Fault-tolerant interfaces for modular quantum computing on diverse qubit platforms

Frederik K. Marqversen [1,2] Gefen Baranes,[3,4] Maxim Sirotin [3,4] and Johannes Borregaard [3,*]
[1]*Department of Physics and Astronomy, Aarhus University, DK-8000 Aarhus C, Denmark*
[2]*Kvantify ApS, DK-2300 Copenhagen S, Denmark*
[3]*Department of Physics, Harvard University, Cambridge, Massachusetts 02138, USA*
[4]*Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts 02138, USA*

(Received 24 October 2025; accepted 18 March 2026; published 30 April 2026)

Modular architectures offer a scalable path toward fault-tolerant quantum computing by interconnecting smaller quantum processing units provided that high-rate, fault-tolerant interfaces can be realized across modules. We present a comprehensive analysis and comparison of known methods for establishing such interfaces, including lattice surgery, transversal gates, and grow-and-distill protocols based on code growing and logical distillation. Using the surface code, we identify optimal interface strategies across a wide range of hardware parameters, such as gate fidelities, entangling rates, and memory resources, and estimate the requirements to achieve logical error rates of 10^{-6} and 10^{-12}. Our results establish when the interface becomes a bottleneck in the computation and provide guidance for experimental implementations with superconducting, atomic, and solid-state hardware.

DOI: 10.1103/ds45-fm9n

I. INTRODUCTION

Quantum computing has made remarkable advancements in recent years, achieving systems with hundreds of physical qubits and demonstrating the first successful implementations of quantum error correction (QEC). This progress has been driven by various qubit platforms including atomic [1–7] and superconducting qubits [8–10]. Despite these advances, the full potential of quantum computing remains out of reach. Many of the transformative applications envisioned for quantum computers, including breakthroughs in materials design [11], quantum chemistry [12], and cryptography [13], still require capabilities far beyond those of current devices. State-of-the-art quantum algorithms demand millions of qubits and substantially lower error rates than currently demonstrated to outperform classical computing [13–16].

The complexity and cost of building a monolithic quantum computer capable of running fault-tolerant algorithms with millions of qubits can be prohibitive. A promising alternative is a modular quantum computer that interconnects smaller quantum processing units (QPUs) with independent control to simplify engineering and reduce costs. Computations across QPUs can then be performed by establishing qubit entanglement, enabling the transfer of quantum states and gate operations via quantum teleportation [17,18].

To enable reliable computation on modular quantum computers, fault-tolerant operation is required not only within QPUs but also across their interfaces, necessitating the use of QEC. By encoding logical qubits in physical qubits, QEC allows fault-tolerant logical operations throughout the system. Recent studies have explored several strategies for establishing fault-tolerant interconnects between QPUs through the generation of logical Bell pairs, including transversal gates [19], lattice surgery [20], and logical distillation [21]. These approaches impose different requirements on inter-QPU entangling rates, the fidelity of physical Bell pairs, and the qubit overhead necessary to achieve fault tolerance. Given the diversity of quantum computing hardware and the wide variation in their physical parameters, it remains an open and timely question, which approaches are most promising for different qubit platforms.

In this work, we address this question by introducing efficient grow-and-distill protocols for logical distillation using code growing techniques [22] and providing a comprehensive comparison with other known methods for establishing fault-tolerant interfaces. Considering the surface code [23], we perform a cross-method optimization that incorporates combinations of physical Bell-pair distillation, lattice surgery, transversal entangling gates, logical distillation, and code growing to determine which approaches maximize the logical Bell-pair generation rate across a broad range of hardware parameters. These parameters include qubit decoherence times, local gate speeds and fidelities, network entangling rates and error rates, and the qubit resources available within QPUs. Through this analysis, we identify the operational regimes in which each method is optimal, providing a reference for experimental implementations of scalable, fault-tolerant modular quantum computing across diverse platforms, including superconducting [24,25], atomic [7,18], and solid-state qubits [26,27].

II. GROW AND DISTILL

We consider a distributed quantum computing architecture composed of multiple nodes, each containing a network zone

*Contact author: borregaard@fas.harvard.edu

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

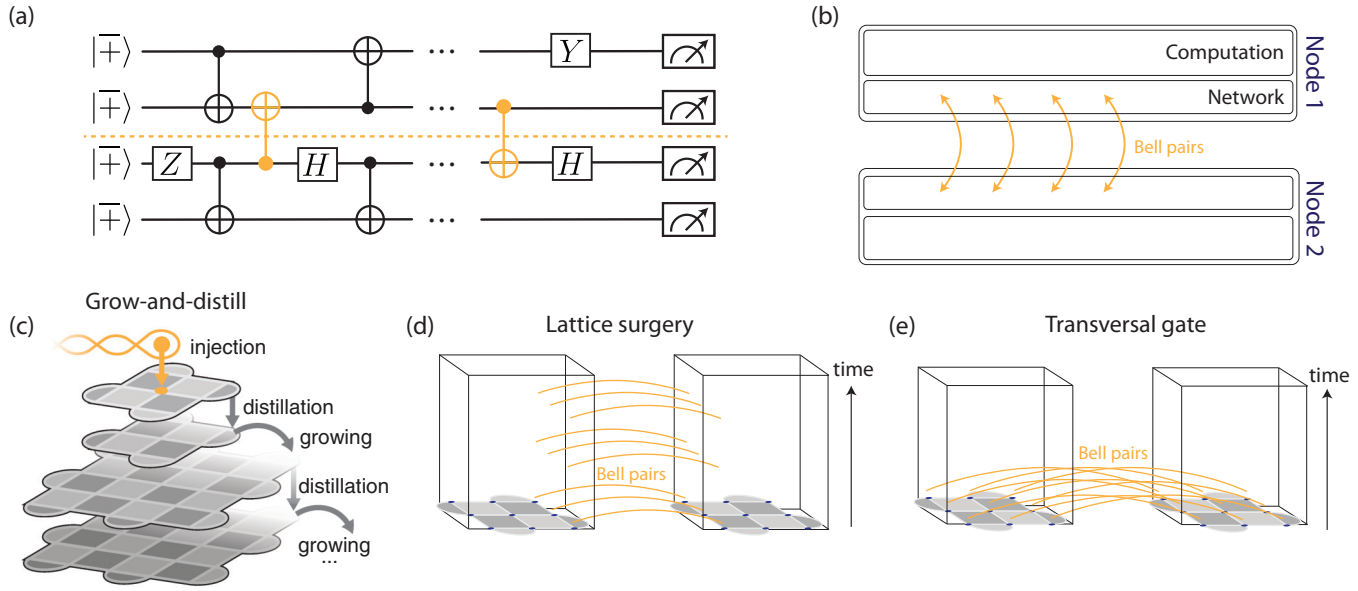


FIG. 1. (a) Distributed algorithm between two nodes, requiring local logical gates (black) as well as distributed logical gates (orange), both with high fidelity. (b) Physical implementation of two nodes, each with a computation zone and a network zone, with physical Bell pairs distributed between both nodes. We are assuming that there is a direct quantum connection between qubits in network and computation zones. Three main approaches for fault-tolerant distributed quantum computation (QC) are presented in panels (c) and (d). (c) Injection and logical distillation are interleaved with growing (grow-and-distill). (d) Lattice surgery. (e) Transversal gate.

and a computation zone, as illustrated in Figs. 1(a) and 1(b). The network zone is dedicated to generating physical Bell pairs and enabling fault-tolerant distributed logical gates, while the computation zone executes the fault-tolerant quantum algorithm. Figures 1(c)–1(e) showcase the three main approaches for distributed fault-tolerant logical gates on the surface code: logical distillation, lattice surgery, or transversal gates with physical Bell pairs [18,20,21].

Previous work on logical distillation [21] showed that it is possible to generate high-fidelity logical Bell pairs from only a few noisy physical Bell pairs. First, state injection techniques are employed to inject noisy physical Bell pairs into logical qubits. Then, quantum error detection at the logical level is used to distill high-fidelity logical Bell pairs via fault-tolerant Clifford operations and measurements. This approach allows for entanglement distillation with a constant encoding rate and supports fault-tolerant logical gates using only a small number of noisy physical Bell pairs. However, it demands a large memory buffer for the logical encoding at each node.

To reduce the memory requirements, we introduce a grow-and-distill method where stages of code growing are interspersed with distillation steps. This allows for the early distillation stages to be performed on small noisy logical qubits. The reduction in logical qubit size directly results in a reduction in space and the effect is amplified by the fact that earlier stages of distillation have much higher throughput and more aggressive postselection. Slightly reducing the size of earlier stages thus reduces the total qubit overhead significantly. Due to the smaller logical code sizes leading to larger logical error rates (LERs), it is conceivable that logical gate errors would play a more significant role than assumed in Ref. [21]. However, we find that the logical error rates do not affect the distilled error rates much as long as they are less than Bell state errors.

Formally, we describe a grow-and-distill sequence S as consisting of a sequence of stages (S_i). The very first stage is always state injection, which injects the physical Bell pair into a small surface code. Apart from the first stage, stages can be of two types: logical distillation and code growing. Logical distillation is done as in Ref. [21] using any error correcting code $[[n, k, d]]$. In this work, we include the same code set as was used in that original work, which includes 512 distinct codes up to size $n = 30$. Code growing is performed as described in Ref. [22]. Taking as input a surface code logical qubit, two code patches are added by lattice surgery to finally output a surface code of a bigger size. We estimate that the logical error rate from code growing is approximately twice that of local logical gates on the initial code distance. Details on the procedure and numerical validation of this model are discussed in Appendix C.

A. Error bounds

We use analytical bounds to evaluate the performance of the grow-and-distill sequence. As such, we provide lower performance bounds rather than approximate optimal values. The relevant bounds for this work are similar to those introduced in Ref. [21], but we have to explicitly take into account the logical gate errors due to the smaller-sized logical codes.

Consider a distillation stage using general error correcting code $[[n, k, d]]$. Using the general parallelized unencoding circuit presented in the Appendix of Ref. [21], we know that unencoding can be performed by a depth D circuit, where

$$D = 3n - 2 - k. \tag{1}$$

Note that this circuit is applicable to a general error correcting code. By specializing the unencoding for each code type, circuits of much smaller depth can be achieved.

Let the surface code size of the logical qubits be L with logical gate error rate p_L. Given input states with error rates p_{in}, the probability that the output of the distillation stage must be discarded p_{fail} is bounded by the probability that all the inputs have no errors and that no errors happened during the depth D unencoding:

$$1 - p_{\text{fail}} \geqslant (1 - p_{\text{in}})^n (1 - p_L)^{nD} \equiv (1 - q)^n. \tag{2}$$

Here, we define the error rate q, which is the probability that any specific qubit after unencoding has an error.

The output from a distillation stage after postselection can include a logical error only if that error was not detected. The probability of such an error p_{out} is thus equal to the probability of the occurrence of an undetectable error. For a distance d code, this requires at least d individual errors. The probability that no error was detected will be at least the probability of no errors occurring. This gives the following bound on p_{out}:

$$p_{\text{out}} \leqslant \frac{\Pr(|E| \geqslant d)}{\Pr(|E| = 0)} = \frac{1 - \sum_{i=0}^{d-1} \Pr(|E| = i)}{\Pr(|E| = 0)}, \tag{3}$$

where $|E|$ is the number of errors after unencoding but before postselection. A final bound is obtained by doing the substitution

$$\Pr(|E| = i) \to \binom{n}{i} q^i (1 - q)^{n-i}. \tag{4}$$

By comparison with the bounds presented in Ref. [21], it is evident that the bounds presented here also can be obtained by the direct substitution $p_{\text{in}} \to q$. The error rate q can thus be interpreted as an effective error rate that takes into account all of the effects of local logical gate errors.

The error rate of logical gates p_L performed between surface code encoded qubits of size L is modeled as

$$p_L \propto \left(\frac{p_b}{p_b^*}\right)^{\frac{L}{2}}, \tag{5}$$

where p_b is the error rate of local gates on physical qubits. For the values of the bulk threshold p_b^* and proportionality constant, we take the numerically estimated values presented in the Supplemental Material of Ref. [20].

B. Distillation throughput

Given a distillation sequence S and an amount of space/memory M that is allocated for networking, we wish to determine the expected rate at which successfully distilled Bell pairs can be produced. There are two regimes to consider, input-limited and memory-limited.

Consider initially a case where memory is not the limiting factor $M \to \infty$. The optimal pipeline strategy is to fully parallelize the distillation process, meaning that we allow multiple unencoding circuits for each stage to be running simultaneously. We refer to each actively running circuit as a stage instance. In practice, this is simply achieved by immediately initializing instances when enough inputs are available. This is called a balanced pipeline since instances never have to be actively paused to not exceed the memory constraint. In steady state, the outputs will be distilled at a rate

$$r_S^{\text{out}} = E_S r_{\text{bell}}, \tag{6}$$

where E_S is the *encoding rate* of sequence S. The encoding rate is equal to the expected number of successfully distilled logical Bell pairs per input physical Bell pair. The encoding rate depends only on the error rate of the input physical Bell pairs and on the local physical gate error rate. Thus, for a given hardware, E_S is an intrinsic quality of the given sequence S.

It turns out that for many pipeline problems, the above strategy can be applied even in memory-limited cases. The balanced pipeline will take up some amount of memory, and the exact amount will depend both on the details of S and the input rate r_{bell}. Given a memory constraint M, the idea is to artificially reduce the input rate r_{bell} such that the balanced pipeline satisfies the constraint. In practice, this means that there exists an upper *input rate cap* C_S restricting the rate at which inputs can be supplied,

$$r_S^{\text{in}} = \min(r_{\text{bell}}, C_S). \tag{7}$$

The value C_S is intrinsic to the sequence S in an equivalent fashion to E_S.

Using this r_S^{in}, the pipeline is balanced in both the input-limited and memory-limited cases, and so outputs will be produced at a rate

$$r_S^{\text{out}} = E_S r_S^{\text{in}} = E_S \min(r_{\text{bell}}, C_S). \tag{8}$$

The implications are that for a given sequence S and fixed memory constraint M, we can expect the distillation rate to increase linearly with the rate at which physical Bell pairs can be prepared. This will hold only to a point $r_S^{\text{in}} \leqslant C_S$, above which inputs have to be discarded or redistributed due to insufficient space, and the distillation rate plateaus.

How E_S and C_S are computed for a given sequence S along with other general details on this subject is presented in Appendix A.

C. Growing versus no growing

In Fig. 2, we compare the performance of distillation with code growing to distillation without code growing for a target Bell-pair error rate 10^{-12}. The comparison is based on a physical Bell-pair rate equal to the local physical gate rate, initial physical Bell-pair error rate of 1%, physical gate error rate of 0.1%. Sequences without growing are evaluated by direct injection into a distance 3 surface code followed by a growing stage, immediately growing the code to the target logical qubit size.

Both of the curves in Fig. 2 experience a lower cutoff point, below which the available space is insufficient to run distillation. Also, both curves exhibit kinks. This is because both curves are the result of multiple distinct distillation sequences. The gradual increases in distillation rate are due to more parallelization of a fixed distillation sequence. The kinks, on the other hand are the result of an optimal sequence becoming available. Finally, the plateaus observed for the upper curve are the results of hitting the limit on the input rate for the currently best sequence. Note that the final plateau is not actually a fundamental upper limit on the

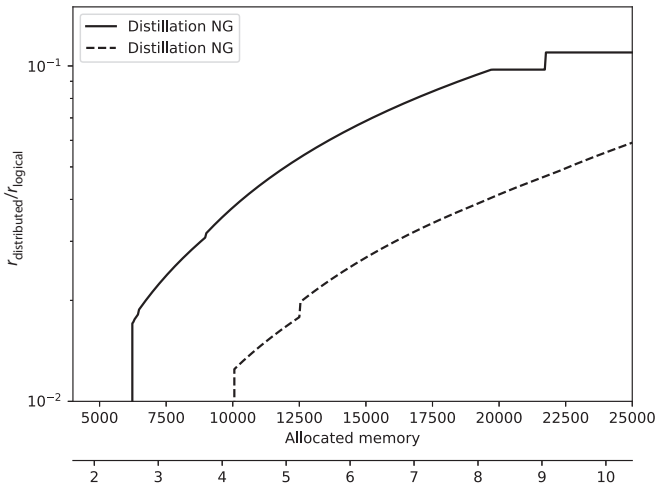


FIG. 2. Rate of distributed logical Bell pairs $r_{\text{distributed}}$ as a function of local memory dedicated to distillation. The two x axes represent the amount of allocated space for distillation in terms of physical qubits (upper) and logical qubits (lower), respectively. Includes rates from both distillation with code growing and distillation without code growing (labeled as Distillation NG in the figure) with a target Bell-pair error rate of 10^{-12}. The figure is made assuming physical Bell pairs are produced at a rate equal to the time of local physical gates. Other parameters are initial physical Bell-pair error rate of 1%, physical gate error rate of 0.1%, and idling errors per physical gate of 10^{-6}.

distillation rate. Provided more memory, a sequence with a higher encoding rate might become available, which then will allow further increasing the distillation rate, resulting in a second step in the curve.

From Fig. 2, we observe that distillation with growing is possible at space $\geqslant$6230 physical or equivalently $\geqslant$5.16 logical qubits, much lower compared to distillation without code growing possible at space $\geqslant$10 000 physical or $\geqslant$8 logical qubits. Also, we find that by including growing, networking rate is improved by at least a factor of 1.85 across Fig. 2.

III. DIRECT-GATE METHODS

Alternatives for implementing nonlocal connections are implementing direct distributed gates with the noisy Bell pairs, either using lattice surgery or transversal gates, as described in Refs. [18,20] and illustrated in Fig. 1. In the lattice surgery approach, two surface code patches are merged across a shared boundary using a set of stabilizer measurements that span the seam. These intermodule parity checks are performed using noisy Bell pairs and additional local operations, effectively stitching the patches into a larger code during the logical operation. Alternatively, transversal gate schemes apply logical gates by teleporting physical-level operations across aligned surface code patches using distributed Bell pairs. Each qubit in one logical patch is paired with a corresponding qubit in the other, and a teleported entangling gate is performed transversally using a network of physical Bell pairs. Recent results [20] show that fault-tolerant logical gates can be implemented in this way even when the Bell pairs exhibit error thresholds as high as 10%, provided that the local gates remain below the bulk surface code threshold (of 1% for the surface code). This method leverages the intrinsic robustness of the QEC codes to boundary noise.

A. Logical Bell error

To get a fair comparison between all three models considered in this work, we utilize the entire amount of memory that is allocated when modeling the throughput of the transversal and lattice surgery procedures. This is done by parallelizing the procedures as much as possible within the constraints of the given space and physical Bell-pair rate. Furthermore, the execution time is important for the final throughput. Transversal gates require only a single round of syndrome extraction by utilizing correlated decoding [28,29]. Lattice surgery, on the other hand, requires the full L rounds for an $L \times L$ surface code since the values of the measured syndromes within the seam between qubits are used for doing logic [23]. Together with qubit idling, this is a significant difference. A transversal gate requires a total of L^2 physical Bell pairs before it can be executed, whereas lattice surgery can be done in L steps of L Bell pairs. This means that the Bell pairs for lattice surgery are idling less than those for transversal gates, leading to smaller errors along the seam.

To model logical errors introduced by the direct distributed gate procedures, we use [20]

$$p_L \propto \left(\frac{p_s}{p_s^*}\right)^{\frac{L}{2}} + \left(\frac{p_b}{p_b^*}\right)^{\frac{L}{2}} + \sum_{j=1}^{L} \left(\frac{p_s}{p_{1s}^*}\right)^{\frac{j}{2}} \left(\frac{p_b}{p_b^*}\right)^{\frac{L-j}{2}}, \quad (9)$$

with

$$p_{1s}^* = p_s^* \left(1 + \alpha_C p_b \frac{\sqrt{p_s^*}}{1 - \sqrt{p_b/p_b^*}}\right)^{-2}. \quad (10)$$

Here, p_b is the local physical gate error rate and p_s is the error rate of the distributed physical Bell pairs including idling errors. For the bulk threshold p_b^*, seam threshold p_s^*, and α_C, we use numerically determined values from the Supplemental Material of Ref. [20]. Logical errors introduced by *local* logical operations during distillation simply correspond to the special case $p_s = 0$.

If the network is noisy, the local logical qubit size must be large enough to protect against the elevated error rates along the seam. In general, the direct distributed gate procedures require the local nodes to operate on qubits that are larger than required merely by local constraints. In our analyses, we find the smallest code size, which can accommodate the sought-after logical distributed error rate.

B. Idling errors

When comparing the performance of our distillation protocol to distributed transversal gates and lattice surgery, we must consider the fact that for the direct-gate methods, a large number of physical Bell pairs are needed before the operation can be executed. The time it takes to prepare these will in many regimes be significant compared to idling error rates, and so, these effects must be accounted for. The same can be said in regard to the distillation of physical qubits. We model idling errors as a single-qubit depolarization channel

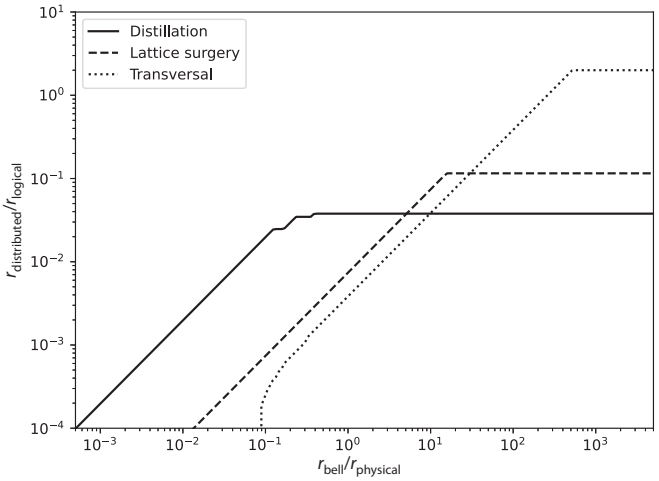


FIG. 3. Rate of distributed logical Bell pairs $r_{\text{distributed}}$ as a function of physical Bell-pair rate r_{bell}. These are given in units of local logical gate rates and physical gate rates, respectively. Rates from each of the three methods: distillation, logical cnot by lattice surgery, and transversal logical cnot are included. The figure is for initial physical Bell-pair error rate of 1%, a target Bell-pair error rate 10^{-12}, with allocated memory of 10 000 physical qubits, physical gate error rate of 0.1%, and idling errors per physical gate of 10^{-6}.

with error probability equal to that of the entire idling duration across all input qubits. Some qubits will be idling for less, all the way down to no idling for the final input qubit. Thus, this model is an overestimate of the actual effects of idling.

For logical qubit distillation, on the other hand, Bell pairs are injected into surface codes as soon as they arrive, offering protection against such idling errors. For later stages of a distillation sequence, where idling will be significantly larger than for earlier stages, the code size also will be larger. We thus model logical distillation without taking into account errors due to idling.

IV. HARDWARE AGNOSTIC METHOD COMPARISON

To describe the interplay between the different methods, we include Fig. 3, which shows, for each of the three methods, the rate at which logical Bell pairs are produced $r_{\text{distributed}}$, equivalently the rate at which logical distributed gates can be performed. The figure is made assuming an initial physical Bell-pair error rate of 1%, a target Bell-pair error rate 10^{-12}, with allocated memory of 10 000 physical qubits, physical gate error rate of 0.1%, and idling errors per physical gate of 10^{-6}.

The distillation curve has three different regimes. From the left: a linear ramp, a step, and then a constant plateau. For a particular amount of space, there will be some sequence with the largest encoding rate. This sequence will always be optimal so long as the sequence is not memory limited. This is the reason for the initial linear ramp extending to the left. Similarly, there will be some sequence with the largest distillation rate when memory is limited. This means that we can talk about input- and memory-limited distillation in general, not just of individual sequences. These observations are properly formalized in Appendix B. The step is the result of a single sequence, which is optimal in the "in between" regime. Here, we have only a singular step, although in general there could be multiple.

The curves for lattice surgery and transversal gate in Fig. 3 are very similar. They both plateau because of the memory constraint. Lattice surgery plateaus earlier due to the required L rounds of syndrome extraction, which leads to a long execution time. To the left of the plateau both curves first ramp down linearly, but then tapers off before finally reaching a point where they drop to zero. The tapering is simply a result of increasing logical qubit size required to reach the target Bell-pair error rate of 10^{-12}. The final cutoff point, left of which the target error rate cannot be reached, is the point where the input rate is so low that the accumulated idling errors bring the initial physical Bell-pair error rate above the threshold of the surface code. Lattice surgery has a lower cutoff point due to the fact that idling is only accumulating in the time it takes for L physical qubits to arrive, whereas for transversal gates the qubits are idling for the time of the full L^2 qubits.

V. ANALYSIS ACROSS HARDWARE PLATFORMS

Several physical platforms are under active development for distributed quantum computation, combining local quantum logic with photonic entanglement generation across remote nodes. Among them, we consider three representative architectures: (1) neutral atoms in optical tweezers, (2) group IV solid-state defects in nanophotonic cavities, and (3) superconducting qubits with microwave links. These platforms differ significantly in connectivity models, gate mechanisms, communication fidelities, and memory scaling. Here, we summarize the key assumptions made for each platform in our resource-performance analysis; full benchmarking data and references are detailed in Appendix D.

Neutral atoms. This architecture uses individual neutral atoms (e.g., Rb or Yb) trapped in optical tweezer arrays, with laser-driven Rydberg interactions enabling high-fidelity local gates. Each node comprises thousands of atoms partitioned into computation and communication zones, supporting all-to-all connectivity via atom movement. Current demonstrations achieve local gate errors below 0.5% and memory sizes up to 6000 qubits with continuous reloading abilities [2,7,30,31]. Distributed entanglement is generated using one of three photonic interconnects: (1) microcavities, (2) a shared cavity, or (3) free-space emission. Each method exhibits distinct scaling behavior in the required number of communication qubits versus Bell-pair generation rate [18]. We assume a total memory budget of 10^4–10^5 atoms per node, and a normalized Bell-pair rate $r_{\text{bell}}/r_{\text{physical}}$ ranging from 10^{-1} to nearly 10^3 depending on interconnect design, making this platform highly versatile for testing multiple QEC regimes. In Appendix E, we model the trade-offs between Bell-pair rates and memory requirements using the analytical expressions from Ref. [18].

Group IV defects in nanophotonic cavities. Each node in this architecture contains a diamond-hosted group IV defect (e.g., SiV) coupled to a nanophotonic cavity [26,32,33], demonstrating large-distance entanglement distribution and blind quantum computation protocols [27,33,34]. Commu-

nication is mediated by electron spins coupled to photons, while memory is stored in long-lived nuclear spins. Local logic and remote entanglement are performed via optically heralded spin-photon gates. This architecture enables photonic interconnects for both local and distributed gates, but currently suffers from limited memory. We assume a memory size of 100–5000 qubits per node and a distributed-to-local gate rate ratio $r_{\text{bell}}/r_{\text{local}}$ between 0.1 and 1, consistent with future projections [27,34].

Superconducting qubits. Superconducting quantum processors offer fast and high-fidelity local gates between qubits on the same chip [25,35]. Each node consists of one or more cryogenic chips, with interchip and internode connections possibly established via microwave links [24,36]. Unlike the other platforms, this architecture relies on lattice surgery for implementing logical operations, requiring $O(d)$ rounds of stabilizer measurements per gate [23,28,29,37]. This excludes the direct transversal gate method for the architecture. We assume a future memory capacity of 1000–10 000 qubits per node, and a distributed-to-local gate rate ratio $r_{\text{bell}}/r_{\text{local}}$ between 10^{-6} and 10^{-3}.

In our cross-platform analysis, hardware platforms are numerically characterized by the following set of five parameters:

(1) p_{physical}: error rate of a single gate performed on physical qubits;

(2) p_{bell}: error rate of distributed physical Bell pairs;

(3) $r_{\text{bell}}/r_{\text{physical}}$: maximum rate of distributed physical Bell pairs in units of the physical gate rate;

(4) p_{idle}: idling errors per one physical gate time;

(5) M: number of physical qubits (memory) allocated for networking.

Together with a target logical Bell-pair error rate p_{target}, these parameters define a space in which the three methods, distillation, lattice surgery, and transversal gates, can be evaluated. We aim to estimate where in this space the platforms described in Sec. V are likely to operate, and to describe the qualitative characteristics of each region.

A. Distributed gate rate landscape

In Fig. 4, we present two three-dimensional landscapes in which the dependent variable (shown on the third color axis) is the distribution rate of logical Bell pairs in units of the local logical gate rate, $r_{\text{distributed}}/r_{\text{logical}}$. The two free parameters are the physical Bell-pair rate $r_{\text{bell}}/r_{\text{physical}}$ and the memory allocated for networking. We fix two of the remaining parameters at reasonable future hardware values: $p_{\text{physical}} = 0.1\%$ and $p_{\text{idle}} = 10^{-6}$ (see Appendix D). The remaining parameters, p_{bell} and p_{target}, differ between the two plots. The top plot reflects an optimistic milestone with $p_{\text{bell}} = 1\%$ and $p_{\text{target}} = 10^{-12}$, while the bottom plot assumes a more conservative scenario with $p_{\text{bell}} = 5\%$ and $p_{\text{target}} = 10^{-6}$.

At each point in Fig. 4, all three networking methods are evaluated, and the maximum achievable distribution rate is plotted. This defines distinct regions where one method outperforms the others. Each platform's expected operational region as defined in Sec. V is also indicated.

The reported networking rate assumes that the entire allocated memory is used for executing one of the three methods. This memory is therefore not available for local computation. Notably, we do not account for additional memory required to achieve a given r_{bell} through, e.g., multiplexing. For a more detailed analysis including these effects for neutral atoms, see Appendix E.

For these figures, unencoding circuits are assumed to be implemented transversally, followed by a single round of syndrome extraction [28,29]. As noted in Sec. V, platforms lacking all-to-all connectivity (e.g., superconducting qubits) cannot perform transversal gates, leading to the time of performing unencoding being longer. However, our analysis including these effects shows that this has but negligible effect on distillation rates, since the dominant factor is discard rates, which remain unaffected. By defining r_{logical} as the rate of a single round of local syndrome extraction (rather than logical gate rate), we enable a consistent comparison across all three platforms in Fig. 4.

To achieve meaningful rates using direct-gate methods under the bottom-plot scenario in Fig. 4, we incorporate physical distillation. Here, the initial 5% error Bell pairs are distilled to 1% before being used by higher-level protocols. The memory cost of this distillation process is fully accounted for in the plot.

Physical distillation is analyzed similarly to logical distillation, with the important distinction that the qubits involved are not error protected. Consequently, idling errors must be considered. Since pipeline performance depends on the degree of parallelization, idling depends on available memory, slightly complicating the analysis. However, by full numerical analysis we find a simple outcome: For distilling from 5% to 1%, the optimal strategy across nearly the entire parameter space is two rounds of the classical [2, 1, 2] parity code.

B. Networking regimes across platforms

The primary difference between the two plots in Fig. 4 lies in the performance of the direct-gate methods: Both degrade more rapidly in the lower-quality case ($p_{\text{bell}} = 5\%$), making logical distillation far more dominant. This occurs because direct methods are highly sensitive to initial error rates; reaching target fidelities under poor conditions requires much larger logical codes, reducing rates significantly. Idling further amplifies these effects.

In general, transversal gates tend to outperform other methods when the physical Bell-pair rate r_{bell} is high. When r_{bell} is low and system sizes are small, lattice surgery is typically the only viable approach. For large systems operating at moderate to low values of r_{bell}, logical distillation emerges as the most effective method. In the lower-quality case, around when $r_{\text{bell}}/r_{\text{physical}} < 0.2$, physical distillation to 1% becomes infeasible. As a result, neither direct-gate method can reach the target error rate, further emphasizing the necessity of logical distillation across a large portion of parameter space.

Regarding the platforms in Sec. V, we emphasize that the figure does not compare absolute distributed gate rates, as all quantities are normalized to local rates. Instead, it illustrates the *relative impact* of modular connections. We collect our observations as follows.

Neutral atoms. The region spans all three regimes in both plots. This reflects relatively slow local gates and high poten-

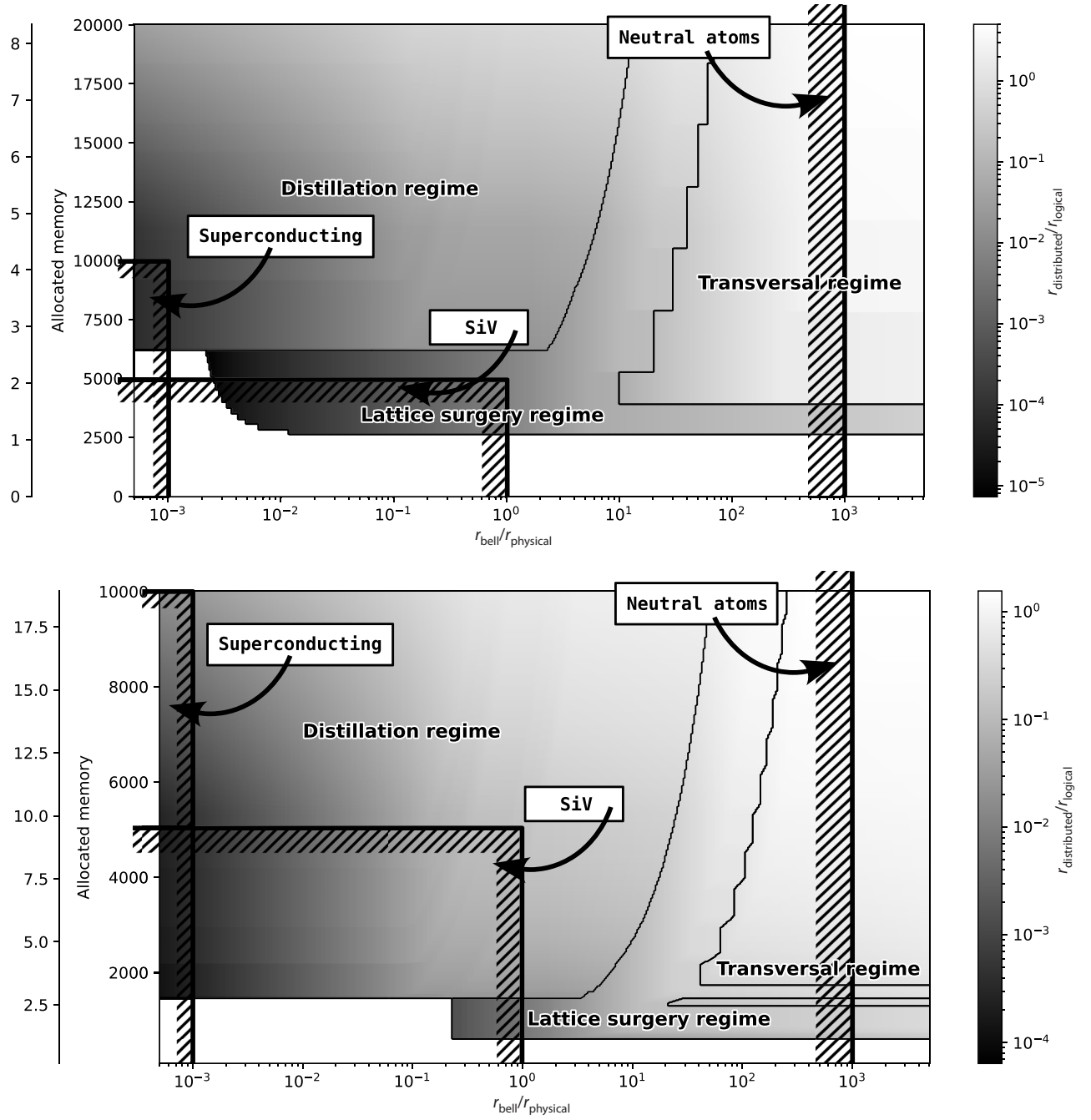


FIG. 4. Logical Bell-pair distribution rate $r_{\text{distributed}}$ as a function of networking memory and physical Bell-pair rate r_{bell}. The two plots correspond to (top) $p_{\text{bell}} = 1\%$, $p_{\text{target}} = 10^{-12}$ and (bottom) $p_{\text{bell}} = 5\%$, $p_{\text{target}} = 10^{-6}$. Rates are expressed in units of the local physical gate rate r_{physical} and local syndrome extraction rate r_{logical}. The y axes show both physical (right) and logical (left) qubit counts. Each point reflects the highest rate achievable among the three methods: distillation, lattice surgery, and transversal gates. Black contour lines divide regions with different optimal methods. The full-white regions extending from the bottom left represent parameter regimes where no method achieves the target error rate. Discrete jumps are due to the discrete nature of surface code sizes and distillation protocols, and are not artifacts of resolution. Expected operational regions for neutral atoms, SiV, and superconducting qubits are outlined, each extending left and downward from the indicated boundaries.

tial for Bell-pair multiplexing, leading to high $r_{\text{bell}}/r_{\text{physical}}$. Modular approaches can match or exceed local performance, making this platform suited for distributed architectures.

Superconducting qubits. Fast local gates shift the region to the left, where only distillation is viable due to severe idling penalties. This suggests that while limited modularity is acceptable, particularly when only a few distributed gates are needed, scaling up the number of modular links quickly turns into a bottleneck. In such cases, the slower distributed operations lag behind fast local gates, negatively impacting performance.

SiV cavities. This hardware platform is characterized by similar parameters for local and nonlocal gates. For high-quality Bell pairs, lattice surgery emerges as the only feasible method while for lower-quality input, logical distillation dominates. Since the physical Bell-pair rate is bounded by the local gate rate, transversal gates are not suitable for this hardware.

VI. CONCLUSION

In summary, we have introduced a grow-and-distill sequence reducing qubit overhead by several thousand while enhancing logical Bell-pair generation rates compared to previous approaches to logical Bell-pair distillation. Additionally, we have developed a systematic framework for cross-method comparison and optimization of fault-tolerant interfaces between QPUs including lattice surgery and transversal gate techniques. Focusing on the surface code, we mapped the regimes of qubit coherence times, local gate fidelities and speeds, network entanglement rates and fidelities, and intra-QPU qubit resources where grow-and-distill, lattice surgery, and transversal gates maximize logical Bell-pair generation. By adopting future projected parameter regimes for superconducting, atomic, and solid-state platforms, we identified both the required number of networking qubits and the optimal interface strategies to reach target logical error rates of 10^{-6} and 10^{-12}.

These results provide a reference for experimental prioritization and architectural design in scalable modular quantum computing. Specifically, for the surface code, matching intermodule operation rates to intramodule gates will typically require several thousand networking qubits and physical entangling rates up to 3 orders of magnitude faster than local gate speeds. These stringent requirements are largely dictated by the encoding rate of the QEC code, suggesting that adopting more efficient codes such as QLDPC codes [38,39], which maintain high error thresholds while improving the logical encoding ratio could significantly lower the requirements for modular computation. A higher logical encoding rate would reduce the qubit overhead of the grow-and-distill techniques and similarly decrease the entanglement distribution rates required for transversal gates, since fewer physical qubits are needed per logical qubit. Consequently, while we expect the overall trend of transversal gates being optimal for high entangling rates, lattice surgery for intermediate rates, and grow and distill for slower rates to persist, as shown in Fig. 4, the axis would shift toward lower memory and physical entangling rates.

Our analysis is intentionally platform agnostic, focusing on universal design principles that offer a reference for future hardware developments. A natural next step would be to incorporate more specific hardware constraints into the optimization. For instance, in our current simulations, we assume fixed rates and error parameters for local gates and syndrome extraction as the size of the qubit memories increased. In practice, however, larger qubit memories may require longer intra-QPU transport times and could introduce additional errors affecting the performance. While beyond the scope of this work, exploring these effects in detail would be an interesting direction for future study.

ACKNOWLEDGMENTS

We would like to thank Mikhail Lukin, Josiah Sinclair, Madelyn Cain, and David Levonian for helpful and stimulating discussions and general support. We gratefully acknowledge support from Innovation Fund Denmark under Grant No. 1063-00046B—"PhotoQ Photonic Quantum Computing" and The AWS Quantum Discovery Fund at the Harvard Quantum Initiative. G.B. acknowledges support from the MIT Patrons of Physics Fellows Society.

DATA AVAILABILITY

The data that support the findings of this article are openly available [40,41]. The code used for producing our numerical results as well as that used for analyzing and plotting of the data is available in Ref. [40], and the code used for validating our model of surface code growing and Bell-pair injection is available in Ref. [41].

APPENDIX A: BALANCED PIPELINE FOR ENTANGLEMENT DISTILLATION

Here, we derive the formulas needed for computing the distillation rate of a given distillation sequence.

In steady state, the optimal pipeline is such that instances of each stage of the sequence are run as soon as the required inputs have been prepared. This means that each stage of the sequence only ever waits because of insufficient inputs and not because of insufficient memory. This is called the balanced pipeline. In many cases, ours included, the output rate r_{out} as well as the memory average usage of actively running processes M_S^{active} of a balanced pipeline are directly proportional to the input rate. Taking into account the average space taken up by idling input qubits M_S^{idle}, we have

$$r_{\text{out}} = E_S r_{\text{in}} \tag{A1}$$

and

$$M_{\text{total}} = M_S^{\text{idle}} + M_S^{\text{active}} = M_S^{\text{idle}} + M_S r_{\text{in}}, \tag{A2}$$

where E_S, M_S^{idle}, and M_S are constants determined by the details of the given sequence S. If the available memory is limited, we find that we must choose

$$r_{\text{in}} \leqslant \frac{M_{\text{total}} - M_S^{\text{idle}}}{M_S} \equiv C_S. \tag{A3}$$

During operations, random fluctuations will mean that the space taken up by the instances of a stage might be larger than the average, but simultaneously others will take up less. This effect in addition to the buffer that is made available through M_S^{idle}, makes cases where the balanced pipeline will hit the memory cap exceedingly rare. Thus, to a good approximation we can take equality in the above equation and can still expect the output rate to be given by $r_{\text{out}} = E_S r_{\text{in}}$. This we also confirm empirically by Monte Carlo simulation (see Ref. [40]).

Obviously, the input rate cannot exceed the rate at which physical Bell pairs can be provided r_{bell}. So, we get two regimes: input-limited and memory-limited:

$$r_{\text{out}} = E_S r_{\text{in}} = E_S \min(C_S, r_{\text{bell}}). \tag{A4}$$

This is the general result presented in the main text.

In the remainder of this section, we derive the specific formulas for E_S, M_S, and M_S^{idle} given an arbitrary distillation sequence S. A balanced pipeline should essentially be viewed as a fully parallelized process with no memory constraint. This is exactly the regime analyzed in the Appendixes of the original work on entanglement distillation [21]. We will in large use the notation and results from there.

A distillation sequence S is a sequence of stages (S_i). Each stage, S_i, that being code distillation, growing, or injection, can be described by the following parameters: The code parameters $[[n_i, k_i, d_i]]$; the rejection rate p_i^{fail}, which in general depends on, and is thus calculated from, the output error of the previous stage p_{i-1}^{out}; the execution time T_i; and the logical qubit size s_i in terms of physical qubits.

At each stage i of a sequence S, we define the encoding rate E_i, which describes the expected ratio of logical Bell-pairs output from stage i to initial physical Bell pairs, and the number K_i of logical qubits that the stage outputs per successful distillation attempt:

$$E_i = \prod_{j=0}^{i}(1 - p_j^{\text{fail}})\frac{k_j}{n_j} \quad \text{and} \quad K_i = \prod_{j=0}^{i} k_i. \tag{A5}$$

Since instances are started as soon as enough inputs are prepared, this means that the output rate of stage i is $r_i^{\text{out}} = (1 - p_i^{\text{fail}})\frac{k_i}{n_i} r_{i-1}^{\text{out}}$. This immediately leads to

$$r_{\text{out}} = E_S r_{\text{in}}, \quad \text{where} \quad E_S = \prod_i \left(1 - p_i^{\text{fail}}\right)\frac{k_i}{n_i}, \tag{A6}$$

with E_S being the encoding rate of the complete sequence.

The average number of instances of stage i that will be running in parallel is given by

$$N_i = T_i r_i^{\text{init}} = T_i \frac{K_{i-1}}{n_i} E_{i-1} r_{\text{in}}, \tag{A7}$$

where r_i^{init} is the rate at which inputs are prepared, and r_{in} is the rate at which physical Bell pairs are supplied at the start of the whole sequence. Each instance takes up n_i qubits of size s_i, and so the average space taken up by active instances of stage i is

$$M_i^{\text{active}} = N_i n_i s_i = s_i T_i E_{i-1} K_{i-1} r_{\text{in}}. \tag{A8}$$

Since instances are initialized as soon as inputs are ready, the average space taken up by idle input qubits to stage i is

$$M_i^{\text{idle}} = \tfrac{1}{2} n_i s_i K_{i-1}. \tag{A9}$$

Summing up the contributions of all stages gives

$$M_{\text{total}} = M_S^{\text{idle}} + M_S r_{\text{in}}, \tag{A10}$$

$$\text{with} \quad M_S^{\text{idle}} = \sum_i \frac{1}{2} n_i s_i K_{i-1}, \tag{A11}$$

$$\text{and} \quad M_S = \sum_i s_i T_i E_{i-1} K_{i-1}. \tag{A12}$$

These are used for explicitly computing C_S, which is then used to evaluate the distillation rate.

APPENDIX B: DISTILLATION SEQUENCE OPTIMIZATION

1. DFS branch pruning

The objective function that we wish to maximize is the distillation rate:

$$r_S^{\text{out}} = \min(E_S C_S, E_S r_{\text{bell}}), \tag{B1}$$

which should be maximized over the space of valid distillation sequences S. This can be done by a standard depth-first search (DFS). However, the space of sequences is huge and has a very large branching factor. Clever branch pruning is thus required to make the search feasible. Here, we present the details of our strategy.

Let S and S' be two distillation sequences. Let $S + S'$ denote the sequence that is the concatenation starting with S and followed by S'. We will consider the objective function evaluated for the concatenated sequence $r_{S+S'}^{\text{out}}$. Using the definitions and results from Appendix A, we note the following relations:

$$E_{S+S'} = E_S \cdot E_{S'}(p_S^{\text{out}}, L_S) \tag{B2}$$

and

$$C_{S+S'} = \frac{M - M_S^{\text{idle}} - K_S M_{S'}^{\text{idle}}}{M_S + E_S K_S M_{S'}\left(p_S^{\text{out}}, L_S\right)}, \tag{B3}$$

where p_S^{out} and L_S are the error rate and code size of the logical Bell-pairs output from sequence S, respectively. Both of these are included explicitly since the success rates and distilled error rate of the stages of S' depend on these values. $M_{S'}^{\text{idle}}$ however is independent of S. These relations identify the fundamentally underlying parameters dictating the value of the objective function:

$$r_{S+S'}^{\text{out}} = r_{S'}\left(M_S^{\text{idle}}, M_S, K_S, E_S, p_S^{\text{out}}, L_S\right) \tag{B4}$$

$$\equiv r_{S'}(\mathbf{v}_S, L_S), \tag{B5}$$

where $\mathbf{v}_S$ is the vector of parameters excluding L_S. It turns out (to be shown) that $r_{S'}$ is a monotonous function in each of the parameters of $\mathbf{v}_S$ independent of the extension S'.

To see how the above observation is relevant, we assume, without loss of generality, that the rate is decreasing in each of the parameters. Now, consider any two sequences S and S' with $L_S = L_{S'}$ and $\mathrm{v}_{S',i} \geqslant \mathrm{v}_{S,i}$ for all i. Since the objective function is decreasing, we can immediately conclude that for any extension S'' we have

$$r_{S'+S''}^{\text{out}} \leqslant r_{S+S''}^{\text{out}}, \tag{B6}$$

and so, the branch extending from S' can be cut immediately without exploring it any further.

What is left is to show that $r_{S'}$ is monotonous in each of the parameters of $\mathbf{v}$. From the explicit form of $C_{S+S'}$, this is trivially true for the parameters M_S^{idle}, M_S, and K_S. Clearly, the input-limited case $E_S r_{\text{bell}}$ is increasing in E_S. This also holds for the memory-limited case $E_S C_S$ since

$$\frac{\partial}{\partial E_S}\frac{E_S}{M_S + E_S K_S M_{S'}} = \frac{M_S}{(M_S + E_S K_S M_{S'})^2} \geqslant 0. \tag{B7}$$

Finally, we will show that the rate is decreasing in p_S^{out}. From the definition of E_i, it follows

$$\frac{\partial E_i}{\partial p_S^{\text{out}}} = -A_i E_i \quad \text{with} \quad 0 \leqslant A_i \leqslant A_{i+1}. \tag{B8}$$

This immediately covers the input-limited case. From the definition of $M_{S'}$ and using the above result, we get

$$\frac{\partial M_{S'}}{\partial p_S^{\text{out}}} = \frac{\partial}{\partial p_S^{\text{out}}} \sum_j s_i T_i K_{i-1} E_{i-1} \tag{B9}$$

$$= -\sum_j s_i T_i K_{i-1} E_{i-1} A_{i-1} \tag{B10}$$

$$\geqslant -\sum_j s_i T_i K_{i-1} E_{i-1} A_{S'} \tag{B11}$$

$$= -A_{S'} M_{S'}. \tag{B12}$$

Putting this inequality to use, we find

$$\frac{\partial}{\partial p_S^{\text{out}}} (E_{S+S'} C_{S+S'}) \tag{B13}$$

$$= |B|[\partial E_{S'} (M_S + E_S K_S M_{S'}) - E_{S'} E_S K_S \partial M_{S'}] \tag{B14}$$

$$= -|B|[A_{S'} E_{S'} M_S + E_{S'} E_S K_S (A_{S'} M_{S'} + \partial M_{S'})] \tag{B15}$$

$$\leqslant -|B| A_{S'} E_{S'} M_S \leqslant 0. \tag{B16}$$

We thus conclude that the objective function indeed is decreasing in p_S^{out}.

2. Reducing the search space

With the pruning method described in the previous section, the DFS is feasible and somewhat fast. However, to explore the two-dimensional parameter space of Fig. 4 requires a huge number of separate optimizations.

Since the distillation rate of a sequence depends in a nontrivial way on the physical Bell-pair rate r_{bell}, so does the optimal sequence

$$S_\lambda^* = \text{argmax}_S[r_S^{\text{out}}(\lambda)] = \text{argmax}_S[\min(E_S \lambda, E_S C_S)], \tag{B17}$$

where we from now on use the simplifying notation

$$\lambda = r_{\text{bell}}. \tag{B18}$$

And so, naively each single point in Fig. 4 requires a separate search. We now discuss how this number can be reduced to get exact values for the vast majority of the space, while also providing a quite reasonable lower bound in the remainder.

Indeed, for a fixed memory constraint, there are two sequences that in general are optimal for a large range of rates. These sequences are the sequence with the largest encoding rate and the sequence with the largest distillation rate cap:

$$S' = \text{argmax}_S[E_S] \tag{B19}$$

and

$$S'' = \text{argmax}_S[E_S C_S]. \tag{B20}$$

In the limit of low Bell rates S' will always be optimal, and in the limit of high Bell rates S'' will be optimal. More precisely,

$$\lambda \leqslant C_{S'} \Rightarrow S^* = S' \tag{B21}$$

and

$$\lambda \geqslant C_{S''} \Rightarrow S^* = S''. \tag{B22}$$

These statements follow directly from the definition of $r_S^{\text{out}}(\lambda)$, S', and S''. The intuition is that when sequences are not memory limited, the distillation rate is linear in the Bell rate, and so, the sequence with the largest encoding rate S' always is optimal. However, as soon as S' becomes memory limited, $\lambda > C_{S'}$, the corresponding distillation rate caps out at $E_{S'} C_{S'}$, which is suboptimal. Equivalently, no sequence can ever provide a higher distillation rate than $E_{S''} C_{S''}$, so as soon as S'' reaches that rate $\lambda \geqslant C_{S''}$, no other sequence can beat that.

Only when λ is between $C_{S'}$ and $C_{S''}$ do these two sequences not provide the optimal distillation rate. However, they do provide lower bounds on the optimum, which significantly restrict the possibilities. Specificity, the optimal distillation rate r_λ^* must lie in a small well-defined region. Specifically, when

$$\lambda \in (C_{S'}, C_{S''}), \tag{B23}$$

then

$$\min(E_{S'} C_{S'}, E_{S''} \lambda) \leqslant r_\lambda^* \leqslant \min(E_{S'} \lambda, E_{S''} C_{S''}). \tag{B24}$$

All of these ideas are illustrated in Fig. 5. By restricting the search to only find S' and S'', we gain a really good bound on the actual optimum across all values of λ.

In making Fig. 4, we further improve on the lower bound provided by the two limiting cases by using the fact that a sequence that successfully distils Bell pairs at allocated memory M also is a valid sequence for all cases $\geqslant M$. Since "good" sequences for one value of M can be expected to be good sequences for values in the neighborhood, this strategy effectively enhances the known bound to be close to optimal, as is evident from Fig. 5. Upon applying this strategy, we find that the number of *unique* sequences that are relevant is only on the order of 10s of sequences. We therefore get close to optimal results by finding the optimal sequences for only a very modest number of distinct Ms, reducing the amount of work even further.

APPENDIX C: SURFACE CODE GROWING

We implement a surface code growing procedure based on the method introduced in Ref. [22], adapted here for the rotated surface code. This approach begins with a rotated code of distance d_{start} and grows it to a larger code of distance d_{final}. To maintain the consistency of logical operators during growth, additional patches are initialized in the $|+\rangle$ state along the right edge and in the $|0\rangle$ state along the bottom edge (see Fig. 6).

Both corner injection [22] and middle injection [42] are compatible with this growing technique. Since their (LERs and rejection rates are comparable, we use corner injection in our simulations for simplicity. Code growth is performed by executing one round of stabilizer measurements over the full d_{final} lattice. Detectors are placed on stabilizers that were part of the initial code as well as on the newly added stabilizers required for the grown patch.

We simulate this growing procedure and compare its LER to that of standard code initialization. The results are shown

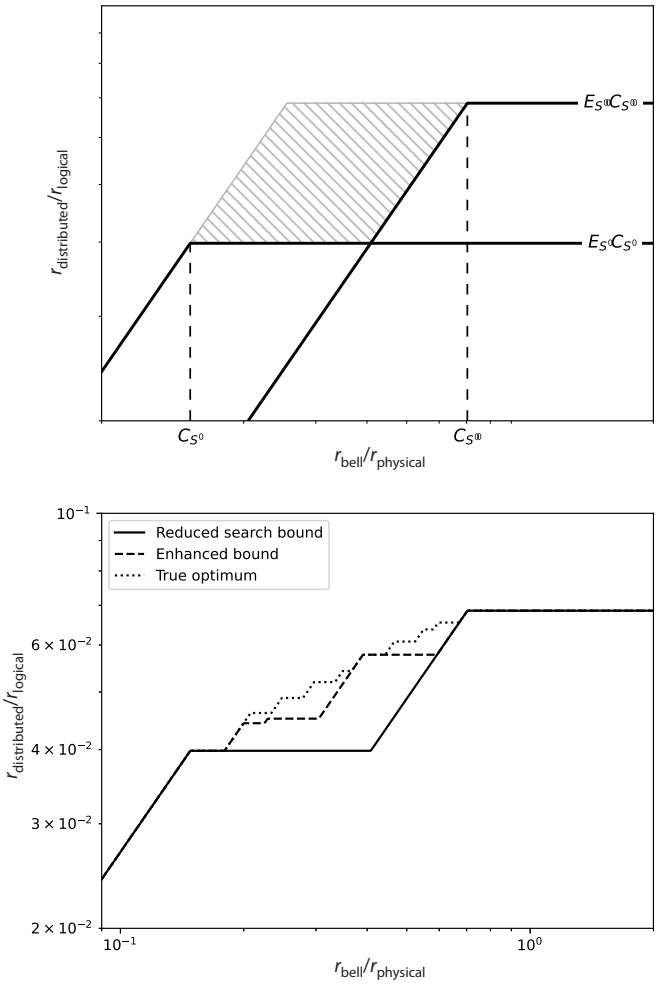


FIG. 5. (Left) The two solid lines illustrate the rates produced by the optimal sequences in the limit of low S' and high Bell-pair rate S''. Together, they provide the maximal distillation rate for most cases. Only for input rates between $C_{S'}$ and $C_{S''}$ can sequences exist that further improve on these rates. Furthermore, the optimum is known to be restricted to within the gray shaded area. (Right) The distillation rate for allocated space of 13 565 physical qubits and target error $p_{\text{target}} = 10^{-12}$. The solid line is the bound given by the reduced search. The dashed line is the enhanced bound induced by solutions for space $<$13 565, and the dotted line is the true maximum rate.

in Fig. 7. In our simulations, a noiseless d_{start} code is prepared in the logical $|0\rangle$ state using a single round of stabilizer measurements. The code is then grown to d_{final} using noisy patch initialization and noisy stabilizer measurement, followed by a logical Z_L measurement. We evaluate four growing transitions: $3 \to 5$, $3 \to 7$, $5 \to 7$, and $5 \to 9$. Simulations are performed under a biased circuit noise model in which single-qubit operations have an error rate $p_1 = p_2/10$, where p_2 is the error rate for two-qubit operations.

Our results show that the logical error introduced during code growing depends only on the initial code distance d_{start}, and is approximately twice the LER of direct initialization for a code of that same distance:

$$p_{\text{growing: } d_{\text{start}} \to d_{\text{final}}} \approx 2A_{d_{\text{start}}} \left(\frac{p_2}{p_b^*} \right)^{\left(\frac{d_{\text{start}}+1}{2}\right)},$$

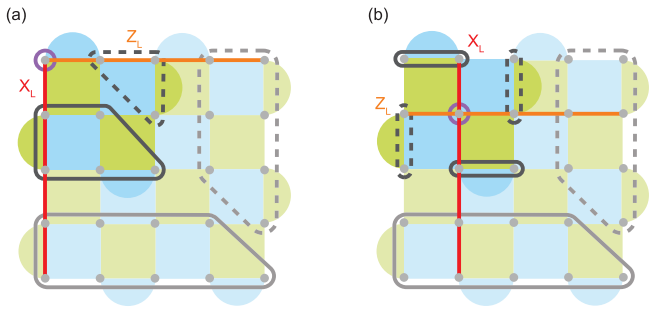


FIG. 6. Schematic of rotated surface code injection and growing from $d = 3$ to $d = 5$: (a) corner injection and (b) middle injection. Data qubits enclosed by solid lines are initialized in the $|+\rangle$ state; those with dashed lines are initialized in the $|0\rangle$ state. The purple circle denotes the injection qubit (from a Bell pair). Green and blue represent Z and X stabilizers, respectively. Logical operators Z_L and X_L are shown as an orange horizontal line and red vertical line, respectively.

where p_b^* is a bulk threshold, and constant $A_{d_{\text{start}}}$ depends only on the initial code distance d_{start} and characterizes LER of direct initialization:

$$p_{\text{init } d_{\text{start}}} \approx A_{d_{\text{start}}} \left(\frac{p_2}{p_b^*} \right)^{\left(\frac{d_{\text{start}}+1}{2}\right)}.$$

For comparison, the initialization LER is obtained by preparing a full $d_{\text{start}} = 3$ or 5 code via a single round of stabilizer measurements across the entire patch.

The simulation is performed using the Python library `Stim` [43] and is available online in Ref. [41].

APPENDIX D: OVERVIEW OF HARDWARE PLATFORMS FOR DISTRIBUTED QUANTUM COMPUTATION

Several physical platforms are under active development for realizing distributed quantum computation. Here, we describe three prominent architectures and summarize their current experimental benchmarks and future goals.

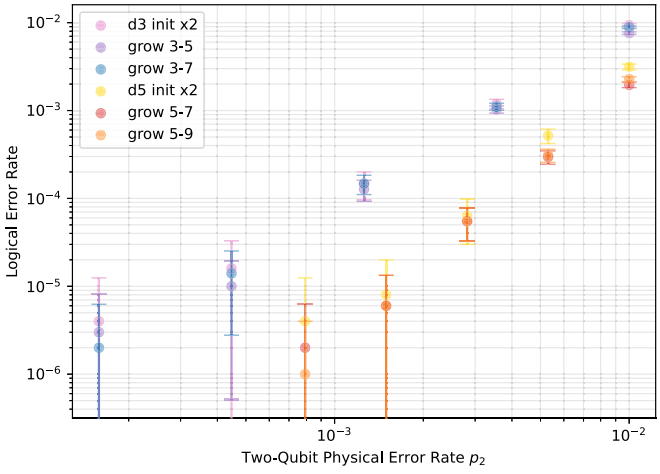


FIG. 7. Comparison of logical error rate for growing vs direct initialization of a surface code for starting distances $d_{\text{start}} = 3$ and 5.

TABLE I. Group IV defects in diamonds platform parameters. Each node uses an electron spin for communication and a nuclear spin for storage. "Local" refers to on-chip operations delegated by optical photons, while "Distributed" parameters refer to entanglement between nodes via photonic links.

Parameter	Demonstrated	Future goal
Local gate error	∼7% [26,32,33]	0.1%
Local gate time	∼50 μs [33]	1 μs
Distr. gate error	∼10% [27,33]	1%
Distr. gate rate	1 Hz [27]	100 kHz to 1 MHz
Idling error rate	0.1% per ms [26,27,33]	10^{-4} per ms
Memory size	3 qubits [33,44]	100–5000

1. Solid-state defects in nanophotonic cavities

Each network node contains a group IV defect in diamond (e.g., SiV) coupled to a nanophotonic cavity, offering strong light-matter coupling and making this a promising platform for quantum networking [26,27,32,33]. Similar to Ref. [34], we envision a platform with an ensemble of such cavities, each hosting a communication qubit (electron spin) and a long-lived memory qubit (nuclear spin). Local deterministic gates between memory qubits are mediated via communication qubits and optical photons, using a loss-tolerant architecture [34]. In this way, a distributed logical qubit can be implemented across multiple cavities, using the local photonic gates within a node. Remote entanglement is established using heralded spin-photon gates and transferred to memory qubits for storage. Since both local and distributed gates are optically mediated, we assume that in the future $r_{\text{Bell}} \leqslant r_{\text{Local}}$, meaning the distributed gate rate may approach or remain slower than the local gate rate. This platform benefits from both cavity enhancement and telecom compatibility [27] Platform parameters shown in Table I.

2. Superconducting qubits

The superconducting platform holds strong promise for quantum computing, with each node comprising multiple chips in a cryogenic environment and internode links implemented using photons [24]. Local gates between superconducting qubits on the same chip achieve high fidelities and sub-100 ns operation times [25]. Future architectures are expected to use microwave links between chips within a fridge, with slightly higher error tolerance: Local gates may operate at 0.1% error with a 1% threshold, whereas interchip gates may allow 1% error with a 10% threshold [18]. Distributed entanglement generation between fridges is expected to rely on microwave photons or transduction to optical [36,45]. Unlike the previous platforms, this architecture performs logical gates using lattice surgery, not transversally. As a result, logical two-qubit operations require $O(d)$ stabilizer measurements rather than $O(1)$, increasing the time overhead for distributed computation [28,29,37,46] Platform parameters shown in Table II.

TABLE II. Superconducting platform parameters. While local operations are extremely fast and high fidelity on superconducting qubits, the microwave links between chips or between nodes currently impose a bottleneck. Future improvements are required to realize high-rate distributed gates with superconducting processors.

Parameter	Demonstrated	Future goal
Local gate error	∼0.5% [25,35]	0.1%
Local gate time	20 ns [25,35]	∼1 ns
Distr. gate error	∼5% [47]	1%
Distr. gate rate	>1 MHz [47]	1–10 MHz
Idling error rate	1% per μs [25,35]	0.1% per μs
Memory size	∼100 [25,35]	1000–10 000

TABLE III. Neutral atom platform parameters. Communication qubits (atoms in cavities) distribute entanglement between nodes, while computational qubits in the array perform local logic. Advanced cooling and trapping techniques give very large array sizes and long-lived qubit states, supporting a scalable modular architecture.

Parameter	Demonstrated	Future goal
Local gate error	∼0.5% [49]	0.1%
Local gate time	200 μs [2]	10 μs
Distr. gate error	2% [50]	<1% [18]
Distr. gate rate	30 Hz [50]	10 kHz to 100 MHz [18]
Idling error rate	10^{-5} per μs [2]	10^{-6} per μs
Memory size	6000 [30]	10 000–100 000

3. Neutral atoms with optical cavities

Neutral atoms are a rapidly advancing platform for scalable quantum computing [2,7,31,37]. Each node consists of a large array of individually trapped atoms (e.g., Rubidium or Ytterbium), partitioned into computational and communication zones [21,34]. Local logic is performed using laser-driven Rydberg interactions, which enable high-fidelity entangling gates across many qubits. While the gate operations are fast, atomic motion currently limits overall local gate times to ∼200 μs [2]. Communication qubits couple to photonic modes to enable internode entanglement generation. Future architectures are expected to support scalable communication with high-rate photonic links and flexible memory allocations [18,48]. These trade-offs are explored in detail in Appendix E, where we analyze performance across several photonic interconnect designs with neutral atoms Platform parameters shown in Table III.

APPENDIX E: RESOURCE-AWARE PERFORMANCE LANDSCAPE FOR NEUTRAL ATOMS

The generation rate of remote Bell pairs in neutral atom systems depends critically on the architecture of the photonic interface, as well as the number of atoms allocated as communication qubits. We consider three prominent designs, each representing a distinct trade-off between parallelism, complexity, and achievable entanglement rates:

(1) 30 microcavities: Each communication atom is individually coupled to a dedicated optical cavity, enabling highly parallel photon generation. This design maximizes the

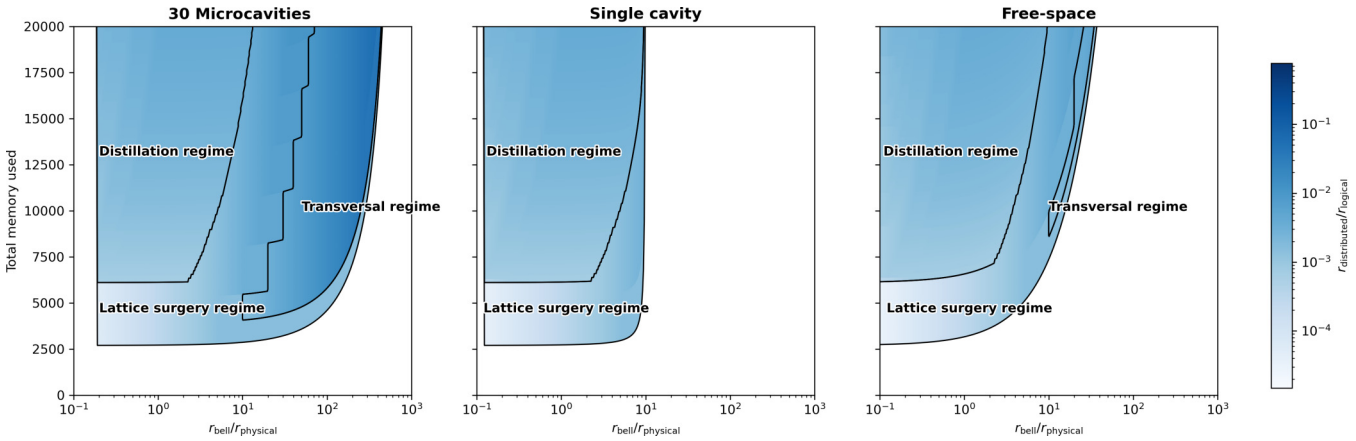


FIG. 8. Simulated resource-performance map for distributed quantum computing with neutral atoms across three interconnect architectures. The x axis shows the normalized Bell-pair rate $r_{\text{bell}}/r_{\text{physical}}$, and the y axis represents the *total physical memory* required (communication + logic). Color intensity reflects the achievable logical rate ratio $r_{\text{distributed logical}}/r_{\text{local logical}}$, with distinct regimes highlighting the best quantum error correction method out of logical distillation, lattice surgery, and transversal gates. Each panel corresponds to a different photonic interface: 30 microcavities, a single shared cavity, and free-space emission. Feasible regions (above 2500 logic qubits and below 20 000 total memory) are calculated using the analytical model from Ref. [18] based on cavity count, P_{aa}, and t_{base}, as described in the text.

Bell-pair rate per communication qubit and is projected to achieve rates up to 10^8 Hz with large arrays [18]. The overhead, however, includes the physical integration of many cavities and the associated optical alignment.

(2) Single cavity: Here, multiple communication atoms interact sequentially with a single cavity mode, enabling time-multiplexed entanglement generation [51]. This approach reduces optical hardware requirements but introduces a trade-off: The Bell-pair rate is limited by cavity cycle time, scaling less favorably with the number of communication qubits.

(3) Free-space: In this architecture, atoms emit photons without the use of cavities, relying on collective emission or frequency encoding schemes [48]. While simpler to scale in hardware, the success probability per attempt is lower, leading to smaller achievable Bell rates for a given communication memory size.

We model the Bell-pair generation rate $r_{\text{bell}}(N)$ as a function of the number of communication qubits N using the analytical model reported in Ref. [18]:

$$r_{\text{bell}}(N) = \frac{P_{\text{aa}}}{t_{\text{base}} + \frac{16}{N} + \frac{100P_{\text{aa}}}{N}} \times 10^6, \tag{E1}$$

where P_{aa} is the per-attempt success probability and t_{base} is the base attempt time. We invert this relationship to determine the required communication memory N for achieving a desired Bell-pair rate $r_{\text{bell}} = r_{\text{bell}}$. The total memory required for a given operating point is then

$$\text{total memory} = \text{logic qubits} + N.$$

Figure 8 shows the *resource-aware performance landscape* for distributed quantum computing with neutral atoms across these three architectures. The x axis denotes the normalized Bell-pair generation rate $r_{\text{bell}}/r_{\text{physical}}$, while the y axis represents the total physical memory (computation + communication).

The color scale indicates the distributed-to-logical gate rate ratio $r_{\text{distributed}}/r_{\text{logical}}$, with contours highlighting the optimal QEC regime (distillation, lattice surgery, or transversal gates) for each operating point.

We assume a total available qubit budget of 20 000 physical atoms, with a minimum of 2500 allocated to logical computation. Feasibility regions for each interconnect architecture are derived by mapping achievable r_{bell} values for different N, based on the following parameters:

(1) 30 Microcavities: $P_{\text{aa}} = 0.24$, $t_{\text{base}} = 0.003\,\mu$s.
(2) Single cavity: $P_{\text{aa}} = 0.10$, $t_{\text{base}} = 0.1\,\mu$s.
(3) Free-space: $P_{\text{aa}} = 0.0035$, $t_{\text{base}} = 0.0\,\mu$s.

This approach reveals how memory allocation for communication qubits directly affects the optimal QEC strategy and the achievable distributed logical gate rate. The resulting performance map, shown in Fig. 8, thus provides a comprehensive comparison of these architectures under realistic hardware constraints.

[1] L. Egan, D. M. Debroy, C. Noel, A. Risinger, D. Zhu, D. Biswas, M. Newman, M. Li, K. R. Brown, M. Cetina, and C. Monroe, Fault-tolerant control of an error-corrected qubit, Nature (London) **598**, 281 (2021).
[2] D. Bluvstein *et al.*, Logical quantum processor based on reconfigurable atom arrays, Nature (London) **626**, 58 (2024).
[3] P. S. Rodriguez *et al.*, Experimental demonstration of logical magic state distillation, Nature **645**, 620 (2025).
[4] B. W. Reichardt *et al.*, Logical computation demonstrated with a neutral atom quantum processor, arXiv:2411.11822.
[5] C. Ryan-Anderson *et al.*, High-fidelity teleportation of a logical qubit using transversal gates and lattice surgery, Science **385**, 1327 (2024).
[6] J.-S. Chen, E. Nielsen, M. Ebert, V. Inlek, K. Wright, V. Chaplin, A. Maksymov, E. Páez, A. Poudel, P. Maunz, and

J. Gamble, Benchmarking a trapped-ion quantum computer with 30 qubits, Quantum **8**, 1516 (2024).

[7] D. Bluvstein, A. A. Geim, S. H Li, S. J. Evered, J. Ataides, G. Baranes, A. Gu, T. Manovitz, M. Xu, M. Kalinowski, *et al.*, Architectural mechanisms of a universal fault-tolerant quantum computer, Nature **649**, 39 (2026).

[8] N. Lacroix *et al.*, Scaling and logic in the color code on a superconducting quantum processor, Nature **645**, 614 (2025).

[9] R. S. Gupta, N. Sundaresan, T. Alexander, C. J. Wood, S. T. Merkel, M. B. Healy, M. Hillenbrand, T. J.-O'Connor, J. R. Wootton, T. J. Yoder, A. W. Cross, M. Takita, and B. J. Brown, Encoding a magic state with beyond break-even fidelity, Nature (London) **625**, 259 (2024).

[10] H. Putterman, K. Noh, C. T. Hann, G. S. MacCabe, S. Aghaeimeibodi, R. N. Patel, M. Lee, W. M. Jones, H. Moradinejad, R. Rodriguez, N. Mahuli, J. Rose, J. C. Owens, H. Levine, E. Rosenfeld, P. Reinhold, L. Moncelsi, J. A. Alcid, N. Alidoust, and P. Arrangoiz-Arriola, Hardware-efficient quantum error correction via concatenated bosonic qubits, Nature (London) **638**, 927 (2025).

[11] Y. Alexeev, M. Amsler, M. A. Barroca, S. Bassini, T. Battelle, D. Camps, D. Casanova, Y. J. Choi, F. T. Chong, C. Chung, C. Codella, A. D. Córcoles, J. Cruise, A. D. Meglio, I. Duran, T. Eckl, S. Economou, S. Eidenbenz, B. Elmegreen, and C. Fare, Quantum-centric supercomputing for materials science: A perspective on challenges and future directions, Future Gener. Comput. Syst. **160**, 666 (2024).

[12] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, Quantum computational chemistry, Rev. Mod. Phys. **92**, 015003 (2020).

[13] C. Gidney and M. Ekerå, How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, Quantum **5**, 433 (2021).

[14] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, Even more efficient quantum computations of chemistry through tensor hypercontraction, PRX Quantum **2**, 030305 (2021).

[15] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoefler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vaschillo, Assessing requirements to scale to practical quantum advantage, arXiv:2211.07629.

[16] A. M. Dalzell, S. McArdle, M. Berta, P. Bienias, C.-F. Chen, A. Gilyén, C. T. Hann, M. J. Kastoryano, E. T. Khabiboulline, A. Kubica, G. Salton, S. Wang, and F. G. S. L. Brandão, *Quantum Algorithms: A Survey of Applications and End-to-End Complexities* (Cambridge University Press, 2025).

[17] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects, Phys. Rev. A **89**, 022317 (2014).

[18] J. Sinclair, J. Ramette, B. Grinkemeyer, D. Bluvstein, M. D. Lukin, and V. Vuletić, Fault-tolerant optical interconnects for neutral-atom arrays, Phys. Rev. Res. **7**, 013313 (2025).

[19] J. Stack, M. Wang, and F. Mueller, Assessing teleportation of logical qubits in a distributed quantum architecture under error correction, arXiv:2504.05611.

[20] J. Ramette, J. Sinclair, N. P. Breuckmann, and V. Vuletić, Fault-tolerant connection of error-corrected qubits with noisy links, npj Quantum Inf. **10**, 58 (2024).

[21] C. Pattison, G. Baranes, J. P. B. Ataides, M. D. Lukin, and H. Zhou, Constant-rate entanglement distillation for fast quantum interconnects, *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, ISCA'25 (Association for Computing Machinery, New York, NY, 2025), pp. 257–270.

[22] Y. Li, A magic state's fidelity can be superior to the operations that created it, New J. Phys. **17**, 023037 (2015).

[23] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[24] S. Majidy, C. Wilson, and R. Laflamme, *Building Quantum Computers: A Practical Introduction* (Cambridge University Press, 2024).

[25] R. Acharya *et al.*, Quantum error correction below the surface code threshold, Nature (London) **638**, 920 (2025).

[26] H. P. Bartling, J. Yun, K. N. Schymik, M. van Riggelen, L. A. Enthoven, H. B. van Ommen, M. Babaie, F. Sebastiano, M. Markham, D. J. Twitchen, and T. H. Taminiau, Universal high-fidelity quantum gates for spin-qubits in diamond, Phys. Rev. Appl. **23**, 034052 (2025).

[27] C. M. Knaut, A. Suleymanzade, Y.-C. Wei, D. R. Assumpcao, P.-J. Stas, Y. Q. Huan, B. Machielse, E. N. Knall, M. Sutula, G. Baranes, N. Sinclair, C. De-Eknamkul, D. S. Levonian, M. K. Bhaskar, H. Park, M. Lončar, and M. D. Lukin, Entanglement of nanophotonic quantum memory nodes in a telecom network, Nature (London) **629**, 573 (2024).

[28] H. Zhou, C. Zhao, M. Cain, D. Bluvstein, C. Duckering, H.-Y. Hu, S.-T. Wang, A. Kubica, and M. D. Lukin, Algorithmic fault tolerance for fast quantum computing, Nature **646**, 303 (2025).

[29] M. Cain, C. Zhao, H. Zhou, N. Meister, J. P. B. Ataides, A. Jaffe, D. Bluvstein, and M. D. Lukin, Correlated decoding of logical algorithms with transversal gates, Phys. Rev. Lett. **133**, 240602 (2024).

[30] H. J. Manetsch, G. Nomura, E. Bataille, K. H. Leung, X. Lv, and M. Endres, A tweezer array with 6100 highly coherent atomic qubits, Nature **647**, 60 (2025).

[31] N.-C. Chiu, E. C. Trapp, J. Guo, M. H. Abobeih, L. M. Stewart, S. Hollerith, P. Stroganov, M. Kalinowski, A. A. Geim, S. J. Evered, *et al.*, Continuous operation of a coherent 3,000-qubit system, Nature **646**, 1075 (2025).

[32] P.-J. Stas, Y. Q. Huan, B. Machielse, E. N. Knall, A. Suleymanzade, B. Pingault, M. Sutula, S. W. Ding, C. M. Knaut, D. R. Assumpcao, *et al.*, Robust multi-qubit quantum network node with integrated error detection, Science **378**, 557 (2022).

[33] Y.-C. Wei, P.-J. Stas, A. Suleymanzade, G. Baranes, F. Machado, Y. Q. Huan, C. M. Knaut, S. W. Ding, M. Merz, E. N. Knall, U. Yazlar, M. Sirotin, I. W. Wang, B. Machielse, S. F. Yelin, J. Borregaard, H. Park, M. Lončar, and M. D. Lukin, Universal distributed blind quantum computing with solid-state qubits, Science **388**, 509 (2025).

[34] G. Baranes, I. W. Wang, F. Machado, A. Suleymanzade, P.-J. Stas, Y.-C. Wei, S. F. Yelin, J. Borregaard, and M. D. Lukin, Designing fault-tolerant blind quantum computation, arXiv:2505.21621.

[35] R. Acharya *et al.*, Suppressing quantum errors by scaling a surface code logical qubit, Nature (London) **614**, 676 (2023).

[36] S. Storz *et al.*, Loophole-free Bell inequality violation with superconducting circuits, Nature (London) **617**, 265 (2023).

[37] G. Baranes, M. Cain, J. Ataides, D. Bluvstein, J. Sinclair, V. Vuletic, H. Zhou, and M. D. Lukin, Leveraging atom loss errors in fault tolerant quantum algorithms, Phys. Rev. X **16**, 011002 (2026).

[38] J. P. B. Ataides, H. Zhou, Q. Xu, G. Baranes, B. Li, M. D. Lukin, and L. Jiang, Constant-overhead fault-tolerant Bell-pair distillation using high-rate codes Phys. Rev. Lett. **135**, 130804 (2025).

[39] T. J. Yoder, E. Schoute, P. Rall, E. Pritchett, J. M. Gambetta, A. W. Cross, M. Carroll, and M. E. Beverland, Tour de gross: A modular quantum computer based on bivariate bicycle codes, arXiv:2506.03094.

[40] F. K. Marqversen, quantum_computations, GitHub (2025), https://github.com/frederik-kofoed-marqversen/quantum_computations.

[41] M. Sirotin, Growing_simulation, GitHub (2025), https://github.com/MaximSirotin/BellPairsCode.

[42] L. Lao and B. Criger, Magic state injection on the rotated surface code, in *Proceedings of the 19th ACM International Conference on Computing Frontiers* (ACM, Turin, Italy, 2022), pp. 113–120.

[43] C. Gidney, Stim: A fast stabilizer circuit simulator, Quantum **5**, 497 (2021).

[44] Y. Q. Huan, A. Suleymanzade, P.-J. Stas, Y.-C. Wei, E. N. Knall, F. A. Arias, E. Kniazev, B. Machielse, U. Yazlar, G. Baranes, *et al.*, *Towards Quantum Repeater Nodes with Weakly-Coupled Nuclear Spins in Diamond*, *Quantum 2.0* (Optica Publishing Group, 2025), pp. QW4A–3.

[45] S. Krastanov, H. Raniwala, J. Holzgrafe, K. Jacobs, M. Lončar, M. J. Reagor, and D. R. Englund, Optically heralded entanglement of superconducting systems in quantum networks, Phys. Rev. Lett. **127**, 040503 (2021).

[46] M. Cain, D. Bluvstein, C. Zhao, S. Gu, N. Maskara, M. Kalinowski, A. A. Geim, A. Kubica, M. D. Lukin, and H. Zhou, Fast correlated decoding of transversal logical algorithms, arXiv:2505.13587.

[47] Y. P. Zhong, H.-S. Chang, K. J. Satzinger, M.-H. Chou, A. Bienfait, C. R. Conner, É. Dumur, J. Grebel, G. A. Peairs, R. G. Povey, *et al.*, Violating Bell's inequality with remotely connected superconducting qubits, Nat. Phys. **15**, 741 (2019).

[48] J. P. Covey, H. Weinfurter, and H. Bernien, Quantum networks with neutral atom processing nodes, npj Quantum Inf. **9**, 90 (2023).

[49] S. J. Evered, D. Bluvstein, M. Kalinowski, S. Ebadi, T. Manovitz, H. Zhou, S. H. Li, A. A. Geim, T. T. Wang, N. Maskara, H. Levine, G. Semeghini, M. Greiner, V. Vuletić, and M. D. Lukin, High-fidelity parallel entangling gates on a neutral-atom quantum computer, Nature (London) **622**, 268 (2023).

[50] S. Ritter, C. Nölleke, C. Hahn, A. Reiserer, A. Neuzner, M. Uphoff, M. Mücke, E. Figueroa, J. Bochmann, and G. Rempe, An elementary quantum network of single atoms in optical cavities, Nature (London) **484**, 195 (2012).

[51] Y. Li and J. D. Thompson, High-rate and high-fidelity modular interconnects between neutral atom quantum processors, PRX Quantum **5**, 020363 (2024).