Towards Early Fault Tolerance on a $2 \times N$ Array of Qubits Equipped with Shuttling

Adam Siegel,[1,2,*] Armands Strikis,[2,3] and Michael Fogarty[2]

[1] *Department of Materials, University of Oxford, Parks Road, Oxford OX1 3PH, United Kingdom*
[2] *Quantum Motion, 9 Sterling Way, London N7 9HJ, United Kingdom*
[3] *Department of Physics and Astronomy, University College London, Gower St, London WC1E 6BT, United Kingdom*

(Received 11 April 2024; revised 22 August 2024; accepted 16 October 2024; published 25 November 2024)

It is well understood that a two-dimensional grid of locally interacting qubits is a promising platform for achieving fault-tolerant quantum computing. However in the near future, it may prove less challenging to develop lower-dimensional structures. In this paper, we show that such constrained architectures can also support fault tolerance; specifically we explore a $2 \times N$ array of qubits where the interactions between non-neighboring qubits are enabled by shuttling the logical information along the rows of the array. Despite the apparent constraints of this setup, we demonstrate that error correction is possible and identify the classes of codes that are naturally suited to this platform. Focusing on silicon spin qubits as a practical example of qubits believed to meet our requirements, we provide a protocol for achieving full universal quantum computation with the surface code, while also addressing the additional constraints that are specific to a silicon spin-qubit device. Through numerical simulations, we evaluate the performance of this architecture using a realistic noise model, demonstrating that both surface code and more complex quantum low-density parity-check codes efficiently suppress gate and shuttling noise to a level that allows for the execution of quantum algorithms within the classically intractable regime. This work thus brings us one step closer to the execution of quantum algorithms that outperform classical machines.

DOI: 10.1103/PRXQuantum.5.040328

I. INTRODUCTION

Quantum computing holds the promise of solving tasks that lie beyond the capabilities of classical computers. Nonetheless, their full potential can only be realized by executing deep quantum algorithms that require extremely low logical error rates, below 10^{-10} [1,2], thus far from what can directly be achieved on physical devices [3]. Quantum error correction promises to bridge this gap by increasing the qubit overhead to build quantum error-correcting codes. However necessary, these architectures represent formidable experimental challenges as they require the entanglement of a very large number of qubits and the repeated measurement of a considerable number of operators called stabilizers [1]. To facilitate their physical implementation, practical constraints are often integrated in the code design, such as locality of the interactions and planar structure of the code. The main example of error-correcting code respecting these constraints is the surface code [4,5], and its high threshold—that is the maximum error rate the code can tolerate to exponentially reduce errors—makes it one of the best candidates for achieving fault tolerance. However, more general quantum low-density parity-check (qLDPC) codes have been constructed that demonstrate better performance in finite-size simulations [6–11]. Although these codes seem more challenging to realize experimentally due to their long-range interactions, they have the potential to considerably lower the qubit overhead due to their higher rate, i.e., the number of encoded logical qubits per physical qubit, and better distance scaling. Consequently, further research has been undertaken to demonstrate the experimental viability of such complex codes, and suggested solutions for the implementation of their long-range connections, either by swapping [12] or displacing the qubits [11,13,14].

It may seem natural to assume the use of fully two-dimensional (2D) architectures to embed such codes, possibly with augmentations to enable some longer-range interactions (as in, e.g., Ref. [7]). However, such complex devices might not be available for some time; their first fault-tolerant iteration may be more limited. For example, the size of an array realized on a quantum chip might be constrained in one direction. This motivated research

[*]Contact author: adam.siegel@materials.ox.ac.uk

toward the feasibility of one-dimensional (1D) or quasi-1D error correction [15,16]. Implementing 2D codes in this kind of device can theoretically be solved by extending the length of the interactions, thereby increasing the nonlocality of the operations. Albeit challenging, this can however be tackled in the same way as long-range interactions in the aforementioned qLDPC codes, i.e., by swapping or displacing the qubits.

In this paper, we study the feasibility of quantum error correction in the most constrained experimental setup beyond 1D: a $2 \times N$ array of qubits. Long-range interactions are implemented by assuming that the qubits can be collectively shuttled along one of the two lines of the array. Several qubit platforms have been shown to be suitable to implement such an operation with high fidelity, including atom arrays [13] or ion traps [17], but spins qubits [18] may be the most promising one for also implementing the proposed $2 \times N$ array. First, we establish a precise framework to determine the classes of codes that can be efficiently implemented in our device. This study is then applied to two specific examples: the surface code and higher-rate qLDPC codes. Regarding the surface code, we demonstrate that universal quantum computation is possible in a practical setup based on silicon spin qubits, despite the additional experimental constraints that the geometry entails. With the help of numerical simulations accounting for additional shuttling errors, we estimate a resource cost that is moderate enough to run meaningful quantum algorithms in the classically intractable regime. As for higher-rate qLDPC codes, we evaluate their performance under the noise processes that are anticipated for relevant devices, and demonstrate that despite their complexity they do give an advantage over the surface code in a noise domain that is practically achievable.

For simplicity, our analysis assumes a strictly $2 \times N$ array. In practice, given the long device length that would be required for meaningful postclassical tasks, it is reasonable to assume that a real device could incorporate a plurality of junctions—thus the overall geometry would be a lattice formed of long $2 \times N$ sections. The processes we analyze in this paper would then occur along each long section, while the overall algorithm would benefit from the additional routing advantages of the lattice. We return to this possibility in the discussion.

II. FRAMEWORK FOR FINDING CODES WITHIN DEVICE RESTRICTIONS

In this section, we introduce the formalism for finding QEC codes that naturally suit the $2 \times N$ architecture. Before describing the architecture in detail, let us note the perspective we take to judge a code's compatibility.

To start with, we require the architectural constraints to permit implementing a quantum circuit that performs repeated QEC cycles of a suitable code. Specifically, the qubit connectivity graph of the architecture should allow one to repeatedly extract syndrome for error correction. In such a graph, the qubits represent vertices and we take it that entangling two-qubit operations can be implemented between qubits that are connected by an edge. On its own this is not a strong requirement. Any QEC code can be implemented on any connectivity graph as long as one of its connected components hosts the required number of qubits. This could always be achieved in principle, for example by applying a sufficient number of SWAP gates.

Therefore, as an addition to the first criterion, we require syndrome extraction to be efficient at some desired scale. That is, we would like to finish a full QEC cycle in some relatively small number of time steps. While currently this criterion is not well defined in the mathematical sense, it is easier to further specify it after introducing the architecture.

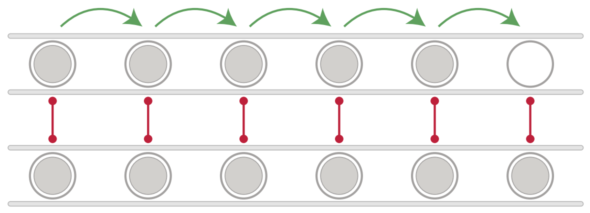
FIG. 1. Representation of the $2 \times N$ architecture. The device is characterized by two parallel rails of evenly spaced qubits (gray circles with gray disks inside). Adjacent qubits from different rails are allowed to interact via two-qubit gate operations (red vertical lines with disks on their ends). Finally, the qubits of the first row are allowed to shuttle along their rail (green arrows). Some empty locations are kept at the end of the device to leave space for the qubits to shuttle (empty gray circle).

A. Architecture

Let us describe the architecture while keeping the above device criteria in mind. The $2 \times N$ architecture is a quasi-1D array where all qubits (data and ancilla) are placed on one of two parallel rails. Separately these rails are treated as 1D strings of evenly spaced qubits, however, an important feature is that the adjacent qubits from separate rails are allowed to interact "across the ridge" to perform two-qubit operations such as controlled-Z (CZ) and controlled-X (CNOT), see Fig. 1. Clearly such a structure might support other interactions, for example qubits in one or both rails may be able to interact with nearest neighbour qubits within the same rail. However, controlling such interactions might imply additional engineering cost, and in fact such capabilities are *not* required in the approaches we describe.

So far the qubit connectivity graph for our $2 \times N$ architecture is very limited. To make this more suitable for a wide range of quantum circuits and codes, we introduce the second feature of the architecture—qubit shuttling.

Shuttling will play a crucial role in generating different kinds of connectivity graphs that can be used for quantum error correction (QEC). Since we are dealing with 1D rails of qubits, we can constrain the shuttling to a "conveyor-belt" model where the whole rail of qubits simultaneously moves along the rail in one direction or the other. After the move, the qubits of one rail may again interact with adjacent qubits in the other rail. Since only the respective qubit positioning of one rail to the other is important, we can always fix one of the rails to stay static.

In this way, the rail of qubits can be shuttled multiple times during a single QEC cycle to generate the desired connectivity graph for syndrome extraction. Here, we consider the canonical syndrome extraction method where ancilla qubits are coupled to the data qubits of the code. If there are no other interactions, the connectivity graph is bipartite—its exact structure depends on the specific choice of the code. However, by locating all ancilla qubits on one rail and all data qubits on the other, any bipartite connectivity graph (corresponding to an arbitrary code) can be generated using a sufficient number of (possibly long) shuttles. As noted above, this generally would not be practical, and therefore, for each QEC cycle we either set the number of shuttles (of any length) to be some small constant number that does not scale with the code size, or we restrict the overall shuttling distance (without constraining the number of shuttles). Due to this, we do not expect to produce asymptotically scalable QEC protocols. The same conclusions are also supported by numerical results presented later in the paper.

Therefore, there seem to be only a limited number of classes of quantum stabilizer codes satisfying these constraints. To describe them, let us formally derive the qubit connectivity graphs that can be obtained with our architecture.

B. Formalism

Consider the scenario where all ancilla qubits are placed on the static rail and data qubits on the other one—the mobile rail. As we explain later, this layout is preferential for the physical platform we showcase. While we do not consider mixed type rails on which both data and ancilla qubits are placed in some sequence, note that they can be more powerful if additional two-qubit operations are allowed between the neighboring qubits on the same rail. Furthermore, assume that both rails have a number N of qubits and are initially aligned. Then, after indexing each data and ancilla qubit along the rail, we see that the ith ancilla qubit may interact with the ith data qubit. Therefore, the biadjacency matrix of the qubit connectivity graph is a diagonal $N \times N$ matrix $H = \mathrm{diag}(1, 1, \ldots, 1)$. Here, the rows correspond to ancilla qubits and columns to the data qubits. If the top rail is shuttled m qubit spaces to either direction, so is the diagonal of H shifted by m spaces to either left or right direction of the principal diagonal. For example, shuttling the data qubit rail one position to the left yields us the following transformation:

$$H_0 = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix} \xrightarrow{\text{shuttle}}$$

$$H_1 = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

Finally, if we consider interacting qubits before and after the shuttle, we obtain a biadjacency matrix $H = H_0 + H_1$. This does imply a particular order of applying two-qubit operations, which we ignore until we consider specific QEC protocols. Note that such shuttling on a $2 \times N$ architecture does not generate periodicity of diagonals (the bottom-left element of H_1 is not a 1). However, it can be created by supplementing the protocol with another shuttle. The farther away two diagonals are, the longer the shuttle operation needed to go between them. Furthermore, not all elements have to be 1's along these diagonals, as some interactions may not be implementable due to limited control, or may not be needed. For example, the biadjacency matrices

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad H' = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

have two and five nontrivial (nonzero) diagonals, respectively.

So far we have described how to generate a biadjacency matrix describing a bipartite qubit connectivity graph. Let us now link this back to finding compatible QEC codes as follows. Consider treating ancilla qubits as parity checks that interact with their respective data qubits during the shuttling phase and afterwards are measured out in some basis. Then the generated biadjacency matrix H describes a potential parity-check matrix of some code $\mathcal{C}$ whose QEC cycle can be implemented in a number of steps proportional to the number of shuttles. The same procedure of shuttling, interacting, and measuring can be repeated over again, and therefore multiple QEC cycles can be continuously performed. There are still some degrees of freedom left. For example, we have not specified what kind of Pauli string each parity check (row of H) is. Therefore, any stabilizer code whose parity-check matrix can be written with nonzero elements matching the 1's of the generated H would count as a suitable code—as long as the number of shuttles or their overall length is small. In fact, that is the approach we take. We find codes with parity-check matrices that have few nontrivial diagonals (low number of shuttles) or all of them are close to the main diagonal (short shuttling distance). We then map them to our $2 \times N$ architecture. One can also go the other way around and try to design a stabilizer code by first generating an arbitrary H and assigning rows to specific parity checks. However, knowing which Pauli strings to choose to create a good code with commuting stabilizers seems to be a difficult task.

Let us again note that sequencing the entangling operations between data and ancilla qubits can be crucial when it comes to extracting syndrome. By only considering CSS codes, we can always perform all Z type stabilizers while we shuttle the data qubits in one direction and all X-type stabilizers while we shuttle them back to their initial position. While this increases the total number of shuttles and the qubit idling time, the overall shuttling distance is kept the same as when both stabilizer measurements are interleaved. Such sequencing is used later when we consider more general qLDPC codes. Unfortunately, this method does not guarantee that hook errors reducing the code's effective distance will not arise; these need to be considered separately.

Our proposed recipe is thus to find parity-check matrices with a few or closely located nontrivial diagonals, which we demonstrated are suitable for the $2 \times N$ architecture. Let us now provide a couple of examples.

C. Code examples

1. Rotated surface code

The first and potentially the most promising example is the rotated surface code. It is usually depicted in a graphical way as in Fig. 2(a). Since for this code there are weight-four stabilizer generators, its parity-check matrix has at least four nontrivial diagonals. In fact, by assigning some parity-check qubits to have no support, we can construct a square parity-check matrix of a rotated surface code that has exactly four nontrivial diagonals. See Fig. 2 for a three-way mapping of a distance-3 rotated surface code. As explained before, having four diagonals is nonetheless not sufficient to ensure that a stabilizer cycle can be implemented in four steps—the order of the entangling gates is highly significant. We however verify in Appendix A that measuring the diagonals of Fig. 2(b) in the order 1-4-2-3 (where the diagonals are indexed from left to right) guarantees a minimum number of shuttles, a minimum shuttling distance, and no distance-reducing hook errors.

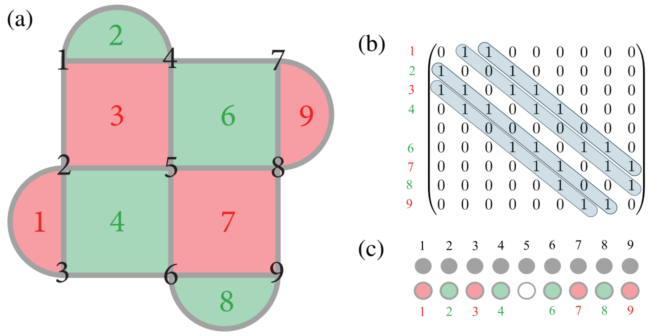

FIG. 2. Three-way mapping for the rotated surface code. (a) The canonical 2D layout, (b) its parity-check matrix with an additional trivial stabilizer and (c) a schematic for the layout on a $2 \times N$ architecture. Here, red and green colors indicate X and Z stabilizers, respectively.

2. Hypergraph product codes

Another interesting class of codes are the hypergraph product (HGP) codes that are composed with a repetition code as one of the seed codes. The other classical code should still be an LDPC code, but can be chosen arbitrarily. For such codes, the number of shuttles required to obtain the full connectivity graph grows with the code size, however, the full shuttling distance scales only as a square-root of the number of qubits. To see this, take A to be an $(n-1) \times n$ parity-check matrix of a classical repetition code

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & \dots \\ 0 & 1 & 1 & 0 & \dots \\ 0 & 0 & 1 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

and B to be a random sparse $l \times k$ matrix representing some classical LDPC code. Then the HGP code of A and B has parity-check matrices

$$H_x = \left(A \otimes \mathbb{I}_k, \mathbb{I}_{n-1} \otimes B^T\right),$$
$$H_z = \left(\mathbb{I}_n \otimes B, A^T \otimes \mathbb{I}_l\right),$$

where $\mathbb{I}_m$ is an identity matrix of size $m \times m$. Since one has the freedom to index the data qubits in any order (i.e., one can switch columns around) it is easy to rearrange H_x in the block form as

$$H_x = \begin{pmatrix} \mathbb{I}_k & B^T & \mathbb{I}_k & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & \mathbb{I}_k & B^T & \mathbb{I}_k & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \mathbb{I}_k & B^T & \mathbb{I}_k & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Similarly, one can rearrange H_z to have the nontrivial diagonals closer to the principal diagonal,

$$H_z = \begin{pmatrix} B & \mathbb{I}_l & 0 & 0 & 0 & 0 & \dots \\ 0 & \mathbb{I}_l & B & \mathbb{I}_l & 0 & 0 & \dots \\ 0 & 0 & 0 & \mathbb{I}_l & B & \mathbb{I}_l & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Finally, the total parity-check matrix H is rearranged by placing block rows of H_z and H_x in an alternating fashion (as for the rotated surface code presented above) resulting in

$$H = \begin{pmatrix} B & \mathbb{I}_l & 0 & 0 & 0 & 0 & 0 & \dots \\ \mathbb{I}_k & B^T & \mathbb{I}_k & 0 & 0 & 0 & 0 & \dots \\ 0 & \mathbb{I}_l & B & \mathbb{I}_l & 0 & 0 & 0 & \dots \\ 0 & 0 & \mathbb{I}_k & B^T & \mathbb{I}_k & 0 & 0 & \dots \\ 0 & 0 & 0 & \mathbb{I}_l & B & \mathbb{I}_l & 0 & \dots \\ 0 & 0 & 0 & 0 & \mathbb{I}_k & B^T & \mathbb{I}_k & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

whose nontrivial diagonals are at most $O(\sqrt{N})$ data qubits apart from each other. This represents the shuttling distance that is necessary to generate the full connectivity graph. Note that, since we take B to correspond to an arbitrary LDPC code, the generation of most nontrivial diagonals involves only $O(\sqrt{N})$ qubits. The rest of the qubits are idle, and hence one needs to account for the potential negative effects of idling on the code's performance in practice (see Sec. IV A).

3. Generalised bicycle codes

Finally, consider two-block LDPC codes whose parity-check matrices take the form of

$$H_x = \begin{pmatrix} C, & D \end{pmatrix}, \tag{1}$$

$$H_z = \begin{pmatrix} D^T, & C^T \end{pmatrix} \tag{2}$$

where both C, D are sparse binary matrices that commute. A way to ensure this commutation is to take C and D as any two $l \times l$ sparse circulant matrices that have the form

$$C = \begin{pmatrix} a_0 & a_{l-1} & a_{l-2} & \dots & a_1 \\ a_1 & a_0 & a_{l-1} & \dots & a_2 \\ a_2 & a_1 & a_0 & \dots & a_3 \\ a_3 & a_2 & a_1 & \dots & a_4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix},$$

where $a_i \in \mathbb{F}(2)$ such that only a constant number of a_i are nontrivial. These matrices have an intrinsically diagonal structure, therefore, the number of shuttles required to generate the qubit connectivity graph for the corresponding parity-check matrices [Eqs. (1) and (2)] is constant. Codes constructed with such sparse circulant matrices have been shown to perform well within the code capacity-based simulations [8], therefore, it is interesting to consider them in our $2 \times N$ architecture where shuttling noise has to be accounted for. While the number of shuttles is constant, note that, in general, the shuttling distance for these codes scale with the size of the code block.

In the following, we present realistic noise simulations of the mentioned codes under the $2 \times N$ architecture. We focus on the rotated surface code for universal fault-tolerant quantum computation and later discuss how codes from other classes can be used for efficient quantum memory.

III. UNIVERSAL QUANTUM COMPUTATION ON A SPIN-QUBIT SURFACE CODE

The previous section discussed how to embed an individual code block on a $2 \times N$ device by taking advantage of shuttling. This is only the first step toward the implementation of a fully functional quantum computer, as we still need to discuss how operations between logical qubits can be implemented despite the strong architectural constraints. In this section, we will demonstrate that universal quantum computation is indeed possible when embedding surface-code-like error-correcting codes in a $2 \times N$ device equipped with shuttling. In order to ensure that our proposition is practical from the experimental point of view, we will focus on a silicon spin-qubit device. Shuttling has been demonstrated to be implementable at high fidelity on such a platform [19,20]. This motivates the introduction of additional constraints discussed below.

A. Silicon spin qubits

Electron spins in silicon quantum dots (QDs) are a promising physical platform to perform quantum computing: single- and two-qubit gates with fidelity well above 99% [21–23] or even beyond 99.9% [24] have been demonstrated, simple instances of quantum error correction have been shown to be feasible [25] or have already been implemented [26], and the technology has been scaled to processors with up to six qubits [27]. Furthermore, the compatibility with advanced manufacturing techniques [28,29] and cryogenic classical electronics [30, 31] make them ideal candidates for large-scale integration [32].

While silicon architectures are expected to eventually provide dense two-dimensional grids of qubits [33,34], early silicon quantum processor designs are particularly interesting since they could meet the requirements for the implementation of the protocol described in this paper. These are (i) a bilinear qubit topology, (ii) information shuttling. Both these criteria have been met experimentally: devices with $2 \times N$ arrays of QDs have been demonstrated [35]; and information transfer has been achieved using spin-shuttling techniques in the form of bucket-brigade [19] or conveyor-belt approaches [20]. The first method makes use of tunneling to shuttle electrons, by successively lowering the potential of subsequent quantum dots along their way to generate their movement. In contrast, the second method aims at always keeping the wave function at a minimum of a smoothly moving sine wave, without tunneling out of it (see Fig. 2 of [18]). In our paper, as we aim to shuttle all data qubits collectively, the conveyor-belt approach is advantageous: indeed electrons can just be placed at the minima of the moving sine-wave potential. Only four signals are thus required for the shuttling of all the data qubits [18]. The drawback is that each electron must occupy the space of four clavier gates rather than one.

There is of course a natural set of gates that one can directly implement with a class of silicon spin-qubit device (and here we assume single-spin representations of qubits). More precisely, local phase rotations [36] and two-qubit operations such as C-Phase gates [37] have been shown. However, general single-qubit rotations can be more challenging to localize, and may be more naturally realized on a wide scale, via the interaction of the spins with a global oscillatory field [38].

For the present analysis, we will require only a modest refinement of unconditionally global Hadamard gates. It will suffice to apply the operation to the entirety of one of the two linear arrays of the processor. This partial global, or "semiglobal" operation could be implemented in silicon devices using frequency engineering, for example, by placing a micromagnet next to the array [39] or by using magnetic materials in the gate stack on one side of the array [40]. Such approaches could create a frequency difference between the two linear arrays that can be selectively addressed with known electron-spin-resonance techniques [41]. A schematic implementation of such frequency difference between the two lines of the array is pictured in Fig. 3, via the use of a micromagnet. Operating at a global magnetic field $B_0 = 1.4$ T, one can expect the g-factor dispersion in the device to be around $\Delta f = 60$ MHz [42] In order to sufficiently separate the frequencies of both lines of the array, one could therefore set the micromagnet-induced magnetic field difference to $\Delta B = 5h\Delta f / g\mu_B = 10$ mT. Here h is Planck's constant, $g = 2$ is the electron's Landé factor and μ_B is Bohr's magneton. Given a spacing of 100 nm between the lines, this would mean a gradient of 0.1 mT/nm, which is comfortably below demonstrated gradients, around 0.8 mT/nm [43].

In our setup, the first and second one-dimensional arrays contain the data and ancilla qubits, respectively, and the processor is subject to a magnetic field to polarize the spins. Each data qubit is encoded by a single quantum dot, initialized in the $|0\rangle$ state (spin down) at the beginning of the computation, for example, by spin relaxation. In contrast, each ancilla qubit is encoded by two electrons in a singlet or triplet state. This is consistent with established techniques for preparation and measurement: at the start of each stabilizer cycle, the ancilla qubits are initialized in the singlet state by first applying a differential gate potential between QDs to produce an energy detuning that will relax the system in the singlet (02) [or (20)] state, followed by a ramp to the singlet (11) state (adiabatic with respect to the singlet anticrossing and diabatic with respect to the singlet-triplet anticrossing). At the end of the cycle, the stabilizer is measured by projecting the pair of spins to one of the QDs, which, through Pauli spin blockade reveals a different measurement outcome for singlet or triplet states [44,45]. Finally, note that this choice of singlet triplets requires the occupation of two quantum dots per ancilla qubit. Besides, measurement apparatuses cannot be included in the data row as they would block the shuttling, and hence can only exist in the ancilla row. For these reasons, the ancilla row has to be denser than the data row. This additional space between data qubits is not going to waste as it can be used to fit in the four clavier gates per electron required by the conveyor-belt mode of shuttling.

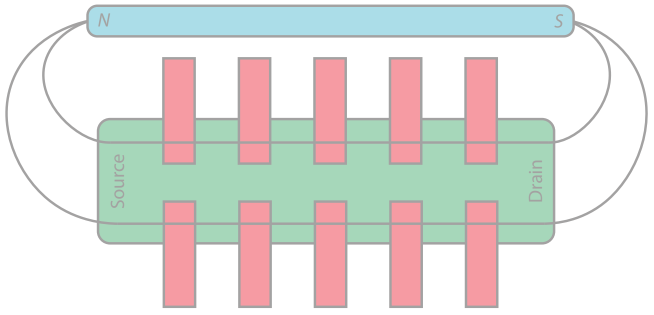

FIG. 3. Schematic representation of a device where the g factors in the first row are all higher than the g factors in the second row. The green rectangle represents the source and drain, while the red and blue ones, respectively, represent the gates confining the electrons and the micromagnet. Two field lines are additionally drawn, showing the difference in magnetic field between both rows of the $2 \times N$ array. This is in turn responsible for a difference in Zeeman splitting, which can be utilized to selectively target either the spins of the first row or those of the second row.

B. Logical gates and qubit layout

One of the major constraints discussed above is the absence of local single-qubit gates (apart from phase gates). As such, if one wants to implement, say, a logical X gate on a given logical qubit, applying transversal X gates on the corresponding surface code would not meet our constraints, as these gates would have to be applied on all data qubits in the device, thereby implementing a global logical X. Local transversal single-logical-qubit gates are thus not permitted in our device (apart from Z gates). This motivates the choice of a protocol that would not make use of such transversal gates, but rather, of lattice surgery only. Indeed, this operation can be made local as our device does feature selective CZ gates. One such approach is described in Refs. [46,47]: using Clifford+T as a universal gate set, these papers show that Clifford gates can be commuted through T gates and incorporated in later measurements, thereby only leaving multiqubit Pauli measurements and multiqubit $\pi/8$ rotations to implement (Fig. 4 in Ref. [47]). These rotations can in turn be performed by consuming a magic state through additional multiqubit measurements. This protocol thus removes the necessity to implement local Clifford gates that are impractical in our device. However, it comes at the cost of complexified measurements, which require a modified lattice surgery protocol. As for the magic state, it can be prepared via state injection and subsequent magic state distillation. The latter only requires the implementation of logical CNOTs, which can equally be performed via lattice surgery. Using this scheme, the remaining operations our device must be able to implement for universal quantum computation are thus (i) stabilizer measurements, (ii) all variants of lattice surgery as described in Ref. [46] and (iii) state injection. An implementation of these three components within our architectural limitations is described in Sec. III C.

Before discussing the circuit-level implementation of these elements, it is first relevant to discuss the layout that we adopt for our logical qubits, as this will have an impact on the ordering of the gates in each circuit (e.g., to avoid distance-reducing hook errors).

In order to embed an $n \times m$ grid of $d \times d$ tiles of data qubits (as in Ref. [47]) in a $2 \times N$ device, one must slice the whole grid, say, in the vertical direction. If we assume for now that each tile represents exactly one surface code, this means two consecutive data qubits within a row of a given patch are now separated by nd data qubits rather than d. The consequence is shuttle operations that are n times longer such that these two qubits can participate in the same stabilizer measurement. As shuttling is prone to errors, one way to minimize such long shuttles while permitting full quantum computation is to set $n = 2$ and consider a $2 \times m$ grid of tiles, where the logical information would only be stored in the first row of the grid (region A), while the second row (region B) would serve as a logical ancilla bus used for long-range interactions between the logical qubits of region A (Fig. 4). Additionally, a magic state factory is included at the right end of the device. In order to enable all types of interactions between the logical qubits of region A, both their logical X and Z operators should be adjacent to region B. This motivates the use of wide qubits as in Fig. 4, on top of an ancilla bus that can be initialized differently depending on the desired operation. The slicing is performed in the vertical direction, meaning that the first row of the $2 \times N$ device will contain an

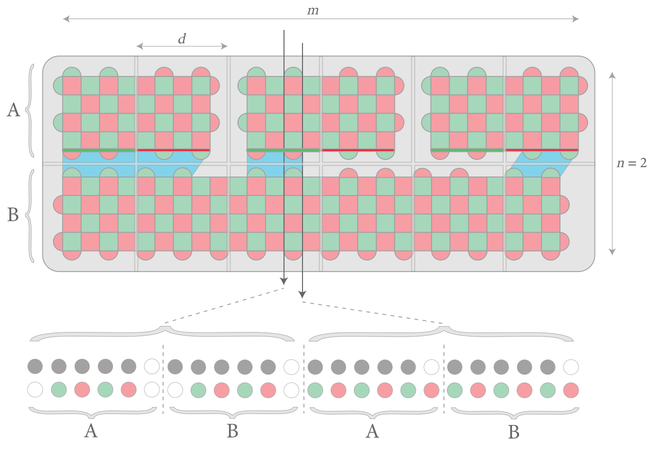

FIG. 4. Qubit layout for universal quantum computation on the $2 \times N$ architecture. We choose to arrange the logical qubits on a grid of $n \times m$ square tiles of size $d \times d$, where d is the code distance [47]. We set $n = 2$ to minimize the shuttling distance. X and Z stabilizers are, respectively, represented with red and green squares or half-disks. The direction of shuttle is indicated by the vertical dark gray arrows. The top half (region A) contains the logical qubits storing the logical information, while the bottom half (region B) is a logical ancilla bus that can be used to perform long-range interactions between the logical qubits. Each logical qubit is a rectangular patch of surface code, whose X and Z logical operators are represented by red and green lines. Both these logical operators are adjacent to region B, so that they can interact with the logical ancilla. In this example, the ancilla is initialized so as to measure the operator $Y_1 \otimes Z_2 \otimes X_3$ with three lattice surgeries, represented with the blue boxes (see Fig. 44 of Ref. [47] for more details). Below, the portion of the layout corresponding to the qubits along the arrows is linearized, showing the alternation of regions A and B in the $2 \times N$ architecture.

alternation of d data qubits from region A and d data qubits from region B. In order to entangle physical data qubits of consecutive columns with a physical ancilla qubit, one must therefore shuttle by roughly $2d$ increments in one direction, and the same distance back (instead of d in the case of a single encoded surface code).

C. Stabilizers' implementation

1. Gate ordering

The next step is to ensure that our three building blocks needed for fault-tolerant quantum computation—stabilizer measurements, modified lattice surgery, and state injection—can indeed be implemented within our constraints. Before giving explicit circuits or procedures implementing these operations, one must first determine the order in which data qubits must be entangled within a given stabilizer measurement. This is an important matter as a suboptimal choice of ordering can lead to hook errors, which effectively reduce the distance of the code. Additionally, a proper sequencing can help reduce the stabilizer circuit depth, number of shuttles, and shuttling distance by

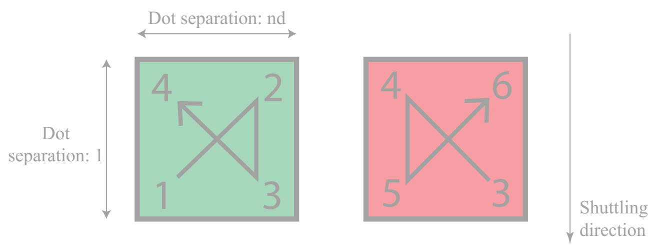

FIG. 5. Gate ordering enabling one to measure X and Z stabilizers in an interleaved (but staggered) fashion. While X stabilizers perform steps 5 and 6, Z stabilizers undergo steps 1 and 2 again, and so on. The distance between data qubits (in terms of the number of quantum dots) is indicated. This sequencing guarantees a minimum shuttling distance, a minimum number of shuttles and no distance-reducing hook errors.

interleaving the gates of the X and Z stabilizers. We verify in Appendix A that the gate ordering depicted in Fig. 5 (i) does not create any hook error; (ii) allows one to interleave the gates of the X and Z stabilizers; (iii) is nearly optimal in terms of the number of shuttles and total shuttling distance.

In the case of modified lattice surgery however, the question of interleaving the dislocations and twists (see next section) with the regular stabilizers is much more complex. Not doing so is always an option, however, that comes at the cost of additional shuttles, and leaves some qubits idle while others are being measured. We leave a study similar to Fig. 16 of Ref. [46] to further research.

2. Stabilizer circuits with singlet-triplet ancilla qubits

Given the silicon spin platform, which we are principally considering, we take it that the preferred embodiment of an ancilla qubit is via two spins, with measurement occurring by differentiating between singlet and triplet states. After the ancilla is initialized in the singlet state, if there are an odd number of errors affecting the data qubits then the ancilla should transform to a triplet state, while an even number of data qubit errors should leave it in the singlet state. A Pauli spin blockade measurement then allows one to extract the value of the stabilizer.

Figure 6 gives a circuit implementation of all types of stabilizers involved in regular error correction as well as lattice surgery, using CZ gates and semiglobal Hadamards only (i.e., affecting all data qubits at the same time). These stabilizers fall into four categories: regular X stabilizers, regular Z stabilizers, dislocations (stabilizer checks involving both X and Z) and twists (stabilizer checks involving X, Z, and Y). The two latter appear in the modified lattice surgery protocols of Ref. [47] when X and Z boundaries are facing (rather than both X or both Z in the conventional lattice surgery picture). One can check that, with this implementation, an error on a data qubit always transforms the singlet state $|S\rangle = (|01\rangle - |10\rangle)/\sqrt{2}$ into the triplet state $|T\rangle = (|01\rangle + |10\rangle)/\sqrt{2}$. This would not be

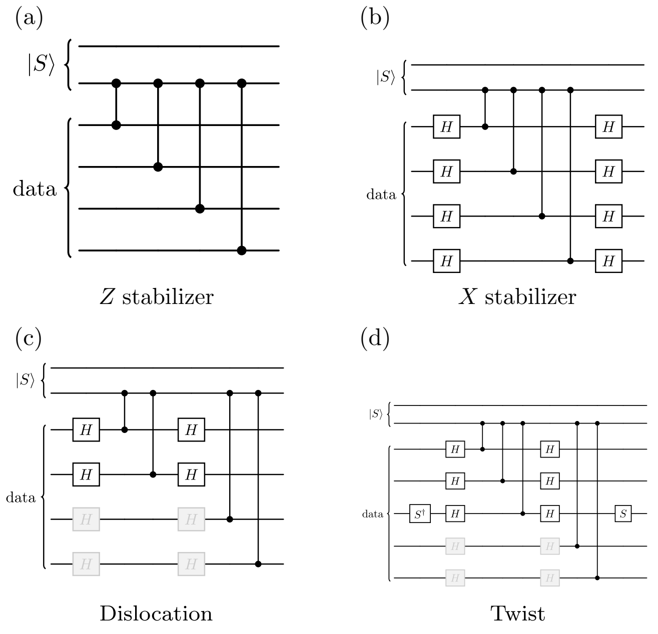

FIG. 6. Circuit implementation of all kinds of stabilizers needed for regular error correction and modified lattice surgery [46], using a singlet-triplet ancilla qubit. This implementation also respects a semiglobal implementation of Hadamard gates, meaning that they are globally applied on all data qubits. The light gray Hadamards are only included for this purpose and cancel each other. (a) Z stabilizer. (b) X stabilizer. (c) Dislocation. (d) Twist.

the case if we used CNOTs for the Z stabilizers (with data qubits as controls and one ancilla line as the target), as it would transform a singlet state into a different triplet state $|T'\rangle = (|00\rangle - |11\rangle)/\sqrt{2}$.

This proves crucial when implementing dislocations and twists. Indeed, as an example, let us consider the measurement of the dislocation operator XZ on data qubits in the $|-\rangle \otimes |1\rangle$ state. This should lead to the final measurement of a singlet state. Let us first assume that the Z parity measurement is implemented with a CNOT targeting one qubit of the ancilla pair (rather than a CZ as in Fig. 6). The quantum state of the data and ancilla system evolves as follows:

$$\begin{aligned}|-\rangle\,|1\rangle\,(|01\rangle - |10\rangle)/\sqrt{2} &\longrightarrow |-\rangle\,|1\rangle\,(|01\rangle + |10\rangle)/\sqrt{2}\\ &\longrightarrow |-\rangle\,|1\rangle\,(|11\rangle + |00\rangle)/\sqrt{2}.\end{aligned}$$

The final ancilla state is a triplet—this implementation gives the wrong measurement outcome. Let us now assume that the Z parity measurement is implemented with a CZ gate, as prescribed in Fig. 6. This time, the quantum state of the data and ancilla system transforms as follows:

$$\begin{aligned}|-\rangle\,|1\rangle\,(|01\rangle - |10\rangle)/\sqrt{2} &\longrightarrow |-\rangle\,|1\rangle\,(|01\rangle + |10\rangle)/\sqrt{2}\\ &\longrightarrow |-\rangle\,|1\rangle\,(|01\rangle - |10\rangle)/\sqrt{2}.\end{aligned}$$

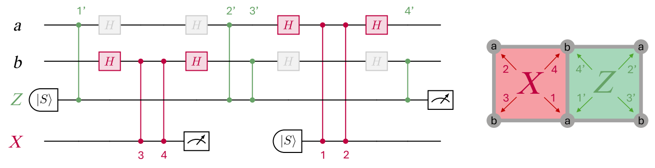

FIG. 7. Circuit implementation of a cycle of X and Z stabilizers over one unit cell of the code. This implementation respects both the ordering avoiding hook errors, and makes use of semiglobal Hadamard gates only. The a/b indexation for the data qubits is chosen so as to respect the periodicity of the lattice. The numbers (1 to 4 for X, 1' to 4' for Z) indicate the order in which gates should be implemented to avoid hook errors.

The ancilla state is thus back to the singlet state, which is the correct behavior.

The twist measurement can be implemented in a similar way as the dislocation. Also, the presence of the S and $S^\dagger$ gates is not problematic as these are phase gates, which can be implemented locally.

3. Full circuit

The two previous sections, respectively, addressed the questions of gate ordering and individual implementation of all types of stabilizers. The last step is now to give a protocol to combine these two elements and obtain the full circuit implementation of a stabilizer cycle. At a given time step, the qubits are scheduled to undergo one of the following operations: shuttling; initialization or measurement; or single- or two-qubit gates. In the latter case, gates can be separated in two sets: gates used for Z parity measurement (single CZ's) (set 1) and gates used for X parity measurement (H-CZ-H sequence) (set 2). We include the Y parity measurement of the twist in set 2 as the S gates can be implemented locally and are thus not problematic. The idea is then simply to perform these two sets of gates one after the other, and the undesired Hadamards will simplify. Note that the same technique was already used in Fig. 6. Additionally, by following the order "set 1–set 2–set 2–set 1" over two consecutive rounds of stabilizer measurements, one can further halve the number of Hadamards. This protocol is illustrated in Fig. 7 in the case of no dislocations or twists (but the same idea would apply if they had to be measured too). Gate set 1 is represented in green and gate set 2 in red.

D. State injection

So far, we have shown how to implement all stabilizers involved in regular error correction as well as lattice surgery on the gate level. The final ingredient needed to enable universal quantum computation is state injection. The first step is to determine which logical states one would want to initialize.

The first one is the ancilla state used for multiqubit measurements. Let us denote P_i (respectively, $P_{i,\text{ancilla}}$) the Pauli operators applied to the ith logical data (respectively, ancilla) qubit. For the logical measurement of $P_1 \otimes \cdots \otimes P_n$, Fig. 44 of Ref. [47] prescribes the initialization of the mediating logical ancilla in the $|+\rangle^{\otimes n-1}$ state, and measurement of the operators $Z_{i,\text{ancilla}} \otimes P_i$ via lattice surgery. In our case, as we can only prepare $|0\rangle$ states, preparing $|+\rangle$ states would require the application of an additional Hadamard. Therefore, it is simplest to initialize the logical ancilla in the $|0\rangle^{\otimes n-1}$ state and measure $X_{i,\text{ancilla}} \otimes P_i$ instead, which is strictly equivalent. Besides, as explained in Ref. [47], the ancilla is insensitive to Z errors, as it is prepared in Z eigenstates. This is particularly handy for us as our code is more prone to Z errors as a result of shuttling (whose precise noise model will be studied in the next subsection).

The second logical state one would want to prepare is the magic state $|m\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$. To do so, we use a modified version of Ref. [48,49]. The main obstacle in the direct implementation of these protocols on our device is the restriction on single-qubit gates. In particular, we cannot prepare a $|+\rangle$ state without applying a global Hadamard. Yet, these schemes require the preparation of the data qubits in both the $|0\rangle$ and $|+\rangle$ states. This issue can be circumvented by staggering the initialization of the data qubits. More concretely, one would first prepare all qubits that should be initialized in the $|+\rangle$ state, as well as the physical magic state, in the $|0\rangle$ state. A global Hadamard is then applied, bringing all these qubits to the $|+\rangle$ state. Then a local phase gate can be applied on the physical magic state, thereby preparing it in the $|m\rangle$ state. All remaining qubits can then be prepared in the $|0\rangle$ state, and the state injection procedure of Ref. [48,49] can be performed normally.

One important thing to note, however, is that whenever a global Hadamard is applied, it does not just target the patch of surface code one intends to prepare, but also all other data qubits in the device. As a result, a transversal H gate is applied to all logical qubits in the middle of the quantum computation. This results in the application of logical Hadamards, as well as the permutation of the X and Z stabilizers of each surface code. While this could seem problematic, these additional Hadamards can easily be undone by the application of another logical Hadamard on all qubits. This is no different than any other Clifford gate in the circuit and can thus be performed virtually, by changing the basis of the following gates and measurements.

E. Performance simulations under realistic noise

In the previous sections, we showed that universal error correction is possible from a theoretical point of view on a $2 \times N$ spin-qubit device with strong constraints, such as the access to semiglobal single-qubit gates only (apart from phase gates). In the following, we argue that our proposition is also practical, i.e., that the required physical error rates and resource requirements for such a device are not experimentally out of reach.

1. Noise model

First, let us discuss the noise model we adopt to estimate the performance of our system. The main ingredient we extensively make use of here and that requires modeling is shuttling. When an electron is shuttled, its g factor varies due to inhomogeneities in the device, which induces unwanted phase rotations in the laboratory reference frame. Systematic rotations can be calibrated away, so that we are effectively describing an electron in a shuttled reference frame [18,50,51]. When the phase is not exactly cancelled out owing to incorrect calibration, one can expect the data qubits of the device to each undergo some small coherent rotation (remember that only data qubits are shuttled). As coherent errors are difficult to simulate classically and potentially more harmful than stochastic errors, twirling is generally used to transform them into Pauli errors [52]. However, this method is unfortunately not fully permitted in our device as some of the single-qubit gates can only be implemented semiglobally. As such, the only natural candidate for a twirling gate set tailored to phase rotations and respecting our constraints is $W = \{I, X_1 \otimes \cdots \otimes X_n\}$ (with I the identity gate, X_i the Pauli X gate on data qubit i, and n the number of data qubits). Nonetheless, one can easily see that a Pauli Z error on an even number of data qubits commutes with both gates of W and will thus not be extracted from the coherent phase rotations by the twirling gates (another, more mathematical justification can be obtained by using Eq. 7 of [53]). Therefore, only odd-order errors (i.e., Pauli Z errors affecting an odd number of qubits) will correctly be twirled. If this poses some difficulties for the exact simulation of our system, it should however not be fatal in terms of performance, as the logical-level noise has been shown to behave similarly for coherent and random errors in large enough surface codes [54].

Therefore, even though the twirling process is not perfect, we still approximate the remaining errors by some random dephasing of probability p_{sh} for simulation purposes, which is correct up to first order (as second-order Z errors are not correctly twirled). p_{sh} here represents the probability that a given data qubit is affected by a Pauli-Z error somewhere within a given stabilizer cycle. To first order it is therefore the sum of four contributions, corresponding to the four shuttles that are carried out within a cycle. Ignoring nonadiabatic effects for now, Eq. 4 of [18] shows that the probability of a dephasing error $\delta\phi^2/2$ *for one shuttle* follows:

$$\delta\phi^2/2 = 2\frac{l_c L_s}{(v T_2^*)^2}, \tag{3}$$

with L_s the shuttling distance, v the shuttling speed, T_2^* the characteristic dephasing time experienced by stationary spins and l_c the coherence length of the dephasing noise due to shuttling. The latter arises from imperfect calibration of the dynamic reference frame used to absorb coherent rotations happening when electrons are shuttled. These calibration errors stem from uncontrolled fluctuations caused by slow nuclear dynamics due to dipolar interaction or low-frequency $1/f$ charge noise affecting the spins' g factors. Now summing the contributions of the four shuttles happening during a stabilizer cycle, one gets

$$p_{\mathrm{adia}} = 2\frac{l_c \times 4d l_{dd}}{(v T_2^*)^2} \tag{4}$$

where l_{dd} is the spacing between two data qubits. The total shuttling distance over a whole stabilizer cycle is roughly $4dl_{dd}$, as we need to shuttle by $2d$ increments and back (remember that we are interleaving logical data qubits and logical ancilla qubits, both of them of distance d, see Fig. 4).

Nonetheless, the above analysis would not be complete if we did not consider dominant nonadiabatic effects. The lowest energy splitting in silicon quantum dots is the valley splitting E_{VS}, which corresponds to the gap between the two out-of-plane conduction bands (or valleys). These two valley states are additionally characterized by distinct g factors [55]. Consequently, an electron in the excited valley state, as opposed to the expected ground valley state, will precess at an unknown frequency, leading to unwanted phase rotations. One can estimate the amount of nonadiabaticity as follows: E_{VS} typically ranges between 10 and 200 μeV for SiGe or can exceed 500 μeV in SiMOS architectures [18]. Furthermore, the two valleys are site-dependent, meaning that they depend on the position $x(t)$ of the electron in the device. Assuming that the length scale of the electron wave function is roughly $l_x = 50$ nm, one can assume that in the worst-case scenario the electron's valley state would switch every 50 nm, leading to a characteristic energy $hv/l_x = 0.83$ μeV, where h is Planck's constant. This is only 12 times smaller than 10 μeV, i.e., the worst-case value for E_{VS}, justifying the inclusion of these nonadiabatic effects in our modeling. Further modeling and simulations found in Appendix B lead to additional dephasing errors due to the valley state of $p_{\mathrm{dia}}/4d = 1.4 \times 10^{-6}$. This value will be used in all the simulations of the paper. The final shuttling-induced dephasing probability is then given by $p_{\mathrm{sh}} = p_{\mathrm{adia}} + p_{\mathrm{dia}}$.

With these considerations in mind, we now have all necessary ingredients to simulate the performance of our device. Our full noise model thus includes (i) dephasing channels of probability $p_{\mathrm{sh}} = p_{\mathrm{adia}} + p_{\mathrm{dia}}$ on the data qubits at the beginning of each stabilizer cycle; (ii) depolarizing channels of error rate p after each gate of the stabilizer circuit (including twirling gates and cancelled out Hadamards arising from their semiglobal implementation); (iii) depolarizing channels of error rate p after any initialization; (iv) classical flip of any measurement outcome with probability p.

Note that here we ignore the idling errors—these account for errors that idle qubits accumulate while others are being operated on. Indeed, spin-coherence times T_2 exceeding 20 ms have been demonstrated for purified silicon electron spins [56], which is 4 orders of magnitude above initialization, measurement, and single- and two-qubit gate times (at most a few microseconds [27,57–59]). These T_2 times are obtained via dynamical decoupling, which can be implemented by periodically flipping all the spins in the device. In our case, it is simplest to flip all spins at the same time, as this is permitted by the semiglobal pulse we are assuming.

In the following simulations, we fix the gate noise p to 0.1%, so as to be below the usual circuit-level noise threshold of the surface code (around 0.7% [60]). This is consistent with experiments showing silicon spin-qubit fidelities exceeding 99.9% [24]. The dephasing time T_2^* will range between 1 μs and 8 μs, which has been achieved [24,61], and is well below demonstrated dephasing times of 100 μs [59]. For the estimation of p_{sh}, we fix $v = 10$ m/s, $l_c = 100$ nm and $l_{dd} = 140$ nm [18]. This choice of 140 nm for the distance between two data qubits takes into account additional space for the clavier gates required by the conveyor-belt mode of shuttling, and leaves extra space for singlet-triplet ancilla qubits and measurement apparatuses in the static row. These values in turn give a dephasing probability *per shuttling increment* $p_{\mathrm{sh}}/4d$ between 5.8×10^{-6} and 2.7×10^{-4}.

The most optimistic end of this range corresponds to a more mature technology than the early demonstrations reported so far, which now approach 10^{-4} [19,20,62]. However, the task of low-noise shuttling is receiving rapidly increasing attention, notably with recent proposals suggesting that g-factor disorder might help improve shuttling fidelities rather than hinder them [63]. We will thus assume that these will keep improving.

As the surface code is a CSS code, X and Z errors can be analyzed separately. Since our code is more prone to Z-type errors, we focus only on them. Therefore, we numerically evaluate the X logical error rate of a single distance-d wide surface code used in memory mode via N_{runs} runs of Monte Carlo simulations, with N_{runs} ranging between 10 000 and 1 000 000 depending on the target logical error rates. In each run, random errors are injected in the code according to the error rates p and p_{sh}, affecting both data and ancilla qubits for d rounds of stabilizer measurements. The syndrome they create is then decoded via minimum-weight perfect matching [1]. The initial errors and the correction are then added to determine if the X logical operator value was flipped.

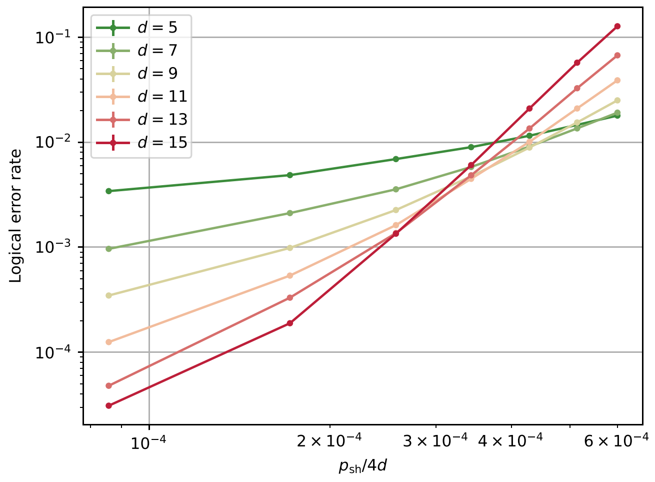

FIG. 8. Logical error rate of a wide surface code against $p_{\mathrm{sh}}/4d$, i.e., the probability that one shuttling increment introduces a dephasing error, plotted for several code distances d. The gate, measurement, and initialization noise p is set to 0.1%.

2. Simulations

We first plot in Fig. 8 the logical error rate for several code distances d, against $p_{\mathrm{sh}}/4d$, i.e., the shuttling error per shuttling increment. The main observation of the plot is that the lines corresponding to surface codes of various distances do not cross at a single point. This is expected as lines crossing at the same point, equivalent to the existence of a threshold at this crossing point, should only appear if the noise model respects certain properties. In particular, the noise strength should not scale with the code size [64]. Yet, here we assumed that $p_{\mathrm{sh}} \propto 4d$. As a result, the crossing points of subsequent-distance surface codes cross at lower and lower values of p. In other words, there does not seem to be any value of p below which increasing the code distance consistently reduces the error rate. Rather, there is a constant trade-off between increasing the code distance for more powerful error correction, and keeping it low enough to keep the noise due to shuttling manageable.

The anticipated nonexistence of a threshold does not present any basic issue for our approach. Since the ideas presented here are intended for the early fault-tolerant era, the relevant quantity is always the logical error rate that can be achieved for realistic experimental parameters and specific system sizes. In Fig. 9, we thus plot the logical error rate against d for T_2^* ranging between 1 μs and 8 μs. The simulation data, represented by small dots with error bars (quantifying the sampling error), is then fitted with the following trial function:

$$p_{\log}(d) = A(\alpha + \beta d)^{\gamma d + \delta}. \tag{5}$$

The motivation for this function is the following. If one denotes p_{tot} the total noise experienced by the system, the surface code promises an exponential suppression of the

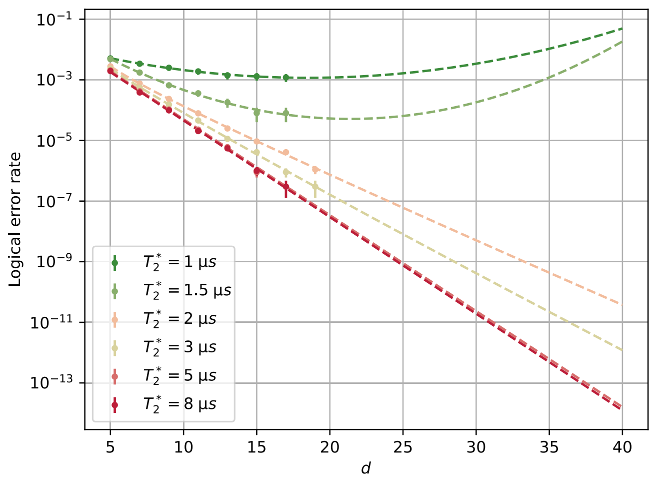

FIG. 9. Logical error rate of a wide surface code against the code distance, plotted for several values of the stationary dephasing time T_2^*. The gate noise p is set to 0.1%. The dots represent the simulation data, their error bars quantifying the sampling error. The dashed lines correspond to a fit of the data according to the trial function of Eq. (5).

errors of the form $p_{\log}(d) \propto (p_{\mathrm{tot}}/p_{\mathrm{th}})^{d/2}$ with p_{th} a constant. Then, up to first order, p_{tot} can be written as a sum of the gate noise p and the shuttling noise p_{sh}, where the latter is proportional to the code distance d. The consequence of p_{tot}'s dependence on d is a subexponential suppression of errors, or worse, an increase in the error rate when the distance of the code becomes too large. Fortunately, with dephasing time that is large enough yet experimentally plausible, low enough error rates (around 10^{-13}) can be reached before this phenomenon occurs.

F. Resource estimation for universal quantum computation

With all the above ingredients, we can now estimate the resources required for our device to achieve two meaningful milestones. In all the following, we set a spins' stationary dephasing time $T_2^* = 8$ μs, a per-dot valley noise $p_{\mathrm{dia}}/4d = 1.4 \times 10^{-6}$ and a gate noise $p = 0.1\%$. The first task we focus on is estimating the size N that would bring the $2 \times N$ architecture beyond the noisy intermediate-scale quantum (NISQ) regime. More concretely, we want to estimate N such that the device would contain 50 logical qubits, able to interact through a universal set of gates, and with logical error rates of at most 0.01% (ten times lower than physical error rates). Using Fig. 9, one can see that distance-9 wide surface codes are sufficient. Moreover, following our state injection protocol (Sec. III D), the probability of preparing a magic state in the wrong state is of the order of the physical error rate of the operations that created it, that is p and p_{sh}. To first order, the probability p_{mag} of initializing an erroneous logical magic state is thus of the order of $p + p_{\mathrm{sh}} = 0.103\%$ (ignoring small constant factors, see Eq. 1 of [49]). This lowers to $35p_{\mathrm{mag}}^3 = 4 \times 10^{-8}$ after a 15-to-1 distillation protocol—way below our target logical error rate of 0.01%. To meet our above requirements, one would therefore need $2 \times (50 + 15) = 130$ distance-9 surface code patches (including logical data qubits and magic state distillation factory, and doubling this number to account for the logical ancilla bus connecting all these logical qubits, see Fig. 4). This translates into an overall size of $N = 2 \times 10^4$ physical data qubits.

Beyond simply outperforming NISQ on the paper, what one would actually want from a fault-tolerant quantum computer is to be able to run meaningful quantum algorithms. The second task we will thus focus on is estimating the ground-state energy of a 2D Hubbard model Hamiltonian. This can be solved by preparing the unitary $\mathcal{W}(H) = e^{i \arccos(H)}$ by qubitization, then passing it to phase estimation. An optimized protocol to run this algorithm can be found in Ref. [65] along with the resource requirements for the smallest instance of this problem that is within the classically intractable regime, i.e., a 6×6 Hubbard model. For this grid size, 100 logical qubits and 10^8 logical T gates would be required.

Let us set a spins' stationary dephasing time $T_2^* = 8$ μs and a gate noise $p = 0.1\%$. Let us assume that both the probability of a logical error impacting *any* individual T gate, and the probability of an error impacting *any* of the logical qubits in the circuit, is below 10%. The algorithm thus fails with a probability of around 20%. The quantum computation can then be run multiple times in parallel to exponentially increase the success probability. Following the procedure of [47], one can estimate the physical resources needed to obtain the correct outcome.

a. Magic state distillation. The error rate of each individual T gate must be under 10^{-9} to ensure that the failure probability of any of the 10^8 gates is under 10%. Again, the probability of initializing an incorrect logical magic state before distillation is $p_{\mathrm{mag}} \sim p + p_{\mathrm{sh}}$. Assuming that the distance of our surface code is under 40 (which we will verify later on), we guarantee $p_{\mathrm{sh}} < 0.02\%$. Adding the gate noise, we obtain a total noise $p_{\mathrm{mag}} \lesssim 0.12\%$. Using the 116-to-12 distillation protocol, the probability of outputting incorrectly distilled magic states is $41.25p_{\mathrm{mag}}^4 \lesssim 10^{-10}$, which is under our target per-gate error of 10^{-9}. 44 surface codes patches are needed to run the 116-to-12 distillation protocol, which outputs 12 logical magic states in $99d$ time steps with 89% success probability [47]. The initialization time of one correct magic state is thus on average $9.27d$ time steps. The magic states can then be consumed one by one in the quantum computation, which requires d time steps for each state. The overall time cost of the quantum computation (to prepare and consume 10^8

magic states) is thus $10.27d \times 10^8$. It requires 100 logical qubits to store the logical information and 44 surface-code patches in the distillation block (not counting the logical ancilla). Adding the long logical ancilla bus mediating all interactions, the logical qubit count doubles to 288.

b. Code distance. Given the total number of logical qubits and the computation time, in order to set the probability of a logical error affecting any of the logical qubits during the computation under 10%, one requires the per-logical-qubit error rate p_L to be below

$$p_L < 0.1/(288 \times 10.27 \times 10^8) = 3 \times 10^{-13}. \qquad (6)$$

By using the fitting function of Fig. 9, we can choose the code distance to be $d = 36$. This is indeed within the upper bound of 40 that we set earlier.

c. Summary. Therefore, in order to run one instance of a 6×6 Hubbard model (which is within the classically intractable regime) on our device, one would need 288 wide qubits of distance $d = 36$, that is 1.4×10^6 physical (data and ancilla) qubits. The algorithm would run in $10.27d \times 10^8$ stabilizer cycles.

The total shuttling time over a stabilizer cycle is $t_{\text{sh}} = 4dl_{dd}/v$, where $l_{dd} = 140$ nm is the distance between two consecutive data qubits, and $v = 10$ m/s is the shuttling speed. Thus $t_{\text{sh}} = 2$ μs. Besides, while two-qubit gates can be implemented much faster (around 0.1 μs [57]), initialization, measurement and single-qubit gates all take around a microsecond [27,58,59]. We can thus realistically assume an overall stabilizer cycle time of 6 μs, which would bring the computation time to a total of 2.5 days. While this time may seem relatively daunting, one must remember that we exclusively used experimental parameters that have been achieved, ignoring any further optimization that will surely happen in the following years before this kind of device can be implemented. Our shuttling noise model, informed by earlier theoretical studies (esp. Ref. [18]), assumes levels of imperfection that are smaller than existing early demonstrations; we are confident that improving experimental techniques driven by rapidly increasing community interest will reach the domain that we have simulated. From Fig. 9, one observes that for the more optimistic coherence times we have a scenario where shuttling noise is among the least significant factors in the model (as there is very little variation in the logical error rate when T_2^* is decreased from 8 μs to 5 μs, i.e., when the shuttling error per increment varies between 6×10^{-6} and 1.3×10^{-5}). Further increasing T_2^* to experimentally achieved values of 20 μs [66] would thus only have marginal effects on the performance here. Rather, a limiting factor seems to be the gate infidelity $p = 0.1\%$. If this quantity could be improved by only a factor 2, logical error rates would drastically reduce due to almost exponentially scaling suppression (as long as the shuttling noise is kept low enough, e.g., by enforcing coherence times of 20 μs this time). Moreover, the assumed speed of the various operations corresponds to already-accomplished demonstrations, and does not begin to approach any fundamental physical limitations. One might reasonably expect orders of magnitude improvement over generations of such devices.

IV. QUANTUM MEMORIES WITH GENERAL QLDPC CODES

In Sec. II we showed that our device is suitable for the implementation of various classes of qLDPC codes beyond the surface code. These codes are known for their high performance and can fully exploit the connectivity (beyond that required by the surface code), which is provided by our shuttling-based system. Therefore, in this section, we explore the potential of such nonlocal codes, constructed by either allowing more ancilla-data qubit interactions, or by increasing the shuttling distance. Here, we only study their use as memory codes.

The gate ordering we choose in the stabilizer circuits is not optimized. All X stabilizers will be implemented when the data qubits are shuttled one way, followed by all Z stabilizers when the data qubits are shuttled back. Regarding the noise model, we adopt the same as in the previous section—circuit-level noise plus additional dephasing due to shuttling (proportional to the shuttling distance). Due to the more complex data-ancilla interactions and to our nonoptimal gate ordering, not all qubits perform their entangling operations synchronously (while it is the case for the surface code). Consequently, the data qubit rail might have to stop and start more often, and qubits that are not interacting will have to stay idle. To model this, we introduce an additional noise parameter p_{idle}, and add depolarizing channels of probability p_{idle} every time a qubit is idle while others are interacting. Note that we neglected this in the previous section due to the high coherence time of silicon spin qubits.

The numerical experiments are run under various values of the gate noise p, the idling noise p_{idle} and the dephasing time T_2^*. The per-dot valley noise is still fixed to $p_{\text{dia}}/4d = 1.4 \times 10^{-6}$. Each code is simulated for d rounds of stabilizer cycles, where d is the code's estimated distance. The decoding is performed via *min-sum* BP-OSD, a limit of 32 iterations, a scaling factor of 0.625 and a serial schedule [8]. The logical error rate *per round per logical qubit* p_{log} defined below is then plotted and used to compare the performance of various codes [7]:

$$p_{\text{log}} = 1 - (1 - P_L(k,d))^{1/kd}. \qquad (7)$$

Here, $P_L(k,d)$ is the probability that a logical error happens on at least one of the k logical qubits of an $[n,k,d]$ code running for d rounds of stabilizer cycles.

A. Hypergraph product code with a repetition code

As explained in Sec. II C, one class of codes that is particularly suitable for our architecture are the hypergraph product (HGP) codes constructed with the repetition code as one of the seed codes. Indeed, let us suppose that shuttling is performed in the vertical direction, and that a HGP code is built from a classical LDPC code placed in the direction of shuttling, and a repetition code placed in the orthogonal direction. Let us first assume that both codes have distance d. It follows that horizontal interactions have length at most 1 in the 2D grid (length of the repetition code's interactions), which translates into length d in the linearized $2 \times N$ architecture. Therefore, any ancilla qubit has to interact with data qubits that are at most d increments away, and compared to the surface code, the shuttling distance is not increased. Rather, the only difference with the latter is that the data qubit row has to stop more often, as stabilizers may have higher weights, but also as it may not be possible to synchronize the interactions as much as for the surface code (adopting the framework of Sec. II, the code's check matrix has more nonzero diagonals). Therefore, implementing such HGP code comes at the cost of increased idling noise (but not increased shuttling noise). This additional noise can be compensated for if a *good* classical code is chosen for the product with the repetition code, as a higher-rate quantum code will be generated, thereby reducing the qubit overhead.

In the following, we study the example of a [234,3,8] HGP code. The code is obtained by taking the product of a classical [17,3,8] LDPC code H_1 (generated randomly until good parameters were obtained, see Appendix C) and a distance-8 repetition code H_2. To limit the number of two-qubit gates, which are expected to be the leading source of noise, we generate H_1 by enforcing two 1's per row on average, thereby guaranteeing that the stabilizers of the quantum code are weight-four on average. This code is more compact than the surface code as it encodes $k = 3$ logical qubits with $(17+14) \times (8+7) = 465$ physical qubits (data and ancilla). In contrast, one would require $k \times (2d-1)^2 = 675$ physical qubits to encode the same amount of logical information with same-distance surface codes.

Figure 10 compares the logical error rates per round per logical qubit $p_{\log}$ of the [234,3,8] HGP code and of surface codes of various distances. The shuttling noise is fixed by setting $T_2^* = 8$ μs. As explained above, the HGP code experiences the same amount of shuttling error as the surface code, but higher idling noise. Indeed, its two-qubit gates cannot be implemented synchronously due to the randomness of the matrix H_1 (it contains many nonzero diagonals). Qubits that are not interacting thus have to stay idle, which induces depolarizing errors at a rate p_{idle}. We thus plotted $p_{\log}$ for two levels of idling errors: $p_{\text{idle}} = 0$ and $p_{\text{idle}} = 0.1p$. One can observe that for no idling errors and for a relevant range of gate noise p, the HGP code performs similarly as surface codes with comparable distances, yet requiring 30% less qubits. As expected however, when the idling noise is increased, the HGP code's performance falls dramatically while the surface code's is very moderately affected.

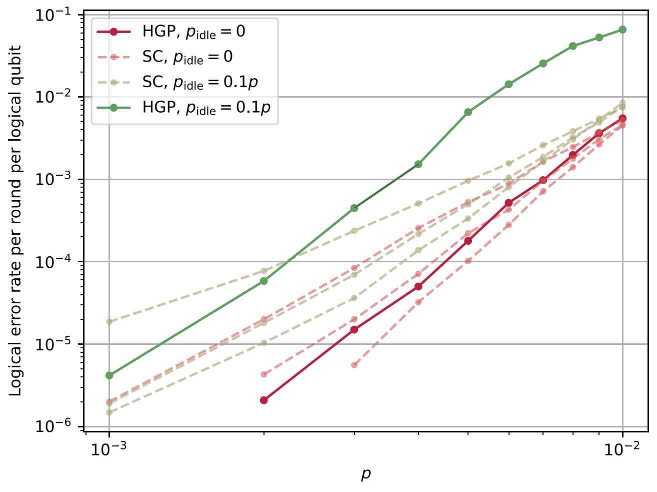

FIG. 10. Logical error rate per round per logical qubit of a [234,3,8] hypergraph product code (HGP, solid lines) and surface codes of various distances d (SC, dashed lines), against the circuit-level noise parameter p. The HGP code is built from the product of a repetition code and a classical code generated randomly. The dephasing time T_2^* is set to 8 μs, and the logical performance of the HGP and surface codes are plotted for $p_{\text{idle}} = 0$ and $0.1p$. For both levels of idling noise, the three surface-code lines correspond from top to bottom to a distance of 5, 7, and 9, respectively.

While observing this, one needs to take into account that we did not optimize the stabilizer circuit. An interleaved scheme of X and Z measurements could drastically reduce the overall idling noise. Besides, in the specific case of silicon spin qubits, p_{idle} is expected to be extremely small. Indeed, given the experimental values observed for the coherence time T_2, around 20 ms [56], and the slower gate times, around $T_{\text{gate}} = 1$ μs [27], one would obtain an idling error probability of

$$p_{\text{idle}} = 1 - e^{-T_{\text{gate}}/T_2} = 5 \times 10^{-5}.$$

When using this value, one would obtain the same qualitative performance as for the $p_{\text{idle}} = 0$ case.

Furthermore, here we set a single parameter p_{idle} quantifying the idling error. It would be more comprehensive to distinguish different p_{idle}'s depending on the operation that is being waited for. Specifically, as two-qubit gates are one order of magnitude faster than single-qubit gates, they would be responsible for shorter wait times, thus lower idling errors. Indeed, it so happens that the difference between the HGP code and the surface code lies in the additional idling times due to asynchronous two-qubit gates, which are the fastest operations in the circuit.

Lastly, do note that the crossing point of the surface-code lines around 1% is not a threshold in the common sense, as we are here plotting the logical error rate per round per logical qubit.

B. Generalized bicycle codes

In the previous section, we studied a type of code that did not require an increase of the shuttling distance compared to the surface code, but we did not set any constraint on the number of stops along the way. We therefore guaranteed both a shuttling and idling noise scaling as $O(\sqrt{n})$, where n is the number of data qubits in the code block. Here we study another paradigm, where we do not restrict the shuttling distance, but instead bound the number of stops: $p_{\mathrm{sh}} = O(n)$ and $p_{\mathrm{idle}} = O(1)$. Codes that satisfy this are generalized bicycle codes [8], as their check matrices by definition contain a constant number of nonzero diagonals (see Sec. II C).

In the following, we study the performance of a [126,28,8] generalized bicycle code (code A2 of [8]). This code contains 252 physical qubits (data and ancilla). Using the same number of physical qubits to encode $k = 28$ logical qubits, one would need to use 28 surface codes of distance

$$d_{\mathrm{sc}} = \frac{\sqrt{252/k} + 1}{2} = 2.$$

This very small number stems from the relatively high rate of the code. However, tackling larger distances would require simulating much bigger codes, which are beyond our numerical capabilities.

Similarly to the previous subsection, we plot in Fig. 11 the logical error rate per round per logical qubit of the [126,28,8] generalized bicycle code and compare it to a distance-3 surface code (the smallest surface code that can correct, not just detect, errors). We set $p_{\mathrm{idle}} = 0.01p$ (with p the measurement, initialization, and gate noise) and plot the logical performance of the codes for different amounts of shuttling noise, which are parameterized by the static dephasing time T_2^*. Compared to the previous subsection, idling noise does not dramatically impact the bicycle code, as it is formed from matrices with only five nonzero diagonals. However, the code is more prone to shuttling errors as the shuttling distance is now of the order of the number of qubits n (while it is of the order $d = \sqrt{n}$ for the surface code).

One can see that for low enough (yet experimentally achievable) shuttling noise, the [126,28,8] code outperforms surface codes with similar qubit overhead for gate

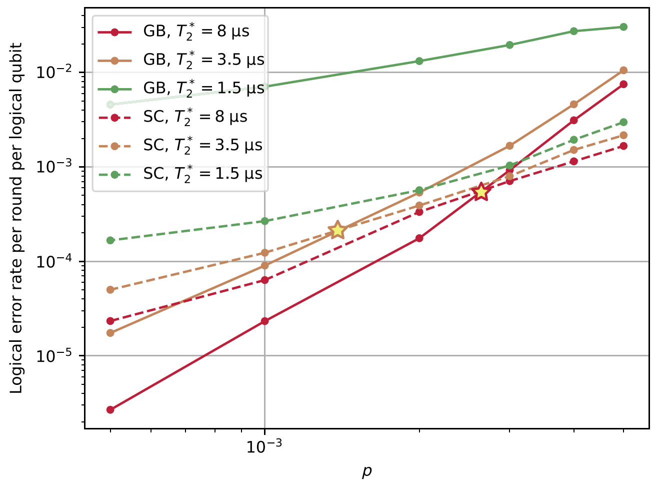

FIG. 11. Logical error rate per round per logical qubit of a [126,28,8] generalized bicycle code [8] (GB, solid lines) and a distance-3 surface code (SC, dashed lines), against the circuit-level noise parameter p. The logical performance of both codes is plotted for $p_{\mathrm{idle}} = 0.01p$ and different levels of shuttling noise, controlled by the dephasing time T_2^*. For $T_2^* = 3.5$ µs and 8 µs, the noise level where generalized bicycle codes start to beat surface codes is indicated by a star.

noise as high as 0.3%, which is also within reach. If the gate noise can be further reduced to 0.05%, the general bicycle code would provide an advantage of around one order of magnitude compared to a distance-3 surface code (which also uses more qubits than the generalized bicycle code, as we were meant to compare with distance 2). Nevertheless, the performance of the generalized bicycle code is as expected more sensitive to the shuttling noise than the surface code, as it involves longer shuttles. But for the considered code size, the effect only becomes predominant for relatively low dephasing times. Of course, when considering larger codes, the demands on the dephasing time would have to be adjusted accordingly, to compensate for the even longer shuttles.

C. Comparative performance

We here confirm the observations of this section via additional simulations aiming at further understanding the trade-off between resource requirements and error-correction performance. Specifically, we examine the performance of the three previously considered code families under three different noise regimes: (A) low idling noise but high shuttling noise; (B) low shuttling noise but high idling noise; and (C) high idling noise and high shuttling noise. The results of such simulations are presented in Table I, where the theoretical parameters of the code are given, along with the logical error rate per round per logical qubit in the three noise cases. We aim to compare codes with similar overhead, defined as the total number of physical qubit per encoded logical qubit, i.e., $(n + n_{\mathrm{anc}})/k$. In

TABLE I. Comparative performance of three code families: surface code (SC), hypergraph product code (HGP), and generalized bicycle code (GB). See the main text for the precise definition of each code. The first three and last two rows, respectively, correspond to lower- and higher-overhead codes, where the overhead is defined as the number of physical qubits per logical qubits. The second column contains said overhead. The third column is the code distance. The fourth and fifth columns contain the shuttling and idling noise scalings of each code. Finally, in the last three columns the logical error rate per round per logical qubit is given for three different noise regimes: (A) $T_2^* = 1.5$ μs, $p_{\text{idle}} = 0$; (B) $T_2^* = 5$ μs, $p_{\text{idle}} = 0.1p$; and (C) $T_2^* = 1.5$ μs, $p_{\text{idle}} = 0.1p$. In each of these columns, the highest-performance low-overhead code, as well as the highest-performance high-overhead code is highlighted with bold letters.

	$\frac{n+n_{\text{anc}}}{k}$	d	Shuttling noise scaling	Idling noise scaling	A	B	C
SC3	25	3	$O(\sqrt{n})$	$O(1)$	2×10^{-4}	1×10^{-4}	$\mathbf{3 \times 10^{-4}}$
GB	9	8	$O(n)$	$O(1)$	7.10^{-3}	$\mathbf{5 \times 10^{-5}}$	7.10^{-3}
HGP4	26	4	$O(\sqrt{n})$	$O(\sqrt{n})$	$\mathbf{4 \times 10^{-5}}$	1.10^{-3}	1.10^{-3}
SC7	169	7	$O(\sqrt{n})$	$O(1)$	1×10^{-5}	$\mathbf{3 \times 10^{-6}}$	$\mathbf{1 \times 10^{-5}}$
HGP8	155	8	$O(\sqrt{n})$	$O(\sqrt{n})$	$\mathbf{8 \times 10^{-6}}$	1×10^{-5}	2×10^{-5}

the first half of the table, we compare low-overhead codes: a distance-3 surface code (SC3), the [126,28,8] generalized bicycle code considered before (GB8) and a [40,3,4] hypergraph product code (HGP4). Its check matrix is given in Appendix C. In the second half of the table, we focus on higher-overhead codes: a distance-7 surface code (SC7) and the [234,3,8] hypergraph product code considered before (HGP8). Note that in this case, we could not simulate *good* generalized bicycle codes due to limited computational power. In each of the last three columns, the lowest error rate is highlighted, confirming the conclusions of the section: each code family considered in this paper excels in distinct noise regimes. Specifically, GB codes perform best when the shuttling noise is kept low; HGP codes do when the idling noise is low; and the SC wins when neither of them is sufficiently small. In the earliest fault-tolerant regimes, the latter scenario might be the most relevant, which justifies the use of the surface code, for which we also proved full computational power in the previous section.

V. DISCUSSION

We have investigated the feasibility of quantum error correction in a highly constrained experimental setup, specifically a $2 \times N$ array of qubits where long-range interactions are enabled by shuttling one of the rows of the array. By establishing a precise framework for determining code classes that naturally embed in our device, we showed that its strong constraints should theoretically not be an obstacle to early fault tolerance. This observation was then complemented by the design of an entire protocol permitting full universal quantum computation with the surface code.

To show the practicality of our approach, we further tailored our protocol to silicon spin qubits, respecting their additional specificities and constraints, such as the difficulty to implement local Hadamard gates. This choice of platform was motivated by the high-fidelity shuttling capabilities that have been demonstrated [19,20,62], making it an ideal candidate for the implementation of our scheme. We then confirmed these theoretical protocols with extensive numerical simulations, showing that error rates as low as 10^{-13} can be reached with experimental parameters that have been achieved today, provided currently observed shuttling errors can be further suppressed to match our theoretical models. While this entails long quantum computations for running algorithms in the classically intractable regime, we are confident that the gate and shuttling performance will further improve in the coming years, making our proposal even more practical.

Furthermore, our exploration extends to the application of our device to more intricate qLDPC codes. Although more powerful than the surface code, their implementation comes at the cost of increased noise (either from longer shuttles or higher idling times). By simulating these codes and evaluating their performance in our constrained setup, we observed that this trade-off is in favour of the complex qLDPC codes when error rates are low enough (while still practically achievable). This underscores the versatility of our platform, even within the limitations of the $2 \times N$ qubit array.

As future work, it would be interesting to engineer a slightly more complex architecture design that would let one include both the surface and better qLDPC codes in the same device and interface them. It would indeed prove advantageous to make use of the latter's strong error-correction capabilities when used as memory codes. Whenever a logical qubit is idle for a significant amount of time during a quantum computation, one could take its logical information and store it in memory, before taking it out again when needed. If this store in-and-out scheme can theoretically be implemented via a modified lattice surgery protocol between a surface code and another type of qLDPC code [10], one would still need to understand how to efficiently embed this in our device.

More generally, throughout this paper we have chosen to work within quite severe limitations, i.e., the low-dimensional array and the restriction to semiglobal

Hadamard operations. We have established that even within these constraints computation is possible. Nonetheless, without further improvement, scaling up the current device to larger instances may prove very challenging. We identify two reasons for this. First, even for quantum algorithms slightly beyond the classically intractable regime, the runtimes reach several days. This is partially due to our suboptimal choice of logical qubit layout, where the logical information is stored in the first row of surface codes while the second row is only used as a logical ancilla bus (Fig. 4). While enabling all-to-all connectivity, this structure also introduces connectivity lockups that slow down the computation: for instance, when the leftmost and rightmost logical qubits are interacting through the ancilla bus, no other interaction can take place in parallel. On top of this, the second obstacle to large-scale computation in our current proposal is the sensitivity of the device to malfunctioning qubits. If a dot becomes inoperable, e.g., due to a fabrication defect, and qubits cannot shuttle through it anymore, the array will be cut in two halves that cannot communicate anymore. This would likely lead to a failure of the algorithm.

To circumvent both the aforementioned issues, we envisage an extension of the strictly $2 \times N$ qubit array presented in this paper: a lattice or weblike structure where long $2 \times N$ filaments meet at occasional three- or four-way junctions through which qubits can shuttle. Such junctions could be well spaced, so that the additional device complexity associated with each junction need not overlap with others. In addition to avoiding the problem of a single point of failure, such a geometry would no doubt afford a rich space of possibilities for novel codes and compilation strategies, thereby reducing runtimes. They would additionally offer superior opportunities to interface the aforementioned better qLDPC codes with surface codes. This is an intriguing direction for future investigation.

ACKNOWLEDGMENTS

The authors are grateful to numerous supportive colleagues and especially Simon Benjamin and Fernando González Zalba. They would also like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility [67] in carrying out this work and specifically the facilities made available from the EPSRC QCS Hub Grant (Agreement No. EP/T001062/1). The authors also acknowledge support from EPSRC's Robust and Reliable Quantum Computing (RoaRQ) project (EP/W032635/1), and EPSRC's Software Enabling Early Quantum Advantage (SEEQA) project (EP/Y004310/1). This research has also received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 951852.

APPENDIX A: SURFACE CODE'S STABILIZER CIRCUIT GATE ORDERING

Here we delve into the details of finding an optimal gate ordering for surface-code error correction on the $2 \times N$ architecture. Indeed, the usual pattern that is used for the regular rotated surface code with N- and Z-shaped orderings [68] leads to unnecessarily long shuttles. To see this, assume that the shuttling direction is vertical—two consecutive data qubits within the same column (respectively, row) of the surface code are thus separated by 1 (respectively, d) shuttling increment(s). Therefore, a Z-shaped ordering would require to shuttle four times along rows, and it would yield a total shuttling distance of roughly $4d$. Our aim is to find an ordering that lets us implement all gates of the stabilizer cycle with a total shuttling distance of roughly $2d$ (which corresponds to having the data qubits do one round trip, not two).

Furthermore, note that N- and Z-shaped orderings mentioned above do lead to distance-reducing hook errors in the wide surface code used in Sec. III. Indeed, its X and Z logical operators both have a horizontal and a vertical representative. Instead, one can measure the stabilizers as shown in Fig. 12. The shortest Z logical operators are horizontal, vertical, or diagonal. The latter type passes through the centers of X stabilizers (red squares). However, using the measurement schedule represented by the gray arrow, Z hook errors are on either diagonal of the Z stabilizers (green squares), thus do not reduce the code distance. The same applies to X hook errors. Besides, with the same reasoning, one can prove that the regular rotated surface code is protected just as well from hook errors when this

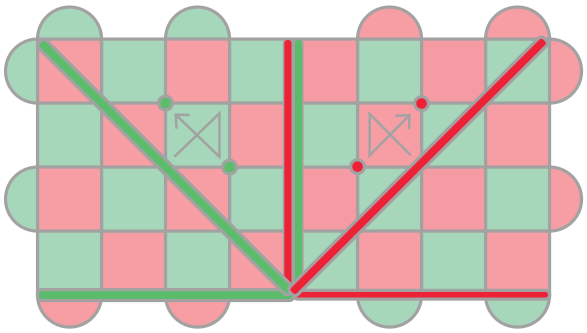
FIG. 12. Representation of the wide surface code. X and Z stabilizers are, respectively, represented with red and green squares or half-disks. Several shortest-length representatives of the logical X and Z operators are drawn with horizontal, vertical, and diagonal lines. The gray arrows represent an order in which X and Z stabilizers can be measured to avoid hook errors reducing the distance of the code. Examples of X and Z hook errors are, respectively, drawn with red and green disks: as they do not coincide with the logical operators, they do not reduce the distance of the code. The same applies to the regular rotated surface code.

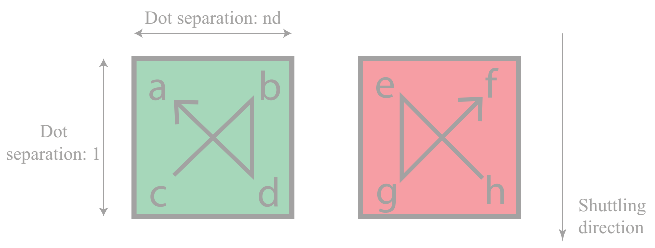

FIG. 13. Notations for conditions (A1)–(A3).

ordering is used. In summary, whether it is for the regular (Fig. 2) or wide (Fig. 12) surface code, this crosslike sequence is the one we consider.

Now, one could theoretically implement it by measuring the X and Z stabilizers separately on alternating rounds. However, this would unnecessarily increase the circuit depth and leave many qubits idle. Instead, a common solution is to interleave the gates of both stabilizers. This is possible provided the gates can be correctly commuted through each other so as to leave two neighboring ancilla qubits disentangled (condition A1); as well as gates can be implemented synchronously respecting the device layout and global shuttling of the data qubits [condition (A2)].

Let us explain this more formally, and call time step k the interval between shuttles $k-1$ and k. At a given time step, some entangling gates must be implemented between certain ancilla-data qubit pairs. Let us use the notations of Fig. 13, where each letter represents the time step when the ancilla and corresponding data qubit must be entangled.

Condition (A1) is verified if and only if [46]

$$((b < e) \wedge (d < g)) \vee ((b > e) \wedge (d > g)). \qquad \text{(A1)}$$

As for condition (A2), it reads

$$(a \equiv e[s]) \wedge (b \equiv f\,[s]) \wedge (c \equiv g[s]) \wedge (d \equiv h[s]), \quad \text{(A2)}$$

where s is the number of shuttles required to go back to the data qubits' initial position. This is because the device is laid out periodically and data qubits move as a whole along their shuttling track. This means that at any time step, ancilla qubits all face either their North-West, or North-East, or South-West, or South-East data qubit.

One last condition (A3) can be added, enforcing the no-distance-reducing-hook-error ordering, which mathematically reads as

$$(c < b < d < a) \wedge (h < e < g < f\,). \qquad \text{(A3)}$$

Values for the time steps a to f respecting all three conditions are given in Fig. 5 (here $s = 4$). While their gates are indeed interleaved, X and Z stabilizers are nonetheless operated on a staggered fashion. The whole operation sequence, including gates, measurements, initializations, and shuttles is the following:

(1) entangle all ancilla qubits with their South-West data qubit
(2) shuttle by $d-1$ increments forwards
(3) entangle all ancilla qubits with their North-East data qubit; measure and reinitialize X ancilla qubits
(4) shuttle by one increment forwards
(5) entangle all ancilla qubits with their South-East data qubit
(6) shuttle by $d+1$ increments backwards
(7) entangle all ancilla qubits with their North-West data qubit; measure and reinitialize Z ancilla qubits
(8) shuttle by one increment forwards
(9) repeat

Note that the shuttling increments given here do not include the additional shuttling accommodating the ancilla bus of region B in Fig. 4 ($n = 1$). Besides, for the first round of stabilizer measurements, X ancilla qubits should only start to undergo their entangling gates from step 3. Similarly, for the last round, only X stabilizers should follow step 1 and 2.

With this protocol, in order to perform N_r rounds of X and Z stabilizer measurements, one thus needs $4N_r + 1$ shuttles and a total shuttling distance of $N_r(2d+2) + d - 1$ (not including the logical ancilla bus of Fig. 4). One can also easily see that $4N_r$ and $N_r(2d+2)$ are the respective lower bounds for the number of shuttles and total shuttling distance, no matter what gate ordering is chosen. Indeed, a given four-body stabilizer trivially requires four steps to entangle the ancilla with all the data qubits, thus four shuttles. Moreover, its North-West and South-East data qubits are separated by a distance $d+1$, hence going back and forth between them requires a shuttling distance of $2d+2$. Therefore, apart from a small correction arising at the last round due to the staggered implementation of the X and Z stabilizers, our solution is optimal.

APPENDIX B: MODELING FOR THE VALLEY DEGREE OF FREEDOM

The band structure of bulk silicon features six degenerate conduction bands: four in plane and two out of plane. In the presence of gates in SiMOS or Si heterostructures, the six-fold degeneracy is lifted and the two out-of-plane bands become the lowest of the six: they are called the ground and excited valley states. Ideally always sitting in the ground valley state, an electron that is shuttled fast enough can nonadiabatically populate the excited valley state. As these states have been shown to exhibit distinct g factors [55], the electron could then start to precess in an uncontrolled manner, causing unwanted phase rotations. This justifies the importance to control and estimate the impact of the excited valley state occupation. To model this, we will use an extension of the valley state modeling in Ref. [18].

Let us first introduce the position-dependent valley phase $\varphi_{\mathrm{VS}}(x)$ and the bare valley splitting $E_{\mathrm{VS},0}$ (which models the valley splitting in the absence of the perturbations described below). The local two-level valley Hamiltonian can most generally be expressed as

$$H_{\mathrm{loc}}(x) = \frac{E_{\mathrm{VS},0}}{2}\left(\cos(\varphi_{\mathrm{VS}}(x))\tau_x + \sin(\varphi_{\mathrm{VS}}(x))\tau_y\right), \quad \text{(B1)}$$

where τ_x and τ_y are Pauli operators in the valley subspace.

Assuming low coupling of the spatial and valley degrees of freedom, an electron of spatial probability distribution $\rho(x)$ would thus experience an average valley Hamiltonian:

$$H_v(x_0) = \int \rho(x - x_0) H_{\mathrm{loc}}(x_0)\mathrm{d}x. \quad \text{(B2)}$$

Here we suppose that ρ is a Gaussian of standard deviation l_x. Two extreme cases can be considered for φ_{VS}: smoothly varying or abruptly changing owing to the presence of atomic steps. In the $v = 10$ m/s regime, Fig. 1 of Ref. [18] shows that the smooth interface model leads to higher noise: this is thus the model we will adopt.

That paper focuses on the simplest case of a linear gradient model: $\varphi_{\mathrm{VS}}(x) = a_x x$. We slightly refine it by assuming that the gradient is not constant but instead slowly varying at the scale of the electron wave function. This means that we can adopt all equations derived in Ref. [18], while assuming a slow variation of a_x to take disorder into account. In the instantaneous valley state basis, the final valley+orbital Hamiltonian is thus the following:

$$\tilde{H}_v(x_0) = \left(\frac{E_{\mathrm{VS}}(x)}{2}\tau_z + \frac{\dot{\varphi}_{\mathrm{VS}}(x)}{2}\tau_x\right)\otimes I + \frac{\Delta g(x)}{2}\frac{I-\tau_z}{2}\otimes\sigma_z \quad \text{(B3)}$$

with

$$E_{\mathrm{VS}}(x) = E_{\mathrm{VS},0}\exp\left(-\frac{a_x(x)^2 l_x^2}{4}\right) \quad \text{(B4)}$$

$$\dot{\varphi}_{\mathrm{VS}}(x) = \dot{a}_x(x)x + a_x(x)v(x) \quad \text{(B5)}$$

σ_z is the Pauli operator characterizing the spin, $(I-\tau_z)/2$ is the projector onto the excited valley state and $\Delta g(x)$ models the difference in g factor between the ground and excited valley states. We assume that the electron is smoothly shuttled back and forth along $2d$ dots separated by a distance l_{dd} (to account for the ancilla bus of Fig. 4), such that

$$v(x) = \begin{cases} v, & \text{if } x < 2dl_{dd} \\ -v, & \text{otherwise} \end{cases} \quad \text{(B6)}$$

and

$$x(t) = \begin{cases} vt, & \text{if } t < 2dl_{dd}/v \\ -vt + 4dl_{dd}, & \text{if } 2dl_{dd}/v \le t < 4dl_{dd}/v \end{cases}. \quad \text{(B7)}$$

To model the disorder, we describe $a_x(x)$ with a smooth random walk of the form:

$$a_x(x) = \sum_{k=1}^{n}\alpha_k \sin(\lambda_k x) \quad \text{(B8)}$$

with $n = 20$ and λ_k chosen randomly between l_x and the maximum shuttled distance $2dl_{dd}$. These bounds guarantee a slow variation of the valley parameters on the scale of the electron wave function but over the whole landscape explored via shuttling. In the worst-case scenario, the ground and excited valley states should swap every l_x, meaning a variation of the valley phase of π. Thus a worst-case value for a_x is π/l_x. For an n-step random walk with unitary steps ($\alpha_k = 1$), the average maximum distance is $\langle \max_x a_x\rangle = \sqrt{n\pi/2}$. We therefore randomly sample α_k between 0 and

$$\alpha_{\max} = \frac{\pi}{l_x}\sqrt{\frac{2}{n\pi}} = \frac{1}{l_x}\sqrt{\frac{2\pi}{n}} \quad \text{(B9)}$$

so that with high probability a_x does not exceed π/l_x. The g-factor difference between ground and excited valley states $\Delta g(x)$ is modeled similarly with

$$\Delta g(x) = \sum_{k=1}^{n}\beta_k \sin(\mu_k x), \quad \text{(B10)}$$

where $n = 20$ and μ_k is a random number between l_x and $2dl_{dd}$. A typical maximum value for Δg at a constant field of 1 T is 100 MHz [38], thus we decide to randomly sample each β_k between 0 and $\sqrt{2/n\pi} \times 100$ MHz.

Finally, in order to understand the phase accumulation over a whole stabilizer cycle, we prepare the valley+spin electron wave function in

$$|\psi_0\rangle = |0\rangle \otimes |+\rangle. \quad \text{(B11)}$$

We assume a shuttling speed $v = 10$ m/s and a dot separation $l_{dd} = 140$ nm as in the main text. The electron spatial distribution is set to $l_x = 50$ nm, and we choose $E_{VS,0} = 150$ μeV, so that in the worst case of $a_x l_x = \pi$, one gets $E_{\mathrm{VS}} = 15$ μeV, which in the lowest possible range for E_{VS} [18] (therefore increasing nonadiabatic errors). We aim to estimate the influence of the code size d on the valley-induced dephasing noise, as it controls the overall shuttling distance. Further, we study the effect of additionally flipping the spin state via an X gate at the turning point in the electron trajectory, in the spirit of dynamical decoupling. Indeed, ignoring any disorder, we observe that a flipped spin simply precesses back to its initial state when shuttled back. Hence one may expect that the same phenomenon happens in the presence of disorder and could therefore be used to reduce the noise. The evolution of the wave function over time is obtained by solving Schrödinger equation by time discretization over 50 000 time steps. The final probability of a Z error is given by the overlap of the final state with the $|-\rangle$ spin state. In the presence of random disorder, we estimate the dephasing probability via Monte Carlo simulations, by repeating the same experiment $N_{\text{reps}} = 10\,000$ times.

Figure 14 shows the evolution of the valley-induced dephasing probability p_{dia} against the code distance d, both with and without flipping the spin at the turning point of the trajectory. One can see that the error rate is not lowered by such manipulation: we therefore will not execute this operation. By fitting the *no flip* data with a simple linear regression $y = \gamma x$, one can infer a per-dot error probability $p_{\text{dia}}/4d = 1.4 \times 10^{-6}$. As a comparison, the per-dot error induced by adiabatic shuttling for $T_2^* = 8$ μs, i.e., in the most optimistic case considered in the main text, is $p_{\text{adia}}/4d = 4 \times 10^{-6}$. This justifies the addition of such nonadiabatic effects in our modeling.

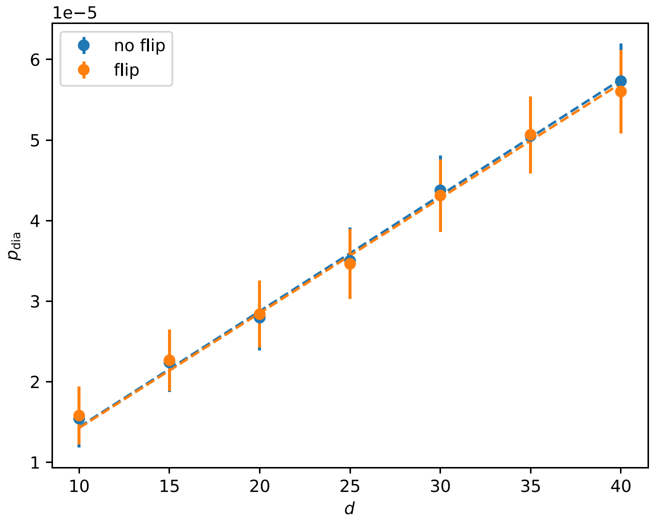

FIG. 14. Solid lines: valley-induced dephasing error p_{dia} against the code distance d, to which the shuttling distance is proportional. Two cases are plotted, depending on if the spin is flipped via an X before the electron is shuttled back. Dashed lines: linear regression of both cases.

APPENDIX C: GOOD MATRIX FOR HGP CODE

In Secs. IV A and IV C, we simulated the performance of a HGP constructed from the product of a [8,1,8] repetition code and a [17,3,8] classical code generated randomly. Its parity-check matrix H_8 is

$$H_8 = \begin{pmatrix} 0&1&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0 \\ 1&0&0&0&1&0&0&1&0&0&0&0&1&0&1&0&0 \\ 1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&1&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1 \end{pmatrix}.$$

In Sec. IV C, we followed the same protocol with a HGP code constructed from the product of a [4,1,4] repetition code and a [7,3,4] classical code generated randomly. Its parity-check matrix H_4 is

$$H_4 = \begin{pmatrix} 1&0&0&1&1&0&1 \\ 0&0&1&0&1&0&1 \\ 0&1&1&0&0&1&0 \\ 1&0&1&0&1&1&0 \end{pmatrix}.$$

[1] 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).

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

[3] Google Quantum AI, Exponential suppression of bit or phase errors with cyclic error correction, Nature **595**, 383 (2021).

[4] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, ArXiv:quant-ph/9811052.

[5] A. Y. Kitaev, Quantum computations: Algorithms and error correction, Russ. Math. Surv. **52**, 1191 (1997).

[6] N. P. Breuckmann, C. Vuillot, E. Campbell, A. Krishna, and B. M. Terhal, Hyperbolic and semi-hyperbolic surface codes for quantum storage, Quantum Sci. Technol. **2**, 035007 (2017).

[7] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, High-threshold and low-overhead fault-tolerant quantum memory, Nature **627**, 778 (2024).

[8] P. Panteleev and G. V. Kalachev, Degenerate quantum LDPC codes with good finite length performance, Quantum **5**, 585 (2019).

[9] O. Higgott and N. P. Breuckmann, Improved single-shot decoding of higher-dimensional hypergraph-product codes, PRX Quantum **4**, 020332 (2023).

[10] Q. Xu, J. P. B. Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang, and H. Zhou, Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays, Nat. Phys. **20**, 1084 (2024).

[11] J. Viszlai, W. Yang, S. F. Lin, J. Liu, N. Nottingham, J. M. Baker, and F. T. Chong, Matching generalized-bicycle codes to neutral atoms for low-overhead fault-tolerance, ArXiv:2311.16980.

[12] C. A. Pattison, A. Krishna, and J. Preskill, Hierarchical memories: Simulating quantum LDPC codes with local gates, ArXiv:2303.04798.

[13] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. B. Ataides, N. Maskara, I. Cong, X. Gao, P. S. Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin, Logical quantum processor based on reconfigurable atom arrays, Nature **626**, 58 (2023).

[14] Z. Cai, A. Siegel, and S. Benjamin, Looped pipelines enabling effective 3D qubit lattices in a strictly 2D device, PRX Quantum **4**, 020345 (2023).

[15] Y. Li and S. C. Benjamin, One-dimensional quantum computing with a “segmented chain” is feasible with today’s gate fidelities, npj Quantum Inf. **4**, 25 (2018).

[16] A. T. E. Shaw, M. J. Bremner, A. Paler, D. Herr, and S. J. Devitt, Quantum computation on a 19-qubit wide 2D nearest neighbour qubit array, ArXiv:2212.01550.

[17] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis, Demonstration of the trapped-ion quantum CCD computer architecture, Nature **592**, 209 (2021).

[18] V. Langrock, J. A. Krzywda, N. Focke, I. Seidler, L. R. Schreiber, and L. Cywiński, Blueprint of a scalable spin qubit shuttle device for coherent mid-range qubit transfer in disordered Si/SiGe/SiO$_2$, PRX Quantum **4**, 020305 (2023).

[19] J. Yoneda, W. Huang, M. Feng, C. H. Yang, K. W. Chan, T. Tanttu, W. Gilbert, R. C. C. Leon, F. E. Hudson, K. M. Itoh, A. Morello, S. D. Bartlett, A. Laucht, A. Saraiva, and A. S. Dzurak, Coherent spin qubit transport in silicon, Nat. Commun. **12**, 4114 (2021).

[20] I. Seidler, T. Struck, R. Xue, N. Focke, S. Trellenkamp, H. Bluhm, and L. R. Schreiber, Conveyor-mode single-electron shuttling in Si/SiGe for a scalable quantum computing architecture, npj Quantum Inf. **8**, 100 (2022).

[21] X. Xue, M. Russ, N. Samkharadze, B. Undseth, A. Sammak, G. Scappucci, and L. M. K. Vandersypen, Quantum logic with spin qubits crossing the surface code threshold, Nature **601**, 343 (2022).

[22] A. Noiri, K. Takeda, T. Nakajima, T. Kobayashi, A. Sammak, G. Scappucci, and S. Tarucha, Fast universal quantum gate above the fault-tolerance threshold in silicon, Nature **601**, 338 (2022).

[23] A. R. Mills, C. R. Guinn, M. J. Gullans, A. J. Sigillito, M. M. Feldman, E. Nielsen, and J. R. Petta, Two-qubit silicon quantum processor with operation fidelity exceeding 99%, Sci. Adv. **8**, eabn5130 (2022).

[24] J. Yoneda, K. Takeda, T. Otsuka, T. Nakajima, M. R. Delbecq, G. Allison, T. Honda, T. Kodera, S. Oda, Y. Hoshi, N. Usami, K. M. Itoh, and S. Tarucha, A quantum-dot spin qubit with coherence limited by charge noise and fidelity higher than 99.9%, Nat. Nanotechnol. **13**, 102 (2017).

[25] J. O’Gorman, N. H. Nickerson, P. Ross, J. J. Morton, and S. C. Benjamin, A silicon-based surface code quantum computer, npj Quantum Inf. **2**, 15019 (2016).

[26] K. Takeda, A. Noiri, T. Nakajima, T. Kobayashi, and S. Tarucha, Quantum error correction with silicon spin qubits, Nature **608**, 682 (2022).

[27] S. G. J. Philips, M. T. Madzik, S. V. Amitonov, S. L. de Snoo, M. Russ, N. Kalhor, C. Volk, W. I. L. Lawrie, D. Brousse, L. Tryputen, B. P. Wuetz, A. Sammak, M. Veldhorst, G. Scappucci, and L. M. K. Vandersypen, Universal control of a six-qubit quantum processor in silicon, Nature **609**, 919 (2022).

[28] R. Maurand, X. Jehl, D. Kotekar-Patil, A. Corna, H. Bohuslavskyi, R. Laviéville, L. Hutin, S. Barraud, M. Vinet, M. Sanquer, and S. De Franceschi, A CMOS silicon spin qubit, Nat. Commun. **7**, 13575 (2016).

[29] A. M. J. Zwerver, *et al.*, Qubits made by advanced semiconductor manufacturing, Nat. Electron. **5**, 184 (2022).

[30] X. Xue, *et al.*, CMOS-based cryogenic control of silicon quantum circuits, Nature (London) **593**, 205 (2021).

[31] A. Ruffino, T.-Y. Yang, J. Michniewicz, Y. Peng, E. Charbon, and M. F. Gonzalez-Zalba, A cryo-CMOS chip that integrates silicon quantum dots and multiplexed dispersive readout electronics, Nat. Electron. **5**, 53 (2022).

[32] M. F. Gonzalez-Zalba, S. de Franceschi, E. Charbon, T. Meunier, M. Vinets, and A. S. Dzurak, Scaling silicon-based quantum computing using CMOS technology, Nat. Electron. **4**, 872 (2021).

[33] M. Veldhorst, H. G. J. Eenink, C. H. Yang, and A. S. Dzurak, Silicon CMOS architecture for a spin-based quantum computer, Nat. Commun. **8**, 1766 (2017).

[34] J. M. Boter, J. P. Dehollain, J. P. van Dijk, Y. Xu, T. Hensgens, R. Versluis, H. W. Naus, J. S. Clarke, M. Veldhorst, F. Sebastiano, and L. M. Vandersypen, Spiderweb array: A sparse spin-qubit array, Phys. Rev. Appl. **18**, 024053 (2022).

[35] L. Hutin, B. Bertrand, E. Chanrion, H. Bohuslavskyi, F. Ansaloni, T.-Y. Yang, J. Michniewicz, D. J. Niegemann, C. Spence, T. Lundberg, A. Chatterjee, A. Crippa, J. Li, R. Maurand, X. Jehl, M. Sanquer, M. F. Gonzalez-Zalba, F. Kuemmeth, Y.-M. Niquet, S. De Franceschi, M. Urdampilleta, T. Meunier, and M. Vinet, in *2019 IEEE International Electron Devices Meeting (IEDM)* (2019), pp. 37.7.1–37.7.4, iSSN: 2156-017X.

[36] L. C. Camenzind, S. Geyer, A. Fuhrer, R. J. Warburton, D. M. Zumbühl, and A. V. Kuhlmann, A hole spin qubit in a fin field-effect transistor above 4 K, Nat. Electron. **5**, 178 (2022).

[37] X. Xue, T. F. Watson, J. Helsen, D. R. Ward, D. E. Savage, M. G. Lagally, S. N. Coppersmith, M. A. Eriksson, S. Wehner, and L. M. K. Vandersypen, Benchmarking gate fidelities in a Si/SiGe two-qubit device, Phys. Rev. X **9**, 021011 (2019).

[38] S. M. Patomäki, M. F. Gonzalez-Zalba, M. A. Fogarty, Z. Cai, S. C. Benjamin, and J. J. L. Morton, Pipeline quantum processor architecture for silicon spin qubits, npj Quantum Inf. **10**, 1 (2024).

[39] E. Kawakami, P. Scarlino, D. R. Ward, F. R. Braakman, D. E. Savage, M. G. Lagally, M. Friesen, S. N. Coppersmith, M. A. Eriksson, and L. M. K. Vandersypen, Electrical control of a long-lived spin qubit in a Si/SiGe quantum dot, Nat. Nanotechnol. **9**, 666 (2014).

[40] O. Crawford, J. R. Cruise, N. Mertig, and M. F. Gonzalez-Zalba, Compilation and scaling strategies for a silicon quantum processor with sparse two-dimensional connectivity, npj Quantum Inf. **9**, 13 (2023).

[41] E. Vahapoglu, J. P. Slack-Smith, R. C. C. Leon, W. H. Lim, F. E. Hudson, T. Day, T. Tanttu, C. H. Yang, A. Laucht, A. S. Dzurak, and J. J. Pla, Single-electron spin resonance in a nanoelectronic device using a global field, Sci. Adv. **7**, eabg9158 (2021).

[42] C. Jones, M. A. Fogarty, A. Morello, M. F. Gyure, A. S. Dzurak, and T. D. Ladd, Logical qubit in a linear array of semiconductor quantum dots, Phys. Rev. X **8**, 021058 (2018).

[43] B. Klemt, V. Elhomsy, M. Nurizzo, P. Hamonic, B. Martinez, B. Cardoso Paz, C. Spence, M. C. Dartiailh, B. Jadot, E. Chanrion, V. Thiney, R. Lethiecq, B. Bertrand, H. Niebojewski, C. Bäuerle, M. Vinet, Y.-M. Niquet, T. Meunier, and M. Urdampilleta, Electrical manipulation of a single electron spin in CMOS using a micromagnet and spin-valley coupling, npj Quantum Inf. **9**, 107 (2023).

[44] A. C. Betz, R. Wacquez, M. Vinet, X. Jehl, A. L. Saraiva, M. Sanquer, A. J. Ferguson, and M. F. Gonzalez-Zalba, Dispersively detected Pauli spin-blockade in a silicon nanowire field-effect transistor, Nano Lett. **15**, 4622 (2015).

[45] A. West, B. Hensen, A. Jouan, T. Tanttu, C.-H. Yang, A. Rossi, M. F. Gonzalez-Zalba, F. Hudson, A. Morello, D. J. Reilly, and A. S. Dzurak, Gate-based single-shot readout of spins in silicon, Nat. Nanotechnol. **14**, 437 (2019).

[46] D. Litinski and F. v. Oppen, Lattice surgery with a twist: Simplifying Clifford gates of surface codes, Quantum **2**, 62 (2018).

[47] D. Litinski, A game of surface codes: Large-scale quantum computing with lattice surgery, Quantum **3**, 128 (2019).

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

[49] L. Lao and B. Criger, in *Proceedings of the 19th ACM International Conference on Computing Frontiers, CF '22* (Association for Computing Machinery, New York, NY, USA, 2022), p. 113.

[50] B. Buonacorsi, Z. Cai, E. B. Ramirez, K. S. Willick, S. M. Walker, J. Li, B. D. Shaw, X. Xu, S. C. Benjamin, and J. Baugh, Network architecture for a topological quantum computer in silicon, Quantum Sci. Technol. **4**, 025003 (2019).

[51] B. Buonacorsi, B. Shaw, and J. Baugh, Simulated coherent electron shuttling in silicon quantum dots, Phys. Rev. B **102**, 125406 (2020).

[52] J. J. Wallman and J. Emerson, Noise tailoring for scalable quantum computation via randomized compiling, Phys. Rev. A **94**, 052325 (2016).

[53] Z. Cai and S. C. Benjamin, Constructing smaller Pauli twirling sets for arbitrary error channels, Sci. Rep. **9**, 11281 (2019).

[54] S. Bravyi, M. Englbrecht, R. König, and N. Peard, Correcting coherent errors with surface codes, npj Quantum Inf. **4**, 55 (2018).

[55] J. D. Cifuentes, *et al.*, Bounds to electron spin qubit variability for scalable CMOS architectures, Nat. Commun. **15**, 4299 (2024).

[56] M. Veldhorst, J. C. C. Hwang, C. H. Yang, A. W. Leenstra, B. de Ronde, J. P. Dehollain, J. T. Muhonen, F. E. Hudson, K. M. Itoh, A. Morello, and A. S. Dzurak, An addressable quantum dot qubit with fault-tolerant control-fidelity, Nat. Nanotechnol. **9**, 981 (2014).

[57] B. M. Maune, M. G. Borselli, B. Huang, T. D. Ladd, P. W. Deelman, K. S. Holabird, A. A. Kiselev, I. Alvarado-Rodriguez, R. S. Ross, A. E. Schmitz, M. Sokolich, C. A. Watson, M. F. Gyure, and A. T. Hunter, Coherent singlet-triplet oscillations in a silicon-based double quantum dot, Nature **481**, 344 (2012).

[58] G. Zheng, N. Samkharadze, M. L. Noordam, N. Kalhor, D. Brousse, A. Sammak, G. Scappucci, and L. M. K. Vandersypen, Rapid gate-based spin read-out in silicon using an on-chip resonator, Nat. Nanotechnol. **14**, 742 (2019).

[59] K. Takeda, A. Noiri, T. Nakajima, L. C. Camenzind, T. Kobayashi, A. Sammak, G. Scappucci, and S. Tarucha, Rapid single-shot parity spin readout in a silicon double quantum dot with fidelity exceeding 99%, npj Quantum Inf. **10**, 22 (2024).

[60] R. Raussendorf, J. Harrington, and K. Goyal, Topological fault-tolerance in cluster state quantum computation, New J. Phys. **9**, 199 (2007).

[61] G. Burkard, T. D. Ladd, A. Pan, J. M. Nichol, and J. R. Petta, Semiconductor spin qubits, Rev. Mod. Phys. **95**, 025003 (2023).

[62] M. D. Smet, Y. Matsumoto, A.-M. J. Zwerver, L. Tryputen, S. L. de Snoo, S. V. Amitonov, A. Sammak, N. Samkharadze, Önder Gül, R. N. M. Wasserman, M. Rimbach-Russ, G. Scappucci, and L. M. K. Vandersypen, High-fidelity single-spin shuttling in silicon, ArXiv:2406.07267.

[63] S. Bosco, J. Zou, and D. Loss, High-fidelity spin qubit shuttling via large spin-orbit interactions, PRX Quantum **5**, 020353 (2024).

[64] E. Knill, R. Laflamme, and W. H. Zurek, Resilient quantum computation: Error models and thresholds, Proc. R. Soc. London, A **454**, 365 (1998).

[65] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, Encoding electronic spectra in quantum circuits with linear T complexity, Phys. Rev. X **8**, 041015 (2018).

[66] T. Struck, A. Hollmann, F. Schauer, O. Fedorets, A. Schmidbauer, K. Sawano, H. Riemann, N. V. Abrosimov, Å. CywiÅski, D. Bougeard, and L. R. Schreiber, Low-frequency spin qubit energy splitting noise in highly purified ^{28}Si/SiGe, npj Quantum Inf. **6**, 40 (2020).

[67] A. Richards, University of Oxford Advanced Research Computing, 10.5281/zenodo.22558 (2015).

[68] Y. Tomita and K. M. Svore, Low-distance surface codes under realistic quantum noise, Phys. Rev. A **90**, 062320 (2014).