rates are in the high range of quantum technology, currently on the order of 10-100 MHz [58, 79]. The qubits in this technology are difficult to move, and thus only interact over short distances with nearby qubits. The implications of this on qubit communication is discussed in Section 4. Superconducting systems meeting the necessary criteria to run surface codes have already been demonstrated [5].

3 RELATED WORK

Several prior works have estimated quantum resources using simple tallying of logical qubits and operations, and multiplying them by the number of resources necessary to correct one qubit and one operation [27, 41, 75]. This "spreadsheet" model simplifies several crucial constraints—for example, that distant qubits cannot arbitrarily interact. In this work we perform a more rigorous evaluation of the details of mapping to the architecture, taking into account data dependencies, qubit layouts, and contentions in communication. This exposes new opportunities; for example we find that communication-aware scheduling saves up to $\sim 7X$ in total execution time by reducing braid congestion.

The initial assignment of the program's logical qubits to the chip's physical qubits is an important optimization which can positively impact performance, if done correctly. Prior work has considered this problem, although several shortcomings exist. For example, some methods are manual or non-scalable, some only consider 1-dimensional architectures, and some ignore error correction or only consider concatenated error correction [12, 14, 23, 48, 49, 62, 63, 65, 68, 69, 81]. For the first time, we optimize two-dimensional surface code mappings by applying graph partitioning techniques to quantum programs to both achieve scalability and to reduce communication distance and congestion.

Furthermore, we examine the sensitivity of results to the characteristics of the high-level application and reliability of the low-level technology, parameters which evolve over time. While prior work has often relied on optimizing a particular QEC/technology choice [27, 41, 77], our work provides insights into the conditions where such choices are warranted. In a rapidly-moving field, this end-to-end toolflow is particularly important since it allows one to change approaches fluidly to try new methods as technologies or assumptions evolve.

Double-defect codes have been the favored QEC option in the literature [27, 28, 41, 59]. This is due to simplicity of its uniform control and the fact that a monolithic lattice can in theory be made as large as needed. In addition, this encoding uses braids for communication, which are very fast forms of interacting distant qubits. However, we find that under certain conditions planar codes are actually better, owing to two properties: first, planar tiles are smaller (i.e. fewer qubits needed for the same code distance). Second, planar tiles communicate through *teleportations*, which are prefetchable (a major advantage in high-contention scenarios).

Recent work has begun addressing the issue of braiding automation by performing automated constructions of 3D topological pipes [57, 59, 60]. However scalability and extending to all operations remains an area of future work in these methods. Our proposed alternative is to use more scalable 2D networking methods. Several recent papers have proposed optimized qubit layouts on 2D grids [24, 48, 68]. However, this problem has not been studied in the context of surface codes or in larger scales. Our work fills this gap.

Finally, this work demonstrates the importance of paying attention to the high-level application when choosing the best type of error correction, a factor that has received little attention in prior work. Patil et al.'s study [61] similarly reveals that high-level application characteristics must be taken into account when choosing the right implementation of a quantum library subroutine, in the event that many implementations of it are available.

4 MICROARCHITECTURAL DESIGN

In this section, we describe the microarchitectural components necessary to support computation on qubits encoded in the planar and double-defect schemes. In this section, we focus on logical qubits and in later sections will implement these logical qubits with multiple physical qubits in our evaluation. We begin by discussing several important factors that must guide microarchitectural design, and then discuss our evaluated microarchitectures.

4.1 Handling Data Movement

A requirement of universal quantum computing is long-range interaction of logical qubits. First, because an algorithm needs to operate on physically-remote qubits. Second, this is required since magic states are prepared in dedicated regions of the hardware, yet need to be communicated to the point of use when a T operation is needed. Many technologies, including superconductors, do not allow easy transport of qubits. However, two methods exist to communicate the data in surface-encoded logical qubits: *teleportation* for planar qubits [8], and *braiding* for double-defect qubits [27].

Teleportation is a technique for long-distance communication of exact quantum states. It works by qubit *entanglement*, which causes operations on one qubit to affect the state of another entangled qubit, even if the two are no longer physically close. When the source and destination qubits of the communication are both entangled to an intermediary qubit, they are "linked," even when far apart. However, this does not entirely solve the problem of data movement. In order to establish a virtual link, two qubits can become entangled while physically together, but they must then be physically transported to the desired source and destination locations of the communication. This is known as *EPR distribution*—supplying pairs of qubits placed in a special EPR state [7, 8, 54]. Fortunately, since EPR pairs are independent of data, they may be entangled and distributed (i.e. prefetched) in advance and in fairly latency-tolerant manner.

Swapping is another suitable method for physical movement of quantum data from one location to another. Since superconducting qubits may only interact with close-range qubits, a chain of swaps can be used to achieve the same effect as long-distance mobility. Swapping is an expensive operation. Not only does the travel time increase substantially with distance, but the channels that are used for movement also consume extra "dummy" qubits, who exist only for the purpose of facilitating swaps. However, once EPRs are distributed, teleportation only has a small constant latency, independent of distance. The parameters of swaps and teleportations have been derived in [56].

Table 1: Summary of tradeoffs in communication efficency among the two main flavors of the surface code.

	Communication Method	Space (Qubits)	Time (Latency)	Pre-fetchable?
Planar	Teleportation	Low	High	Yes
Double-Defect	Braiding	High	Low	No

While planar codes can use teleportation to communicate, double-defect codes use an alternate communication form known as *braiding*. Braiding consists of extending defects through space (turning off all syndrome measurements along the way). The source and destination of the braid will become entangled, and the braid can then shrink back to its original starting defect. The advantage of braids is that their extension and shrinkage only takes one cycle, regardless of the distance. Their disadvantage, however, is that they cannot cross, and they cannot be prefetched—each communication must occur at the point it is needed since there are no separate communication ancillas involved. To summarize, Table 1 shows the main differences in how the two communication methods behave.

4.2 Differences to Classical Communication

While the problem of communicating qubits may seem similar to a classical networking problem, in fact several key distinctions call for new approaches. First, quantum data is physical—it cannot exist in two places at once and cannot be copied. This puts more pressure on QEC since any corrupted or lost data cannot simply be retransmitted. Second, communication not only has a time overhead, it also incurs a space overhead in terms of ancillas. Third, quantum applications are almost always specialized to concrete problem inputs (e.g. factoring a 2048-bit number). This means that the execution trace and therefore the point of each communication is known in advance. This global knowledge yields opportunities for aggressive optimization.

More specifically, the two surface code communication options discussed present new challenges. Teleportation is a non-traditional 2-step communication process, where step 1 may cause collision but is prefetchable. The goal of the optimizer must be to orchestrate this highly flexible step. Namely, do not distribute EPRs too early since they may cause traffic, and do not distribute too late since they may stall computation. A smooth, low-contention "just-in-time" distribution can be achieved with full knowledge of the program path. In braiding, there is no actual transmission of information (only expansion of lattice defects). So a braid can stretch all the way in 1 error correction cycle. However, these defects cannot physically co-exist close by, prohibiting crossing, buffers or virtual channels. The static availability of communication points again facilitates optimization heuristics to reduce traffic.

4.3 Ancilla Generation

In addition to the physical ancillas required for syndrome measurement on the surface code lattice, we also need ancillas at the logical level. In particular, a steady supply of magic states and EPR states are needed for performing T operations and teleportations, respectively.

We use so-called "ancilla factories" [39, 41, 74, 78] to dedicate specialized regions of the architecture to continuously prepare and supply ancillas. This creates more communication pressure: for every T operation and every teleportation, the relative ancillas are produced in factories and consumed at the location of data. Furthermore, ancilla factories can be large—every magic state factory consumes 12 encoded qubits (and much more on a faultier device) [41]. Previous studies have found that a factory-to-data footprint of almost 10:1 is needed if ancilla generation is to be taken off of the critical path[39]. Conversely, saving more space will cost more in time. In our empirical model, we have found that a good space-time balance is achieved with a 1:4 ancilla-to-data ratio.

4.4 Multi-SIMD Architecture for Planar QEC

An advantage of planar codes, beside their smaller size, is that applying a logical operation on the plane amounts to applying identical bitwise operations on individual physical qubits inside the plane, and qubit communication can occur through bitwise interactions between two planes (Figure 3a). This suggests that a SIMD-style architecture is suitable. We use the *Multi-SIMD* architecture proposed by [35], which combines qubit-level parallelism with operation-level parallelism: many qubits undergoing the same operation are clustered in one SIMD region, and multiple (reconfigurable) SIMD regions can accommodate heterogenous types of operations at any cycle. There is well-established technological support for this: microwave broadcasts can be used to operate simultaneously on many superconducting qubits [52, 53].

Figure 3a depicts the different regions at use: a checkerboard layout of SIMD and memory regions allows for easy access for qubit computation and storage. Movement is a natural consequence of this architecture. We use teleportation as it decouples the hard part of communication (the network congestion) from the data communication itself. The former is done on EPRs and the latter on data qubits. In this sense, only EPRs use the communication mesh, whereas data and magic states get teleported to where they are needed.

The bitwise coupling of planar qubits—needed for 2-qubit operations and swap channels—has traditionally been difficult in superconductors, because it requires 3D connections between non-adjacent qubits. Recent work in developing air-bridge crossovers [13], vias [5] and flip-chip crossovers [2] has, however, mitigated this problem. These connections can connect medium-distance qubits and are protected from cross-talk by shielding methods [11]. Also importantly, the development of low-loss, 3D connections seem to be necessary even in double-defect codes, because of the need for off-chip control and measurement. Other possible 3D architectures to couple medium-range planar qubits have also been described [36].

4.5 Tiled Architecture for Double-Defect QEC

The double-defect code defines all qubits on one large, monolithic lattice. The tiled architecture in Figure 3b assigns one tile per qubit, and opens channels between them to allow for communication braids. Similar to the Multi-SIMD architecture, we reserve some tiles for continuous generation of magic states, to be braided to various points of use. No EPR factory is needed, since no teleportation occurs.

5 TOOLFLOW AND APPLICATIONS

This section presents our comparative evaluation methodology, including the studied quantum applications and the end-to-end toolflow developed for this purpose.

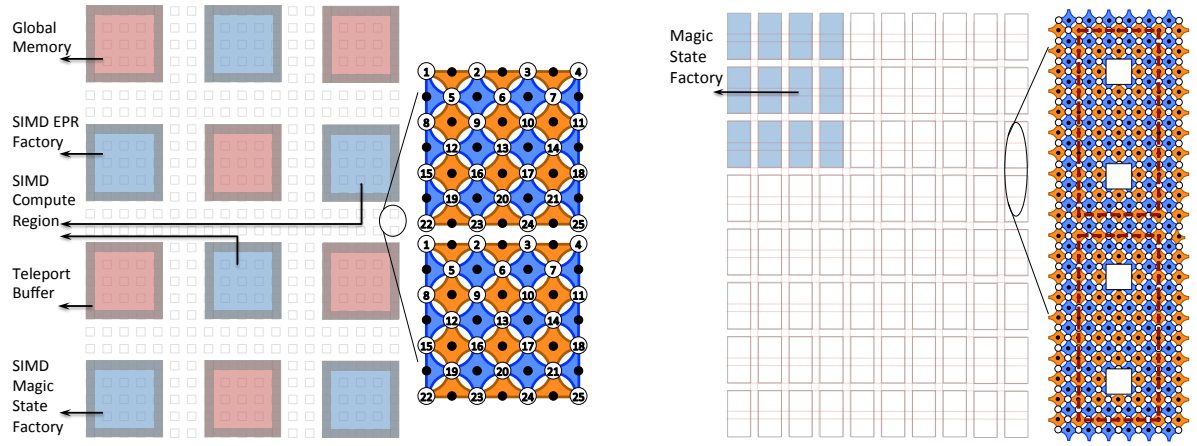
5.1 Overall Toolflow

Designing a quantum computer is a complex task. Automated tools play an invaluable part in highlighting the tradeoffs of the design, and ensuring that every step is optimized such that quantum resources are used as efficiently as possible. Suppose, for example, that we wish to run a particular application on a given hardware. First, the application must be analyzed for its parallelism potential, and this must be matched with the microarchitecture and hardware's support for parallelism. Second, points of qubit communication during the program run must be identified, and the overhead due to network congestion accurately analyzed. Third, challenging space-time tradeoffs must be evaluated which go beyond mere qualitative reasoning. For example, in designing quantum applications it is typical to trade qubits for time [34]. However, these two parameters also affect each other. If we decrease qubits, time will be increased which causes more error accumulation, which may force us to compensate with stronger error correction which will in turn increase the qubits. In summary, a design toolflow must model all of these pieces, yielding a resource and performance estimate informed by application characteristics (e.g. parallelism), technology characteristics (e.g. clock speeds, reliability rates, mobility of qubits), and communication characteristics (e.g. contention, teleportation vs. braiding models, straight moves vs. turns).

Our toolflow's integration allows for analysis of program graphs through multiple stages and application of various optimization heuristics. It also facilitates iterative designs through feedback capabilities. For example, the overhead of downstream error correction can often depend on the degree of code inlining in earlier compilation stages, which can be optimized iteratively (as discussed in Section 7). Finally, the toolflow also enables sensitivity studies such as in Section 7.3). Figure 4 depicts our overall toolflow.

5.2 Applications

As input to our toolflow we use a set of applications of varying size, characteristics, and functions. Table 2 summarizes



(a) Multi-SIMD Architecture (b) Tiled Architecture

Figure 3: Microarchitectural designs with zoomed-in physical layers. (a) Multi-SIMD architecture for planar QEC. SIMD regions (blue) are used to apply similar operations to many planar qubits in parallel. Distributed memory regions (red) store idle data. Dedicated SIMD regions are used as ancilla factories, for steady generation of magic states and EPRs. Each region is surrounded by a teleport buffer (grey), that entangle EPRs to data to be teleported. EPR distribution occurs through planar swaps—numbered qubits in the zoomed-in figure show bitwise couplings needed to perform swaps. (b) Tiled architecture for double-defect QEC. Braid channels (red) connect tiles (black), and also pass between the double defects in every tile. Dedicated factories supply magic states to surrounding tiles.

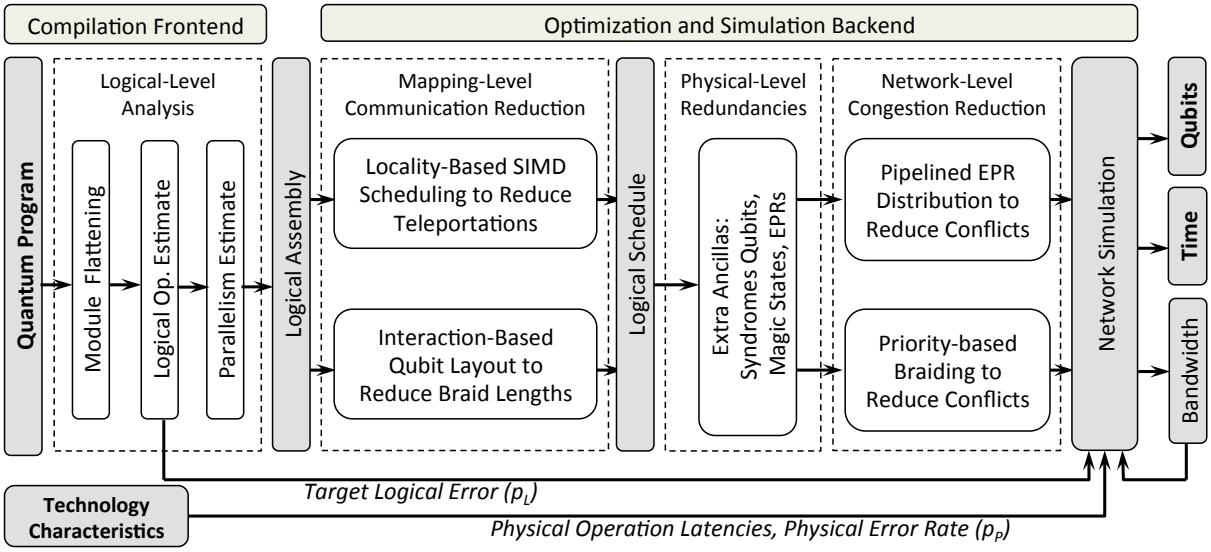


Figure 4: Overview of our toolflow. The frontend performs logical compilation, and is based on [40]. The backend performs mapping to and optimizations on the relevant microarchitecture. The top optimization path pertains to teleportation-based communication in planar codes; the bottom path concerns braid-based communication in double-defect codes (Section 6).

these. Of particular interest is the parallelism potential of applications, which affects the performance of optimization protocols, and therefore the tradeoffs between the different QEC and communication methods we consider. We evaluate two highly-parallel applications (IM and SHA-1) and two mostly-serial applications (SQ and GSE).

Table 2: Summary of studied quantum applications. Parallelism factor is average number of logical operations that can be concurrently executed, were hardware resources not a constraint (ideal parallelizability).

	Application Purpose	Parallelism Factor
Ground State Estimation (GSE)	Compute ground state energy for molecule of size m [80]	1.2
Square Root (SQ)	Find square root of an n-bit number [32]	1.5
SHA-1 Decryption (SHA-1)	SHA-1 decryption of n-bit message [55]	29
Ising Model (IM)	Finding ground state for ising model on n-qubit spin chain [6]	66

5.3 Compilation Frontend

We use the ScaffCC compiler [40] as the frontend to our toolflow, lowering high-level descriptions of quantum algorithms to a standard logical-level ISA known as QASM (quantum assembly) [16, 17].

The frontend performs logical-level resource and parallelism estimations. They guide the backend network optimization policy, as well as choice of code distance to meet the logical reliability requirements. That is, the resource estimation helps discover the number of logical operations that must be executed (size of computation), which is inversely proportional to the target logical error (p_L). This, in conjunction with the physical error rate (p_P) furnished by the technology characteristics, helps determine the strength of surface code error correction that is needed (d).

5.4 Optimization and Simulation Backend

The backend is responsible for adding physical-level redundancies to ensure error tolerance, as well as performing optimizations and simulations to calculate the final space-time overhead. Optimizations occur at two levels: *mapping-level* reduction of total communication needs through exploitation of data locality, and *network-level* improvements in the overhead of remaining communications through optimizing the network load.

Mapping-level communication reduction is an important optimization stage: error correction is expensive, and a reduced operation count yields multiplicative benefits. First, fewer operations must be protected against errors, and second, those that do need to be protected can afford a weaker form of correction since the overall program is smaller. We apply the mapping approach of [35] to the Multi-SIMD architecture, which reduces unnecessary teleportations between regions. Network-level optimizations on the Multi-SIMD architecture consist of pipelining EPR distributions to avoid high congestion, discussed in Section 8. The next section will discuss novel mapping- and network-level optimizations in the context of braids on the tiled architecture.

6 OPTIMIZED BRAIDING

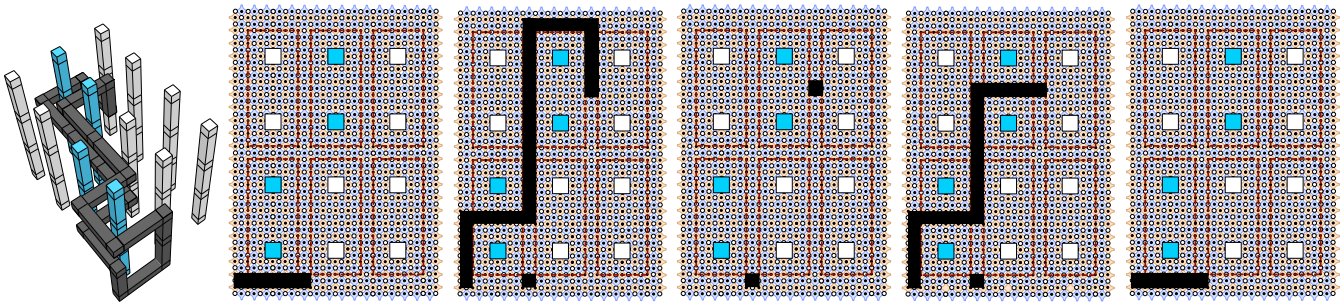
In this section, we describe a scalable yet efficient solution to the braiding problem, which is the main method of computation and communication in double-defect surface codes. Section 6.1 lays out our general method, while Sections 6.2 and 6.3 propose mapping- and network-level optimizations. This corresponds to the optimization flow on the bottom path of Figure 4.

6.1 Braid Simulation using Message Passing

Figure 5 shows how a simple 2-qubit logical operation may be performed between two distant double-defect logical qubits. As discussed previously, this becomes a braiding operation that amounts to topological pipes in the 3D space-time volume (Figure 5a). Traditionally, such volumes have been compressed by hand to decrease space and time [20, 26, 28]. For example, [19] creates a game in which players try to manually compress 3D "pipes" using topological transformations. Although these techniques achieve very good results on small problems, they are difficult to scale. Instead, we propose an automated braid synthesis and optimization approach that is much more scalable, and yet achieves good performance. We do this by tracing the state of the 2D lattice over various vertical time slices in the 3D volume. As such, the problem is reduced to simulating a mesh network, with braids as messages in this network. Figures 5b-5f show how the 3D volume can be broken down into five snapshots of braid opening and closing (i.e. message passing). We call each stage an "event".

This translation of the problem is an overconstraining, since not all possible topological transformations can be captured in this way (e.g. contrary to network messages, 3D topological pipes can be morphed back in time too). However, we demonstrate that with appropriate heuristics, we can reclaim much of the efficiency of optimal solutions, with the advantage of staying scalable (Sections 6.2 and 6.3). Our approach relies on performing dynamic routing for a static problem. The reason network routing works well here is that our applications have parallelism that is not too high and not too low. That is, we can afford to overconstrain the problem and still resolve most conflicts since there aren't too many, but we can't be too naive about it. Manual approaches try to solve every conflict as optimally as possible, but that is not necessary in practice and not scalable. In fact, our tools are what allow us to obtain these insights. Since we replay the dynamic schedule as a static one at execution time on the quantum computer, we need not worry about deadlock and livelock. We only have to use heuristics to find a deadlock- and livelock-free solution at compile time. This is much easier than guaranteeing deadlock- and livelock-freedom at runtime in truly dynamic schemes.

Braids differ from conventional messages in several ways: (a) braids travel n hops all the way from source to destination in one cycle, (b) some braids have to remain stable for d (code distance) cycles to stabilize syndrome measurements, (c) routers cannot buffer braids and (d) virtual channels cannot be used as braids cannot physically use the same



(a) Time-space volume of a 2-qubit logical operation
(b) $time = 0$ Intialize Ancilla
(c) $time = d$ Braid Part 1
(d) $time = d+1$ Stabilize
(e) $time = 2d$ Braid Part 2
(f) $time = 2d+1$ Stabilize/Measure Ancilla

Figure 5: Logical operation between two qubits (light blue shade for clarity) of a six-qubit setup, arranged in a 2x3 tiled architecture. (a) depicts the entire space-time volume (time goes up vertically)—the topological method used in traditional optimizations. (b-f) each represent a slice in time, called an "event." From a network perspective, black defects are messages routed in the mesh, and the tile corners are routers.

channel. For these reasons, we use a circuit-switched network. The n hop/cycle property means that a braid claims the route's resources (nodes and links) at once when opened, and releases them when closed. We use a greedy approach of trying to place as many braids as possible, in order to reduce overall cycles.

In our braiding algorithm, we maintain a ready queue of operations whose dependencies have been met, and execute as many of them as possible in each cycle. Each event pertains to an open or close braid, and has a timer corresponding to how long the braid should remain in case error syndromes need to be extracted. To improve forward progress in a busy network, we add route adaptivity to a dimension-ordered route and a drop/re-inject mechanism, both after certain timeouts. Note that we are using a dynamic mechanism (network routing) to find a *static schedule* (which will be replayed during the execution of our quantum program). This means that our protocols do not need to be deadlock- or livelock-free. If we encounter lack of forward progress, our mechanisms just backtrack and try again. There will be no cost at execution time, since failed schedules are not recorded and used.

6.2 Optimizing Qubit Arrangement

The first step to reduce braid contention is applying mapping-level optimizations to the placement of qubit tiles on the 2D mesh. Our goal is to map logical tiles which interact frequently close to each other. Specifically, the optimized arrangement of qubit tiles attempts to minimize the sum of Manhattan distances between pairs of tiles involved in non-local, braiding operations. We do this through iterative calls to a graph partitioning library, METIS [42], to separate the qubits (each represented as a vertex on a graph of qubit interactions) into two partitions, such that the weight of crossing edges is small. Relative to a naive arrangement of qubits, the optimized qubit arrangement reduces the lengths of braids, hence reducing the chance of braid collisions.

6.3 Optimized Braid Priorities

- *Policy 0*: No optimization. All operations and events in program order.
- *Policy 1*: Interleave: Allow individual events to be interleaved, but keep operations in program order.
- *Policy 2*: Interleave + layout: Optimize initial qubit layout for interaction distances.
- *Policy 3*: Interleave + layout + criticality: Sort operations by highest criticality first.
- *Policy 4*: Interleave + layout + length: Sort braids by longest first.
- *Policy 5*: Interleave + layout + type: Sort by closing braids first, opening braids next.
- *Policy 6*: Combines Policy 1–5: Interleave, optimize initial layout, sort closing braids first, sort by criticality, sort short-to-long for highest-criticality braids, sort long-to-short for lower criticality braids.

In our braiding algorithm, when multiple braids are eligible to be scheduled but not all of them fit on the mesh, the choice of priorities can have a significant impact on performance. We have devised heuristic policies to prioritize "more important" braids. It must be noted that such policies are only essential in highly-parallel applications (SHA-1 and IM), where there is an initially large discrepancy between the schedule length and critical-path length, due to communication conflicts. Low parallelism reduces the need for interference optimization from the start.

The metrics we use to assess priority are the criticality of the braid (how many future operations depend on it), its length, and its type (whether it is a closing or opening braid). Our prioritization policies are summarized below:

Policy 1 (event interleaving) allows for multiple braids to progress concurrently and at different rates. Policy 2 is the same layout localization heuristic of Section 6.2. To these, Policies 3 through 5 each add one important metric by which the importance of a braid may be judged: criticality, length, type. Policy 6 combines all of the metrics above. Here, we give higher priority to those braids that are closing rather than

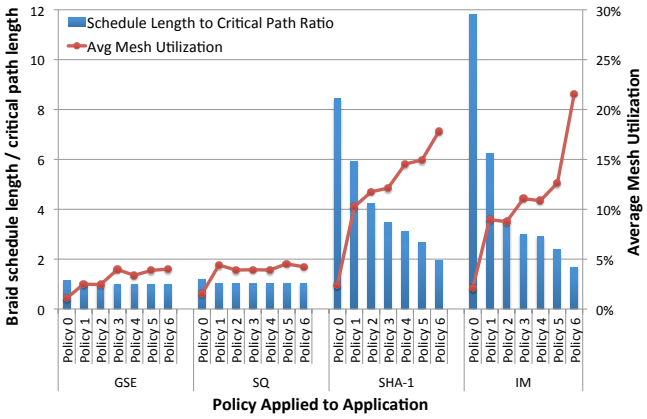


Figure 6: Braid simulation results for the double-defect surface code. Blue bars show schedule length can be reduced by up to $\sim 7X$, to within $\sim 70\%$ of the critical path for highly-parallel applications (SHA-1 and IM), where risk of contention is high. Serial applications (GSE and SQ) already achieve close-to-critical-path schedules. Red curves show the increase in network utilization when using these policies, up to about 22%. These improvements are achieved through interaction-aware qubit placements and priority-aware braid placements.

opening (so they can release network resources), and those that have higher criticality (to remove the bottleneck). If two braids have the same criticality, we decide based on length: shorter braids for the most critical operations, because we want to accomplish as many as possible, and longer braids for lower-criticality operations, because we want to conclude the toughest braids ahead of time.

Figure 6 shows our results. The blue bars (associated with the left vertical axis) show how the above prioritization policies improve performance by reducing the gap between the braid schedule lengths and critical path lengths. Our results show that evaluated individually, braid type causes the largest improvement, while improvements due to length and criticality are smaller.

The most successful policy, Policy 6, is able to reduce schedule length by up to $\sim 7X$, from 12X longer than the critical path, to within 70% of it. The red curves (right vertical axis) shows network utilization rate (i.e. percentage of busy links). It demonstrates that better prioritization results in an almost 8-fold increase in network utilization—up to about 22%. This is an acceptable range for this highly scalable approach, comparable to similar circuit-switched networks.

7 RESULTS

In the context of our microarchitectures, technology, and tools, we can now compare our QEC methods for each benchmark application.

We begin by quantifying some absolute values for the number of qubits and time required to run a quantum application, and then proceed to evaluate the effect of QEC choice on performance. These results can help determine the most suitable type of error correction and microarchitecture for a particular application (p_L) on a particular physical technology (p_P).

7.1 Scaling Effects on Space and Time

Figure 7 shows concrete values for the number of qubits and amount of time needed to execute a fully-error-corrected SQ application. The graphs are plotted for various input problem sizes, each of which directly determines the size of computation (inversely proportional to target logical error rate (p_L)). Superconductors are fast, and we see that small instances of the problem can execute in under one second. Increasing the problem size, however, can greatly increase time of computation. The number of qubits does not rise as sharply, but it still illustrates the scaling needs of future computers: in order to run even modest problem sizes, around 1000 qubits are needed. The main increases in qubit usage occur when the code distance (d) must be increased to support larger computations. Other applications exhibit similar scaling trends.

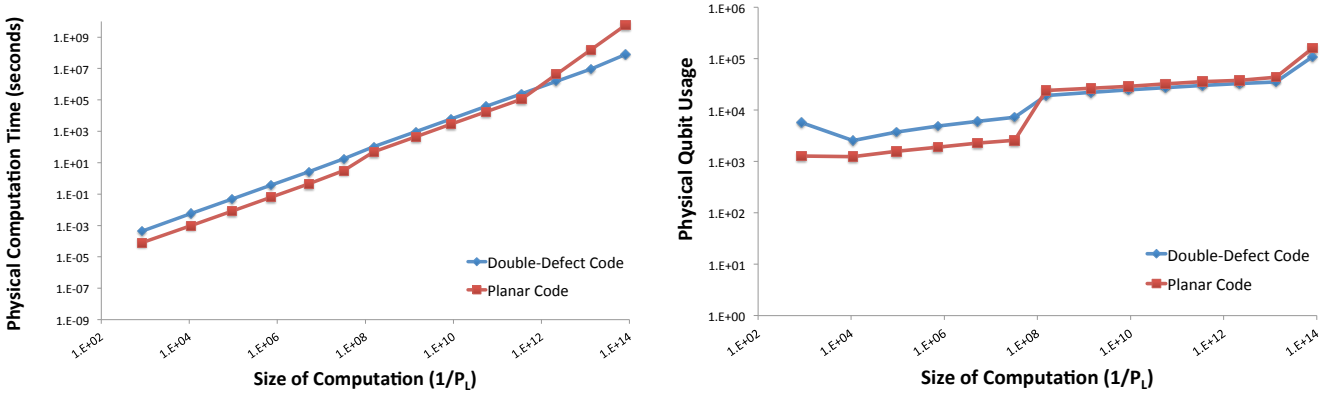
Although Figure 7 is useful for obtaining a sense of absolute overheads, it does not depict the difference between different encodings well—the lines are close to each other due to the logarithmic axes. Next, we look at the ratios of double-defect to planar resource usage.

7.2 Effects of Encoding

We now explore the differing overheads when using planar and double-defect encodings in two applications, SQ and IM, specifically chosen for their parallelism characteristics (Figure 8b). Our metric is resource usage (qubits and time), normalized to planar codes as a baseline.

SQ is largely serial, whereas IM is highly parallel. In both applications, when the computation size is small, planar codes fare better. This is due to the smaller size of planar lattices. However, once the computation size exceeds a certain amount (indicated as the "cross-over point"), double-defect codes become better, due to the efficiency of braids compared to slower swaps. Favorability cross-over occurs where the space-time ratio ($qubits \times time$) crosses 1.

Comparing Figures 8a and 8b we find that the cross-over point in IM occurs much later than SQ (at a larger computation size). This is directly due to the high parallelism of the IM application. Parallelism causes braid congestions in double-defect codes, but the EPRs in planar codes can still be pipelined in a way to avoid congestion. Therefore, planar qubits remain better for longer. Furthermore, the Multi-SIMD architecture of planar codes supports data and instruction parallelism, again improving the performance of planar codes which can use parallel bitwise operations in this architecture. Note that these results pertain to the fundamental difference in braiding versus teleportation, and even though we have evaluated them in the context of specific architectures, our observations apply to other proposed architectures [50, 51] (which, in fact, are more favorable to planar codes).



(a) Time (b) Qubits (Space)

Figure 7: Absolute resource usage in (a) time and (b) space to run error corrected SQ applications of varying sizes. In these simulations we assume $p_P = 10^{-8}$ and single-qubit operations are 10X faster than 2-qubit operations.

7.3 Sensitivity Analysis

Having established how a cross-over occurs from the favorability of planar codes to the favorability of double-defect codes, we can now plot this point for a full sweep of physical and logical error rates for all applications. Figure 9 shows this graph. Each line on this graph is a collection of cross-over points, showing their sensitivity to changing physical error rates from $p_P = 10^{-8}$ (future optimistic) to $p_P = 10^{-3}$ (current [70, 71]). For each application, the line demarcates the boundary of designs where planar codes should be used (below) and where double-defect codes should be used (above).

Such lines are application- and technology-dependent, due to the entirely different models of computation (bitwise vs. braids) and communication (teleportation vs. braids) in these codes, which must be simulated in order to yield accurate contention rates. We observe that boundaries are generally higher for more parallel applications, suggesting that these applications would benefit more from using planar codes. We have used two variations of the IM application—with medium and maximal inlining. More code inlining creates more parallelism, consistent with the upward boundary movement.

In summary, our results suggest that although double-defect codes have been considered the standard method of error correction on superconducting computers before [5, 27], subtleties related to communication contention and application parallelism may alter this view. As device error rates continue to improve (left in Figure 9), a shift to planar encoding may be warranted (bigger area under the curves).

8 DISCUSSION

8.1 Pipelined EPR Distribution

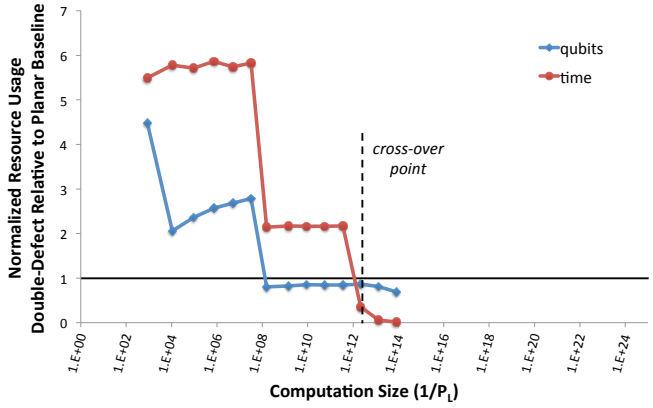
In the communication technique referred to as teleportation, the physical interconnection network does not directly move the data qubits themselves, but instead carries EPR qubits that facilitate the communication of entangled qubits at a distance.

Because of the delay-tolerant nature of the distribution of EPRs, they are not bound by stringent data dependencies, and they can be prefetched at arbitrary points in time. Our goal is to achieve "just-in-time" distribution by smoothing the network load to reduce congestion. This can have favorable effects on both space and time: the number of EPR qubits will be reduced as they are consumed (and recycled) shortly after being introduced to the network; latency will be reduced since EPRs are present when they are needed, and hence do not stall teleportations.

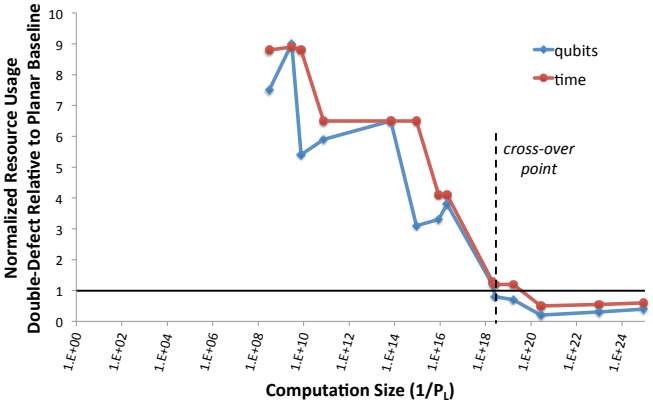
Given the specialized nature of quantum applications, we have perfect static knowledge of when each EPR will be needed. Hence, walking the dependency graph, we use look-ahead windows to anticipate usage points, and launch their communication with appropriate lead time. Such approaches achieve good performance across all applications, with up to $\sim 24X$ savings in qubit cost and only a maximum of $\sim 4\%$ extra latency. The choice of lead time ("window size") is important to achieve just-in-time distribution. Smaller window sizes cap qubit usage at the expense of starving data qubits in need of teleportation. In contrast, large windows release more EPRs into the network than necessary. Suitable window sizes depend on application size, but degree of application parallelism has little effect, since ancillas do not follow regular data dependencies.

8.2 Alternative Communication Methods

While this paper has discussed teleportation and braiding as two main communication methods, recent work has investigated *lattice surgery* [38] as a hybrid scheme that combines the low qubit overheads of the planar code with nearest-neighbor-only interactions. Communication in lattice surgery occurs using *merge* and *split* operations: two adjacent planar



(a) SQ Application (Serial)



(b) IM Application (Parallel)

Figure 8: Comparing resource usage in double-defect and planar codes, for a range of problem sizes in the (a) SQ and (b) IM applications. Lower is better, and overall preference is given to the QEC method that has a smaller $qubit \times time$ product. In both (a) and (b), planar codes are better at smaller sizes but at some cross-over point, double-defect codes become better. However, the cross-over point occurs at a much larger computation size for IM, compared to SQ. This is due to high braid congestion in the highly parallel IM app, which causes planar (teleport-based) codes to remain better for longer. Figure shown for a technology with error rate $p_P = 10^{-8}$.

encoded qubits are merged by turning on syndrome measurements at their connecting boundary, creating a larger continuous plane. Similarly, turning those syndromes off again will split the two planes into their original form. Repeated applications of these operations in a chain can cause distant planes to interact. Optimal lattice surgery is an NP-hard problem [37], and scalable heuristics remain as an area of future work. Crucially in the context of this study, the chain of merges and splits does not have the benefits of braids

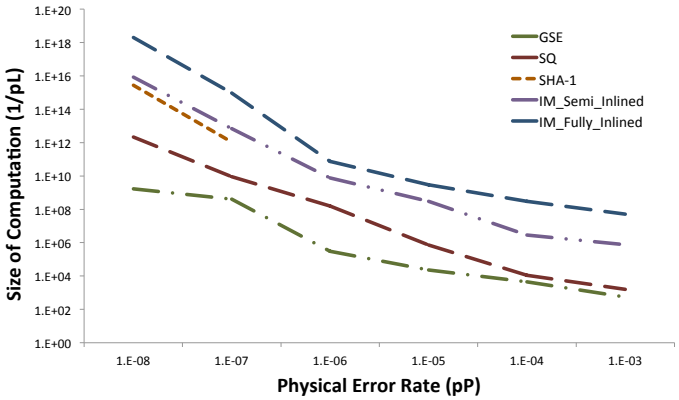


Figure 9: Comparison of double-defect vs. planar surface codes, across the range of possible physical and logical error rates. Higher y-axis value (up) means more logical operations in the application. Higher x-axis values (right) mean faultier technology. Design points under the curves work better with planar codes. Each line delineates the cross-over boundary for a particular application. The curve is higher for more parallel applications (SHA-1, IM), since congestion hurts braids more.

(fast movement) nor teleportation (prefetchability). Therefore, we instead focused on two more promising forms of communication.

9 CONCLUSIONS

This paper has focused on optimizing qubit communications as a major source of latency in quantum computers. Focusing on surface code error correction in superconducting qubits, the leading candidate to implement a future quantum computer, we investigate teleportation-based and braid-based communications. We present optimization algorithms that achieve near-critical-path performance, while staying scalable to problem sizes of more than 10^{20} operations. Our design-space exploration allows us to plot favorability curves such as Figure 9, which is useful as a guide in QC design. For example, it shows us that for near-term superconductor error rates of 10^{-4}–10^{-3}, planar encoding is better for serial applications of shorter than $\sim 10^4$ logical operations and parallel apps of shorter than $\sim 10^8$ logical operations. We believe the large gains from running a quantum application is such that each real quantum machine will likely be dedicated to solving a specific task; e.g. breaking large encryption keys or finding the ground state of large molecules. It is therefore suitable to pick design parameters based on the intended application. Similarly, we have shown that the physical device characteristics (i.e. error rate) must play a role in choosing suitable error correction codes and microarchitectures. This paper therefore argues for a co-design of applications, microarchitectures and physical hardware in building future quantum computers.

REFERENCES

[1] Scott Aaronson. 2005. Guest column: NP-complete problems and physical reality. *ACM Sigact News* 36, 1 (2005), 30–52.

[2] David W Abraham, Jerry M Chow, Antonio D Corcoles Gonzalez, George A Keefe, Mary E Rothwell, James R Rozen, and Matthias Steffen. 2015. Removal of spurious microwave modes via flip-chip crossover. (Dec. 22 2015). US Patent 9,219,298.

[3] D. Aharonov and M. Ben-Or. 1997. Fault-tolerant quantum computation with constant error. In *STOC*. ACM.

[4] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. 1995. Elementary gates for quantum computation. *Physical Review A* 52, 5 (1995), 3457.

[5] R Barends, J Kelly, A Megrant, A Veitia, D Sank, E Jeffrey, TC White, J Mutus, AG Fowler, B Campbell, Y Chen, Z Chen, B Chiaro, A Dunsworth, C Neill, P. J. J O'Malley, P Roushan, A Vainsencher, J Wenner, A. N Korotkov, A. N Cleland, and John M Martinis. 2014. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature* 508, 7497 (2014), 500–503.

[6] R Barends, A Shabani, L Lamata, J Kelly, A Mezzacapo, U Las Heras, R Babbush, AG Fowler, B Campbell, Yu Chen, Z Chen, B Chiaro, A Dunsworth, E Jeffrey, A Lucero, A Megrant, JY Mutus, M Neeley, C Neill, P. J. J O'Malley, C Quintana, P Roushan, D Sank, A Vainsencher, and J Wenner. 2016. Digitized adiabatic quantum computing with a superconducting circuit. *Nature* 534, 7606 (2016), 222–226.

[7] John S Bell. 1964. On the Einstein Podolsky Rosen paradox. *Physics* 1, 3 (1964), 195–200.

[8] Charles H Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K Wootters. 1993. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical review letters* 70, 13 (1993), 1895.

[9] Sergey Bravyi and Alexei Kitaev. 2005. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Physical Review A* 71, 2 (2005).

[10] Sergey B Bravyi and A Yu Kitaev. 1998. Quantum codes on a lattice with boundary. *arXiv preprint quant-ph/9811052* (1998).

[11] Teresa Brecht, Wolfgang Pfaff, Chen Wang, Yiwen Chu, Luigi Frunzio, Michel H Devoret, and Robert J Schoelkopf. 2015. Multilayer microwave integrated quantum circuits for scalable quantum computing. *arXiv preprint arXiv:1509.01127* (2015).

[12] Amlan Chakrabarti, Susmita Sur-Kolay, and Ayan Chaudhury. 2011. Linear nearest neighbor synthesis of reversible circuits by graph partitioning. *arXiv preprint arXiv:1112.0564* (2011).

[13] Zijun Chen, Anthony Megrant, Julian Kelly, Rami Barends, Joerg Bochmann, Yu Chen, Ben Chiaro, Andrew Dunsworth, Evan Jeffrey, JY Mutus, et al. 2014. Fabrication and characterization of aluminum airbridges for superconducting microwave circuits. *Applied Physics Letters* 104, 5 (2014), 052602.

[14] Byung-Soo Choi and Rodney Van Meter. 2011. On the effect of quantum interaction distance on quantum addition circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 7, 3 (2011), 11.

[15] Jerry M Chow, Jay M Gambetta, AD Córcoles, Seth T Merkel, John A Smolin, Chad Rigetti, S Poletto, George A Keefe, Mary B Rothwell, JR Rozen, et al. 2012. Universal quantum gate set approaching fault-tolerant thresholds with superconducting qubits. *Physical review letters* 109, 6 (2012), 060501.

[16] I Chuang. 2016. Quantum Architectures: Qasm2Circ. (March 2016). http://www.media.mit.edu/quanta/qasm2circ/

[17] Andrew W. Cross. [n. d.]. qasm-tools. ([n. d.]). http://www.media.mit.edu/quanta/quanta-web/projects/qasm-tools

[18] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. 2002. Topological quantum memory. *J. Math. Phys.* 43, 9 (2002), 4452–4505.

[19] Simon J Devitt. 2016. Programming quantum computers using 3-D puzzles, coffee cups, and doughnuts. *XRDS: Crossroads, The ACM Magazine for Students* 23, 1 (2016), 45–50.

[20] Simon J Devitt, Ashley M Stephens, William J Munro, and Kae Nemoto. 2013. Requirements for fault-tolerant factoring on an atom-optics quantum computer. *Nature communications* 4 (2013).

[21] Michel H Devoret and Robert J Schoelkopf. 2013. Superconducting circuits for quantum information: an outlook. *Science* 339, 6124 (2013), 1169–1174.

[22] David P DiVincenzo. 2009. Fault-tolerant architectures for superconducting qubits. *Physica Scripta* 2009, T137 (2009), 014020.

[23] Mohammad Javad Dousti and Massoud Pedram. 2012. Minimizing the latency of quantum circuits during mapping to the ion-trap circuit fabric. In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 840–843.

[24] Mohammad Javad Dousti, Alireza Shafaei, and Massoud Pedram. 2014. Squash: a scalable quantum mapper considering ancilla sharing. In *Proceedings of the 24th edition of the great lakes symposium on VLSI*. ACM, 117–122.

[25] Jack Edmonds. 1965. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B* 69, 125-130 (1965), 55–56.

[26] Austin Fowler. 2012. Time-Optimal Quantum Computation. *arXiv preprint quant-ph/1210.4626* (2012).

[27] Austin G. Fowler. 2012. Surface codes: Towards practical large-scale quantum computation. *Physical Review A* 86, 3 (2012). https://doi.org/10.1103/PhysRevA.86.032324

[28] Austin G Fowler and Simon J Devitt. 2012. A bridge to lower overhead quantum computation. *arXiv preprint arXiv:1209.0510* (2012).

[29] Austin G Fowler, Ashley M Stephens, and Peter Groszkowski. 2009. High-threshold universal quantum computation on the surface code. *Physical Review A* 80, 5 (2009), 052312.

[30] Jay M Gambetta, Jerry M Chow, and Matthias Steffen. 2015. Building logical qubits in a superconducting quantum computing system. *arXiv preprint arXiv:1510.04375* (2015).

[31] Daniel Gottesman. 1998. Theory of fault-tolerant quantum computation. *Physical Review A* 57, 1 (1998), 127.

[32] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *STOC*. 8.

[33] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. 2009. Quantum algorithm for linear systems of equations. *Physical review letters* 103, 15 (2009), 150502.

[34] Matthew B Hastings, Dave Wecker, Bela Bauer, and Matthias Troyer. 2014. Improving quantum algorithms for quantum chemistry. *arXiv preprint arXiv:1403.1539* (2014).

[35] Jeff Heckey, Shruti Patil, Ali JavadiAbhari, Adam Holmes, Daniel Kudrow, Kenneth R Brown, Diana Franklin, Frederic T Chong, and Margaret Martonosi. 2015. Compiler management of communication and parallelism for quantum computation. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM.

[36] Ferdinand Helmer, Matteo Mariantoni, Austin G Fowler, Jan von Delft, Enrique Solano, and Florian Marquardt. 2009. Cavity grid for scalable quantum computation with superconducting circuits. *EPL (Europhysics Letters)* 85, 5 (2009), 50007.

[37] Daniel Herr, Franco Nori, and Simon J Devitt. 2017. Optimization of Lattice Surgery is Hard. *arXiv preprint arXiv:1702.00591* (2017).

[38] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. 2012. Surface code quantum computing by lattice surgery. *New Journal of Physics* 14, 12 (2012), 123011.

[39] Nemanja Isailovic, Mark Whitney, Yatish Patel, and John Kubiatowicz. 2008. Running a quantum circuit at the speed of data. In *ISCA*. IEEE Computer Society.

[40] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T Chong, and Margaret Martonosi. 2014. ScaffCC: A framework for compilation and analysis of quantum computing programs. In *Proceedings of the 11th ACM Conference on Computing Frontiers*. ACM, 1.

[41] N Cody Jones, Rodney Van Meter, Austin G Fowler, Peter L McMahon, Jungsang Kim, Thaddeus D Ladd, and Yoshihisa Yamamoto. 2012. Layered architecture for quantum computing. *Physical Review X* 2, 3 (2012), 031007.

[42] George Karypis and Vipin Kumar. 1999. A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing* 20, 1 (1999), 359–392.

[43] Julian Kelly, R Barends, AG Fowler, A Megrant, E Jeffrey, TC White, D Sank, JY Mutus, B Campbell, Yu Chen, et al. 2015. State preservation by repetitive error detection in a superconducting quantum circuit. *Nature* 519, 7541 (2015), 66–69.

[44] A Yu Kitaev. 1997. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys* 52, 6 (1997), 1191–1249.

[45] A Yu Kitaev. 2003. Fault-tolerant quantum computation by anyons. *Annals of Physics* 303, 1 (2003), 2–30.

[46] Emanuel Knill, Raymond Laflamme, and Wojciech H Zurek. 1998. Resilient quantum computation: error models and thresholds. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, Vol. 454. The Royal Society, 365–384.

[47] Daniel A Lidar and Haobin Wang. 1999. Calculating the thermal rate constant with exponential speedup on a quantum computer. *Physical Review E* 59, 2 (1999), 2429.

[48] Chia-Chun Lin, Amlan Chakrabarti, and Niraj K Jha. 2014. FTQLS: Fault-tolerant quantum logic synthesis. *IEEE Transactions on very large scale integration (VLSI) systems* 22, 6 (2014), 1350–1363.

[49] Chia-Chun Lin, Susmita Sur-Kolay, and Niraj K Jha. 2015. PAQCS: Physical design-aware fault-tolerant quantum circuit synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 7 (2015), 1221–1234.

[50] Tzvetan S. Metodi et al. 2005. A Quantum Logic Array Microarchitecture: Scalable Quantum Data Movement and Computation. In *MICRO*. IEEE Computer Society, 305–318.

[51] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim. 2012. Large Scale Modular Quantum Computer Architecture with Atomic Memory and Photonic Interconnects. *arXiv:quant-ph/1208.0391* (2012).

[52] JE Mooij, TP Orlando, L Levitov, Lin Tian, Caspar H Van der Wal, and Seth Lloyd. 1999. Josephson persistent-current qubit. *Science* 285, 5430 (1999), 1036–1039.

[53] Yu Nakamura, Yu A Pashkin, and JS Tsai. 1999. Coherent control of macroscopic quantum states in a single-Cooper-pair box. *Nature* 398, 6730 (1999), 786–788.

[54] Michael A Nielsen and Isaac L Chuang. 2010. *Quantum computation and quantum information*. Cambridge University Press.

[55] National Institute of Standards and Technology. 2012. *FIPS PUB 180-4: Secure Hash Standard (SHS)*. U.S. Department of Commerce.

[56] Mark Oskin, Frederic T Chong, Isaac L Chuang, and John Kubiatowicz. 2003. Building quantum wires: The long and the short of it. In *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*. IEEE, 374–385.

[57] Adam Paetznick and Austin G Fowler. 2013. Quantum circuit optimization by topological compaction in the surface code. *arXiv preprint arXiv:1304.2807* (2013).

[58] Hanhee Paik, DI Schuster, Lev S Bishop, G Kirchmair, G Catelani, AP Sears, BR Johnson, MJ Reagor, L Frunzio, LI Glazman, Steven M Girvin, Michel H Devoret, and Robert J Schoelkopf. 2011. Observation of high coherence in Josephson junction qubits measured in a three-dimensional circuit QED architecture. *Physical Review Letters* 107, 24 (2011), 240501.

[59] Alexandru Paler, Simon J Devitt, and Austin G Fowler. 2016. Synthesis of Arbitrary Quantum Circuits to Topological Assembly. *arXiv preprint arXiv:1604.08621* (2016).

[60] Alexandru Paler, Ilia Polian, Kae Nemoto, and Simon J Devitt. 2015. A compiler for fault-tolerant high level quantum circuits. *arXiv preprint arXiv:1509.02004* (2015).

[61] Shruti Patil, Ali JavadiAbhari, Chen-Fu Chiang, Jeff Heckey, Margaret Martonosi, and Frederic T Chong. 2014. Characterizing the performance effect of trials and rotations in applications that use Quantum Phase Estimation. In *Workload Characterization (IISWC), 2014 IEEE International Symposium on*. IEEE, 181–190.

[62] Massoud Pedram and Alireza Shafaei. 2016. Layout Optimization for Quantum Circuits with Linear Nearest Neighbor Architectures. *IEEE Circuits and Systems Magazine* 16, 2 (2016), 62–74.

[63] Paul Pham and Krysta M Svore. 2013. A 2D nearest-neighbor quantum architecture for factoring in polylogarithmic depth. *Quantum Information & Computation* 13, 11-12 (2013), 937–962.

[64] Robert Raussendorf. 2007. Fault-Tolerant Quantum Computation with High Threshold in Two Dimensions. *Physical Review Letters* 98, 19 (2007). https://doi.org/10.1103/PhysRevLett.98.190504

[65] Mehdi Saeedi, Robert Wille, and Rolf Drechsler. 2011. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Information Processing* 10, 3 (2011), 355–377.

[66] Daniel Sank, Evan Jeffrey, JY Mutus, TC White, J Kelly, R Barends, Y Chen, Z Chen, B Chiaro, A Dunsworth, A Megrant, P. J. J O'Malley, C Neill, P Roushan, A Vainsencher, J Wenner, A. N Cleland, and John M Martinis. 2014. Fast Scalable State Measurement with Superconducting Qubits. *arXiv preprint arXiv:1401.0257* (2014).

[67] Eyob A Sete, William J Zeng, and Chad T Rigetti. 2016. A functional architecture for scalable quantum computing. In *Rebooting Computing (ICRC), IEEE International Conference on*. IEEE, 1–6.

[68] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. 2013. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In *Proceedings of the 50th Annual Design Automation Conference*. ACM, 41.

[69] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. 2014. Qubit placement to minimize communication overhead in 2D quantum architectures. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*. IEEE, 495–500.

[70] Sarah Sheldon, Lev S Bishop, Easwar Magesan, Stefan Filipp, Jerry M Chow, and Jay M Gambetta. 2016. Characterizing errors on qubit operations via iterative randomized benchmarking. *Physical Review A* 93, 1 (2016), 012301.

[71] Sarah Sheldon, Easwar Magesan, Jerry M Chow, and Jay M Gambetta. 2016. Procedure for systematically tuning up cross-talk in the cross-resonance gate. *Physical Review A* 93, 6 (2016), 060302.

[72] Peter W Shor. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*. IEEE, 124–134.

[73] Tom Simonite. 2016. Google's quantum dream machine. *TECHNOLOGY REVIEW* 119, 2 (2016), 17–19.

[74] Andrew Steane. 1997. Space, time, parallelism and noise requirements for reliable quantum computing. *arXiv preprint quant-ph/9708021* (1997).

[75] M. Suchara, J. Kubiatowicz, A. Faruque, F. T. Chong, C. Y. Lai, and G. Paz. 2013. QuRE: The Quantum Resource Estimator toolbox. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*. 419–426. https://doi.org/10.1109/ICCD.2013.6657074

[76] Maika Takita, AD Córcoles, Easwar Magesan, Baleegh Abdo, Markus Brink, Andrew Cross, Jerry M Chow, and Jay M Gambetta. 2016. Demonstration of weight-four parity measurements in the surface code architecture. *Physical Review Letters* 117, 21 (2016), 210505.

[77] Darshan D. Thaker, Tzvetan S. Metodi, Andrew W. Cross, Isaac L. Chuang, and Frederic T. Chong. 2006. Quantum Memory Hierarchies: Efficient Designs to Match Available Parallelism in Quantum Computing. In *ISCA*. IEEE Computer Society, 378–390.

[78] Rodney Van Meter, Thaddeus D Ladd, Austin G Fowler, and Yoshihisa Yamamoto. 2010. Distributed quantum computation architecture using semiconductor nanophotonics. *International Journal of Quantum Information* 8, 01n02 (2010), 295–323.

[79] Andreas Wallraff, David I Schuster, Alexandre Blais, L Frunzio, R-S Huang, J Majer, S Kumar, Steven M Girvin, and Robert J Schoelkopf. 2004. Strong coupling of a single photon to a superconducting qubit using circuit quantum electrodynamics. *Nature* 431, 7005 (2004), 162–167.

[80] James D. Whitfield, Jacob Biamonte, and Alan Aspuru-Guzik. 2010. Simulation of electronic structure Hamiltonians using quantum computers. *Molecular Physics* 109, 5 (2010), 735.

[81] Mark Whitney, Nemanja Isailovic, Yatish Patel, and John Kubiatowicz. 2007. Automated generation of layout and control for quantum circuits. In *Proceedings of the 4th international conference on Computing frontiers*. ACM, 83–94.

[82] William K Wootters and Wojciech H Zurek. 1982. A single quantum cannot be cloned. *Nature* 299, 5886 (1982), 802–803.