Methods to simulate fermions on quantum computers with hardware limitations

Steudtner, M.

Chapter 4

Quantum error correction in Crossbar architectures

4.1 Background

Spin qubits in silicon quantum dots are a promising platform for quantum computation. Isotopically enriched silicon-28 not only promises long coherence times, but also the compatibility with semiconductor manufacturing techniques. Offering a high qubit density, silicon quantum dots are an important chance for the future of quantum computing. However, controlling a vast amount of qubits is nontrivial. Out of an array of $N \times N$ quantum dots, we would have to be able to select singular qubits for quantum gates and measurements. Those operations can be performed by manipulating electric potentials on or around the corresponding dots, for which gate lines have to connect the corresponding elements with a classical interface. This typically means to steer $O(N^2)$ elements in the bulk of the dot array from its boundary, which only has space to connect $O(N)$ gate lines. The solution for this mismatch is known from classical electronics: a crossbar switch allows to address single elements in a matrix of components by the use of certain row and column lines. For the quantum case, a similar strategy can be adopted, and so only $O(N)$ gate lines are necessary to control the entire grid. The idea is that grid operations like quantum gates and measurements only happen where pulsed lines connect to the same physical elements, such that individual qubit control is achievable. The price to pay for this is a reduced ability to perform operations on different units in the grid in parallel. For clas-

sical systems this is not a fundamental problem, but when the computational units are qubits, whose information decays over time, parallelism becomes absolutely essential. This introduces a formidable roadblock for the development of crossbar systems for quantum computing systems. Nevertheless various crossbar architectures for quantum computers have been proposed in the past [18, 76–79]. This chapter is focusing on our proposal for crossbar-controlled spin quantum dots in silicon [18].

Any realistic quantum computing device, including the one we propose in [18], will suffer from noise processes that degrade quantum information. This noise can be combated by quantum error correction [80, 81], where quantum information is encoded redundantly in such a way that errors can be diagnosed and remedied as they happen without disturbing the encoded information. Many quantum error correction codes have been developed over the last two decades and several of them have desirable properties such as high noise tolerance, efficient decoders and reasonable implementation overhead. Of particular note are the planar surface [82] and color codes [83], which can be implemented in quantum computing systems in which only nearest-neighbor two-qubit gates are available.

However these codes, and all other quantum error correction codes, are developed under the assumption that all physical qubits participating in the code can be controlled individually in parallel. While practical large-scale quantum computers most likely pose control limitations, surprisingly little work has been done in this area [72]. Here we investigate the minimal amount of parallel control resources needed for quantum error correction in the proposed architecture [18].

4.2 Results

- We analyze the crossbar architecture we propose in [18]. We give a full description of the layout and control characteristics of the architecture in a manner accessible to non-experts in quantum dots. We develop a language for describing operations in the crossbar system. Of particular interest here are the regular patterns (see e.g. Section 4.4.4) that are implied by the crossbar structure. These configurations provide an abstraction on which we build mappings

of quantum error correction codes (see below) This analysis is particular to the system in [18] but we believe many of the considerations to hold for more general crossbar architectures.

- We map the planar surface code and the $6.6.6.$ (hexagonal) and $4.8.8.$ (square-octagonal) color codes [83] to the crossbar architecture, taking into account its limited ability to perform parallel quantum operations. The tools we develop for describing the mapping, in particular the configurations described in Section 4.4.4, should be generalizable to other quantum error correction codes and general crossbar architectures.

- Due to experimental limitations the mappings mentioned above might not be attainable in near term devices. Therefore we adapt the above mappings to take into account practical limitations in the architecture [18]. In this version of the mapping the length of an error correction cycle scale with the distance of the mapped code. This means the mapping does not allow for arbitrary logical error rate suppression. Therefore we analyze the behavior of the logical error rate with respect to estimated experimental error parameters and find that the logical error rate can in principle be suppressed to below 10^{-20} (an error rate comparable to the error rate of classical computers [84]), allowing for practical quantum computation to take place.

- Our work raises several interesting theoretical questions regarding the mapping of quantum algorithms to limited control settings, see Section 4.7.

In Section 4.3 we introduce the architecture proposed in [18]. We forgo a deeper discussion of the device physics and only regard its peculiarities as abstract control aspects. We aim to explain the operation of the device in a largely self-contained manner accessible to non experts in quantum dot physics. For that purpose introduce classical helper objects such as the BOARDSTATE matrix which will aid later developments. We discuss one- and two-qubit operations, measurements, and qubit shuttling. In Section 4.4 we focus on difficulties inherent in parallel operation

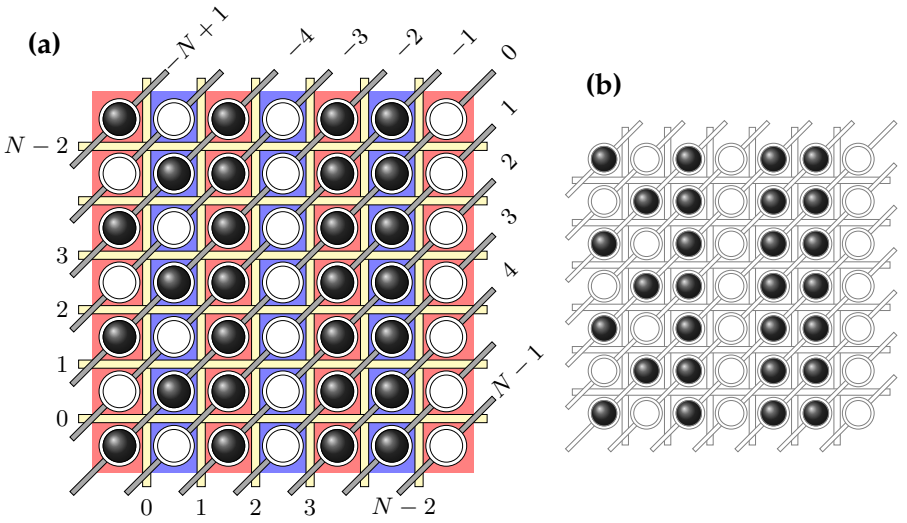


Figure 4.1. (a) A schematic of the Quantum Dot Processor (QDP) that we propose [18], see Section 4.3.1 for details. The white circles correspond to quantum dots, with the black filling denoting the presence of electrons, whose spins are employed as qubits. All dots are found in either red or blue columns, representing areas of different magnetic field. Single qubit gates can only be applied globally on either all qubits in all blue columns or all qubits in all red columns. The vertical, horizontal (both yellow) and diagonal lines (gray) are a feature of this crossbar scheme. The horizontal and vertical gate lines implement barriers that isolate the dots from each other. The diagonal lines simultaneously control the dot potentials of all dots coupled to one line. Quantum operations are effected by pulsing individuals lines. In order to perform two-qubit operations on adjacent dots, one typically needs to lower the barrier that separates them and change the dot potentials by operating the diagonal lines. Note that two-qubit gates applied to adjacent qubits in the same column are inherently different (by nature of the QDP design) from two-qubit gates between two adjacent qubits in the same row. With the control lines, we can also move qubits from dot to dot and measure them. However, since each control line influences $O(N)$ qubits, individual qubit control, as well as parallel operation on many qubits is limited. **(b)** Abstracted version of the QDP scheme representing the classical BOARDSTATE matrix. The BOARDSTATE holds no quantum information, but encodes where qubits are located on the QDP grid.

of the crossbar system. We also introduce several BOARDSTATE configurations which feature prominently in quantum error correction mappings. We describe how these configurations can be reached efficiently by parallel shuttling. In Section 4.5.3 we bring together all previous sections and devise a mapping of the planar surface code to the crossbar architecture. This we continue in Section 4.5.4 for the $6.6.6.$ and $4.8.8.$ color codes. Finally in Section 4.6 we analyze in detail the logical error probability of the surface code mapping as a function of the code distance and estimated error parameters of the crossbar system.

4.3 The quantum dot processor

In this section we will give an overview of the quantum dot processor (QDP) architecture as proposed in [18]. Although this chapter considers the concrete realization of quantum dots in silicon, our main focus is going to lie on its crossbar control structure. Therefore, we will not engage too much with the physics of the host system, but abstract its peculiarities into operational properties as they are relevant for our purposes of controlling this device. The basic organization of the QDP is that for an $N \times N$ grid of qubits interspersed with control lines that effect operations on the qubits. The most notable feature of the QDP (and crossbar architectures in general) is the fact that any classical control signal sent to a control line will be applied simultaneously to all qubits along it. This means that every possible classical instruction applied to the QDP will affect $O(N)$ qubits (these qubits will not necessarily be physically close to each other). This has important consequences for the running of quantum algorithms on the QDP (or any crossbar architecture) that must be taken into account when compiling these algorithms to hardware level instructions. Notably it places strong restrictions on performing quantum operations in parallel on the QDP. To deal with these restrictions it is important to have an understanding of how operations are performed on the QDP. For this reason that we begin our study of the QDP with an examination of its control structure at the hardware level. We describe the physical layout of the system and develop nomenclature for the fundamental control operations. This nomenclature might be called the 'machine code' of the QDP. From these basic instructions we go on to construct all elementary operations that can be applied to qubits. These are quantum operations, such as single qubit gates, nearest-neighbor two-

qubit gates and qubit measurements but also a non-quantum operation called coherent shuttling which does not affect the quantum state of the QDP qubits but changes their connectivity graph (i.e. which qubits can be entangled by two-qubit gates). All of these operations are restricted by the nature of the control architecture in a way that gives rise to interesting patterns (Section 4.4.4) and which we will fully examine in Section 4.4.

4.3.1 Layout

A schematic overview of the QDP architecture is given in Figure 4.1, where qubits (which are electrons, denoted by black balls) occupy an array of $N \times N$ quantum dots. The latter are denoted by white dots when empty, since they either are occupied by a qubit or not. We will label the dots by tuples containing row and column indices $(i, j) \in [0 : N-1]^{\otimes 2}$ beginning from the *bottom left* corner[1]. We assume all qubits to be initialized in the state $|0\rangle$. For future reference we note that $|0\rangle$ corresponds to the spin-up state and $|1\rangle$ to the spin-down state of the electron constituting the qubit.

Typically we will work in a situation where half the dots are occupied by a qubit and half the dots are empty (as seen in Figure 4.1 (a)). Because (as we discuss in Section 4.3.3.1) the qubits can be moved around on the grid, it is important to keep track of which dots contain qubits and which ones do not. This can be done efficiently in classical side-processing. To this end we introduce the BOARDSTATE object. BOARDSTATE consists of a binary $N \times N$ matrix with a 1 as the (i, j)-th entry if the (i, j)-th dot contains an electron and 0 otherwise. The BOARDSTATE does not contain information about the qubit state, only about the electron occupation of the grid. A particular BOARDSTATE is illustrated in the left panel of Figure 4.1.

We now turn to describing the control structures that are characteristic for this architecture. As a first feature, we would like to point out that each dot is either located in a red or a blue region in Figure 4.1 (left panel). The blue (red) columns correspond to regions of high (low) magnetic fields, which plays a role in the addressing of qubits for single qubit gates. We

[1] This is a difference from last chapter's notation, where we started counting from 1, and the components of the index appeared in the opposite order to resemble euclidean coordinates.

will denote the set of qubits in blue columns (identified by their row and column indices) by $\mathcal{B}$ and the set of qubits in red columns by $\mathcal{R}$.

Much finer groups of dots can be addressed by the control lines that run through the grid. The crossbar architecture features control lines that are connected to $O(N)$ dots. At the intersections of these control lines individual dots and qubits can be addressed. This means that using $O(N)$ control lines $O(N^2)$ qubits can be controlled. As seen in Figure 4.1 the rows and columns of the QDP are interspersed with horizontal and vertical lines (yellow), as a means to control the tunnel coupling between adjacent dots. We refer to those lines as barrier gates, or barriers for short. Each line can be controlled individually, but a pulse has an effect on all $O(N)$ dot pairs it separates. Another layer of control lines is used to address the dots itself rather than the spaces in between them. The diagonal gate lines (gray), are used to regulate the dot potential. We label the horizontal and vertical lines by an integer running from 0 to $N-2$ and the diagonal lines with integers running from $-N+1$ to $N-1$ where the $-(N-1)$-th line is the top-left line and increments move towards the bottom right (see Figure 4.1(a)). We count horizontal and vertical lines starting at zero from the lower left corner of the grid (see Figure 4.1). Note that the barriers at the boundary of the grid are never addressed in our model and are thus not labeled. Next we describe how all control lines can be used to effect operations on the qubits occupying the QDP grid.

4.3.2 Control and addressing

As described above, the QDP consists of quantum dots interspersed with barriers and connected by diagonal lines. For our purposes these can be thought of as abstract control knobs that apply certain operations to the qubits. In this section we will describe what type of gates operations are possible on the QDP. We will not concern ourselves with the details of parallel operation until Section 4.4.

There are three fundamental operations on the QDP which we will call the "grid operations". These operations are "lower vertical barrier" (V), "lower horizontal barrier" (H) and "set diagonal line" (D). The first two operations are essentially binary (on-off) but the last one (D) can be set to a value $t \in [0 : T]$ where T is a device parameter. At the physical

OPCODE	Effect
$\mathrm{V}[i]$	Lower vertical barrier at index i
$\mathrm{H}[i]$	Lower horizontal barrier at index i
$\mathrm{D}[i][t]$	Set diagonal line at index i to value t

Table 4.1. Table of grid operations.

level this corresponds to how many clearly distinct voltages we can set the quantum dot plunger gates [18]. Although the actual pulses on those gates differ by amplitude and duration between the different gates and operations, this notation gives us a clear idea which lines are utilized. This can be done because realistically one will not interleave processes in which pulses have such different shapes. We can label the grid operations by mnemonics (which in a classical analogy we will call OPCODES) as seen in Table 4.1. These OPCODES are indexed by an integer parameter that indicates the label of the control line it applies to.

We indicate parallel operation of a collection of OPCODES by ampersands, e.g. $\mathrm{D}[1]\&\mathrm{H}[2]\&\mathrm{D}[5]$. The three grid operations are summarized in Table 4.1. These grid operations can be used to induce some elementary quantum gates and operations on the qubits in the QDP. Below we describe these operations.

4.3.3 Elementary operations

Here we give a short overview of the elementary operations available in the QDP. We will describe basic single qubit gates, two-qubit gates, the ability to move qubits around by coherent shuttling [20] and a measurement process through Pauli Spin Blockade (PSB) [85]. All of these operations are implemented by a combination of the grid operations defined in Table 4.1, and are inherently dependent on the BOARDSTATE .

4.3.3.1 Coherent qubit shuttling

An elementary operation of the QDP is the coherent qubit shuttling [20, 86], of one qubit to an adjacent, empty dot. That means that an electron (qubit) is physically moved to the other dot utilizing at least one diagonal line and the barrier between the two dots. It thereby does not play a role whether the shuttling is in horizontal (from a red to a blue column or the

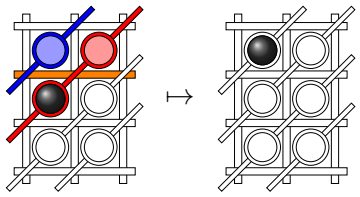
other way around) or vertical direction (inside the same column). However, the shuttling in between columns results in a Z-rotation, that must be compensated by timing operations correctly, see [18] for details. This Z-rotation can also by used as a local single qubit gate, see Section 4.3.3.3. The operation is dependent on the BOARDSTATE by the prerequisite that the dot adjacent to the qubit to must be empty. Collisions of qubits are to be avoided, as those could lead to the formation of different charge states (see however the measurement process in Section 4.3.3.2). We now describe the coherent shuttling as the combination of grid operations.

We lower the vertical (or horizontal) barrier in between the two dots and instigate a 'gradient' of the on-site potentials of the two dots. That is, the diagonal line of the dot containing the qubit must be operated at $t \in [0:T]$ while the line overhead the empty dot must have the potential $\hat{t} \in [0:T]$ with $\hat{t} = t - 1$. Note that this implies it might not be operated at all (if it is already at the right level). We will subsequently refer to the combination of a lowered barrier and such a gradient as a "flow". A flow will in general be into one of the four directions on the grid. We define the commands $\mathrm{VS}[i,j,k]$ (vertical shuttling) and $\mathrm{HS}[i,j,k]$ (horizontal shuttling). The command $\mathrm{VS}[i,j,k]$ shuttles a qubit at location (i,j) to $(i+1,j)$ for $k=1$ (upward flow) and shuttles a qubit at location $(i+1,j)$ to (i,j) for $k=(-1)$ (downward flow). Similarly, the command $\mathrm{HS}[i,j,k]$ shuttles a qubit at location (i,j) to $(i,j+1)$ for $k=1$ (rightward flow) and shuttles a qubit at location $(i,j+1)$ to (i,j) for $k=(-1)$ (leftward flow). See Table 4.2 for a summary of these OPCODES.

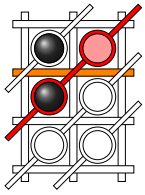
Using only these control lines, we can individually select a single qubit to be shuttled. However, when attempting to shuttle in a parallel manner, we have to be carefully take into account the effect that the activation of several of those lines has on other locations. We will deal with this in more detail in Section 4.4.1.

4.3.3.2 Measurement and readout

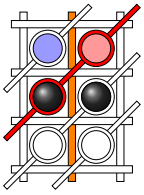
The QDP allows for local single qubit measurements in the computational basis $|0\rangle, |1\rangle$. We can measure a qubit by using essentially the same lines as if we were to shuttle it to a horizontally adjacent dot that is already occupied by a qubit in a fixed state of reference: that qubit will



(a) Coherent shuttling



(b) $\sqrt{\text{SWAP}}$



(c) CPHASE* or measurement

Figure 4.2. Schematic representation of the use of control lines for the native operations in the QDP. Qubits are represented by black balls on the grid. Red or blue colored dots are empty, but their dot potentials change due to an operation of the diagonal line they are coupled to. Empty dots, unaffected by grid operations, are white. **(a)** Vertical shuttling of a qubit (to the top left dot) requires to lower the orange barrier. One can than either raise the dot potentials on the red diagonal line, or lower the potential on the blue dot by pulsing the blue diagonal. **(b)** Schematic representation of the control lines used for performing two-qubit $\sqrt{\text{SWAP}}$ gate between the two qubits on that grid. The orange barrier is lowered and the red diagonal line is utilized to detune dot potentials. **(c)** Grid operations necessary to perform a measurement or a two-qubit effective CPHASE* gate between the two qubits. The orange barrier between the two qubits is lowered, and the dot potentials along the red diagonal line is raised by pulsing the latter. Note that the empty, red colored dot is also effected by that action, and its barrier to the adjacent dot is lowered. If the two dots in the upper row were not empty, side effects would occur. See Section 4.3.3.4 for more information on the nature of the two-qubit gates. Note also that the readout procedure of the measurement requires us to have the upper dot (light blue) empty, if the barrier gate between them is used for readout.

therefore be referred to as reference qubit.[2] When the first qubit is in contact with the reference, their total spin wavefunction collapses into either a singlet or a triplet state. Due to the Pauli principle, those two spin wave functions produce an antisymmetric spatial wave function resulting in a different distribution of charge over the two dots. This charge distribution can be detected in the readout. This process is called Pauli Spin Blockade (PSB) measurement [18, 85]. However, the QDP's ability to perform this type of qubit measurements is limited by three factors.

Firstly, the measurement requires a reference qubit horizontally adjacent to the qubit to be measured. Not only must the reference be in a known computational basis state, but the choice of state depends on the magnetic field, i.e. whether the dots are in red or blue columns in Figure 4.1(a). A reference qubit in the set $\mathcal{B}$ must be in the state $|0\rangle$, whereas one found in $\mathcal{R}$ must be in $|1\rangle$. The qubit that is to be measured, has to be in the respective other column, vertical measurements are not allowed. This effectively means that when a qubit pair is in the wrong configuration we must first shuttle both qubits one step to the left (or to the right). Note that this takes two additional shuttling operations, which means it is important to keep track at all times where the two qubits are on the BOARDSTATE, or else incur a shuttling overhead (which might become significant when dealing with large systems and many simultaneous measurements). We will deal with the problem of qubit-pair placement in more detail in Section 4.4.3.

Secondly, assuming that the qubit pair is in the right configuration to perform the PSB process, one still needs to perform a shuttling-like operation to actually perform the measurement. On the technical level, the operation is different from coherent shuttling, but the use of the lines is similar with the difference that after the readout, the shuttling-like operation is undone by the use of the same lines as before - which are not necessarily the lines one would use to reverse a coherent shuttling operation. However, scheduling measurement events on the QDP is at least as hard as the scheduling of shuttle operations discussed above. Depending on the state the qubit is in, it will now assume one of two possible states that can be distinguished by their charge distribution.

[2] Not to be confused with the measurement qubit in the surface code. The role of the latter is assumed by the first qubit, that is supposed to be measured.

OPCODE	Control OPCODES	Effect
$\mathrm{HS}[i,j,k]$	$\mathrm{V}[i]$ & $\mathrm{D}[i-j][t-1/2-k/2]$ & $\mathrm{D}[i-j+1]$[t-1/2+k/2]	$(k=1)$: Shuttle from (i,j) to $(i,j+1)$ $(k=-1)$: Shuttle from $(i,j+1)$ to (i,j)
$\mathrm{VS}[i,j,k]$	$\mathrm{H}[j]$ & $\mathrm{D}[i-j][t-1/2-k/2]$ & $\mathrm{D}[i-j-1][t-1/2+k/2]$	$(k=1)$: Shuttle from (i,j) to $(i+1,j)$ $(k=-1)$: Shuttle from $(i+1,j)$ to (i,j)
$\mathrm{M}[i,j,k]$	$\mathrm{HS}[i,j+1/2+k/2,-k]$	Measurement of qubit (i,j) using the qubit at $(i,\ j+k)$

Table 4.2. OPCODES for horizontal and vertical shuttling and measurement together with the control OPCODES required to implement these operations on the QDP.

Thirdly, the readout process requires to have a barrier line that borders to the qubit pair, with an empty dot is across the spot of the qubit to be measured. This is a consequence of the readout procedure [18].

In Table 4.2 we introduce the measurement OPCODE $\mathrm{M}[i,j,k]$ with $k \in \{-1,1\}$ to denote a measurement of a qubit at location (i,j) with a reference located to the left $(k=-1)$ or to the right $(k=1)$.

4.3.3.3 Single-qubit rotations

There are two ways in which single qubit rotations can be performed on the QDP, both with drawbacks and advantages. The first method, which we call the semi-global qubit rotation, relies on electron-spin-resonance [87]. Its implementation in the QDP allows for any rotation in the single qubit special unitary group $SU(2)$ [88] to be performed but we do not have parallel control of individual qubits. The control architecture of the QDP is such that we can merely apply the same single qubit unitary rotation on all qubits in either $\mathcal{R}$ or $\mathcal{B}$ (even or odd numbered columns). Concretely we can perform in parallel the single qubit unitaries

$$U_{\mathcal{R}} \;=\; \bigotimes_{(i,j)\in\mathcal{R}} U_{(i,j)}\,, \qquad\qquad U_{\mathcal{B}} \;=\; \bigotimes_{(i,j)\in\mathcal{B}} U_{(i,j)} \tag{4.1}$$

where $U_{(i,j)}$ means applying the same unitary U to the state carried by the qubit at location (i,j). In general the only way to apply an arbitrary

single qubit unitary on a single qubit in $\mathcal{B}$ (or $\mathcal{R}$) is by applying the unitary to all qubits in $\mathcal{B}$ ($\mathcal{R}$), moving the desired qubit into an adjacent column, i.e. from $\mathcal{B}$ to $\mathcal{R}$ ($\mathcal{R}$ to $\mathcal{B}$) and then applying the inverse of the target unitary to $\mathcal{R}$ ($\mathcal{B}$). This restores all qubits except for the target qubit to their original states and leaves the target qubit with the required unitary applied. The target qubit can then be shuttled to its original location. A graphical depiction of the BOARDSTATE associated with this maneuver can be found in Figure 4.3. This means applying a single unitary to a single qubit takes a constant amount of grid operations regardless of grid size.

The second method does allow for individual Z-rotations on single qubits: $\exp(i\phi Z) = \cos\phi \cdot \mathbb{I} + i \sin\phi \cdot Z$. This operation can be performed on a given qubit at (i, j) by shuttling it to an empty dot at $(i, j \pm 1)$ (and perhaps back). When the qubit leaves the column it was originally defined on ($\mathcal{B}$ to $\mathcal{R}$ or vice versa) it will effectively start precessing about its Z-axis [18]. This effect is always present but it can be mitigated by timing subsequent operations such that a full rotation happens between every operation (effectively performing the identity transformation, see Section 4.3.3.1). By changing the timing between subsequent operations any rotation angle ϕ can be effected. This technique will often be used to perform the Z-gate ($\phi = \pi/2$) and the $S = \sqrt{Z}$ phase gate ($\phi = \pi/4$) in error correction sequences.

4.3.3.4 Two-qubit gates

As the last elementary tool, we have the ability to apply entangling two-qubit gates on adjacent qubits. The QDP can perform two different types of two-qubit gates. Inside one column, so between qubits at locations (i, j) and $(i \pm 1, j)$, a square-root of SWAP ($\sqrt{\text{SWAP}}$) can be realized [89]. This can be done by lowering the horizontal barrier between the two qubits and toggling the voltage on the diagonal lines overhead the two qubits. This situation is illustrated in Figure 4.2 (c). The $\sqrt{\text{SWAP}}$ gate is defined as

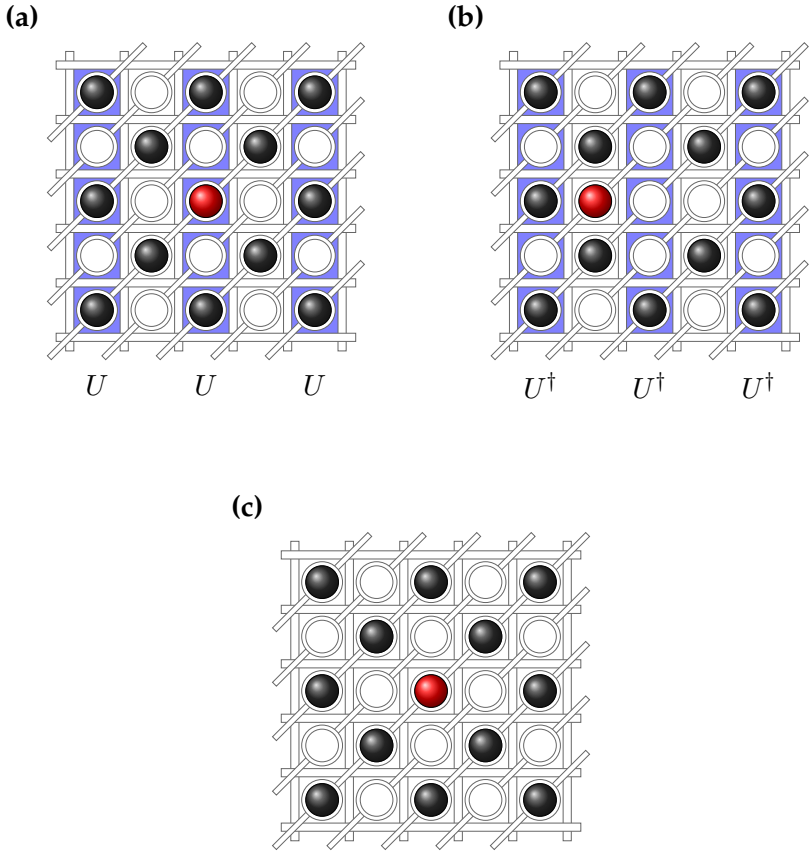


Figure 4.3. BOARDSTATE schematic for applying the unitary U to a single qubit (red). Time flows from (a) to (c) in this schematic. This process illustrates both, the possibility to retain single qubit control by using coherent shuttling, and the overhead that comes with it. In **(a)** we firstly apply the unitary U (blue bars) to all qubits in $\mathcal{R}$ ($\mathcal{B}$). We then move the qubit to the adjacent column. Note that this takes two operations because we do not want any other qubits transitioning with it. In **(b)**, we apply the inverse unitary $U^\dagger$ to all qubits in $\mathcal{R}$ ($\mathcal{B}$). In the last step we move the red qubit back, such that it is in its original position in **(c)**.

$$\sqrt{\text{SWAP}} = \begin{pmatrix} 1 & & & \\ & (1+i)/2 & (1-i)/2 & \\ & (1-i)/2 & (1+i)/2 & \\ & & & 1 \end{pmatrix}, \tag{4.2}$$

in the computational basis, and has the name-lending property $\sqrt{\text{SWAP}} \cdot \sqrt{\text{SWAP}} = \text{SWAP}$. Alternatively, between horizontally adjacent qubits, e.g. between $(i, j) \in \mathcal{R}$ and $(i, j \pm 1) \in \mathcal{B}$ the native two-qubit gate is an effective CPHASE gate which acts on the computational basis as

$$\text{CPHASE}^{\star} = \begin{pmatrix} 1 & & & \\ & e^{i\phi_1} & & \\ & & e^{i\phi_2} & \\ & & & 1 \end{pmatrix}, \tag{4.3}$$

where the two angles obey $(\phi_1 + \phi_2 \bmod 2\pi) = \pi$ (demonstrated in [90–92]). This gate can be performed between horizontally adjacent qubits by lowering the vertical barrier between them and toggling the overhead diagonal lines. This is illustrated in Figure 4.2(a). The CPHASE* can be corrected to a CPHASE by the readily available methods of performing individual Z-rotations. In practice, however, we expect the $\sqrt{\text{SWAP}}$ gate to have significantly higher fidelity than the CPHASE* gate [18], so in any application (e.g. error correction) the $\sqrt{\text{SWAP}}$ gate is the preferred native two-qubit gate on the QDP. In Table 4.3 we define OPCODES for the horizontal interaction (CPHASE*) and the vertical interaction ($\sqrt{\text{SWAP}}$).

4.3.3.5 CNOT subroutine

Many quantum algorithms are conceived using the CNOT gate as the main two-qubit gate. However the QDP does not support the CNOT gate natively. It is easy to construct the CNOT gate from the CPHASE* gate by dressing the CPHASE gate with single qubit Hadamard rotations as seen in Figure 4.4(center). It is slightly more complicated to construct a CNOT gate using the $\sqrt{\text{SWAP}}$ but it can be done by performing two $\sqrt{\text{SWAP}}$ gates interspersed single qubit rotations [91–93] as seen in Figure 4.4(right). If the control qubit is moved from an adjacent column on the QDP (as it is in most cases we will deal with) the Z- and S-gates can be performed by the Z-rotation-by-waiting technique described in

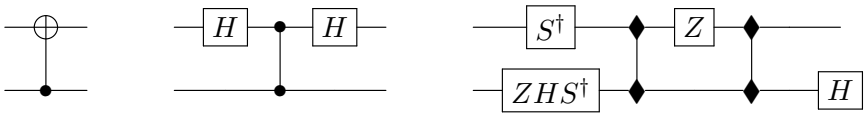


Figure 4.4. Construction of the CNOT gate out of the native CPHASE* and $\sqrt{\text{SWAP}}$ gates. Note that one requires two $\sqrt{\text{SWAP}}$ gates to construct a CNOT gate [93]. When performing arbitrary algorithms it would be preferable to forgo this substitution and instead compile the algorithm directly into a gate-set containing the $\sqrt{\text{SWAP}}$ gate.

OPCODE	Effect	Parameter
HI$[i,j]$	Perform CPHASE* gate between dots (i,j) and $(i,j+1)$	$(i,j) \in [0:N-2]^{\otimes 2}$
VI$[i,j]$	Perform $\sqrt{\text{SWAP}}$ gate between dots (i,j) and $(i+1,j)$	$(i,j) \in [0:N-2]^{\otimes 2}$
HC$[i,j]$	Perform CNOT (using CPHASE*) between (i,j) and $(i,j+1)$	$(i,j) \in [0:N-2]^{\otimes 2}$
VC$[i,j]$	Perform CNOT (using $\sqrt{\text{SWAP}}$) between (i,j) and $(i+1,j)$	$(i,j) \in [0:N-2]^{\otimes 2}$

Table 4.3. OPCODES for horizontal and vertical two-qubit operations on the QDP, respectively the CPHASE* and $\sqrt{\text{SWAP}}$ gates. We also include OPCODES for the performing of CNOT gates composed of $\sqrt{\text{SWAP}}$ or CPHASE* gates.

the last section. For completeness we also define an OPCODE for the CNOT operation in Table 4.3.

4.4 Parallel operation of a crossbar architecture

In this section we focus on performing operations in parallel on the QDP (or more general crossbar architectures). Because of the limitations imposed by the shared control lines of the crossbar architecture, achieving as much parallelism as possible is a nontrivial task. We will discuss parallel shuttle operations, parallel two qubit gates, parallel single qubit gates and parallel measurement. As part of the focus on parallel shuttling we also include some special cases relevant to quantum error correction where full parallelism is possible.

Before we start our investigation however, we would like to put three issues into focus that are likely to be encountered when attempting parallel operations. Firstly, it must be understood that an operation on one location on a crossbar system can cause unwanted side effects in other locations (that might be far away). As indicated in Section 4.3 many elementary operations on the grid in particular take place at the crossing points of control lines. This means that any parallel use of these grid operations must take into account "spurious crossings" which may have such unintended side effects. Let us illustrate such a spurious crossing with an example. Imagine we want to perform the vertical shuttling operations $\mathrm{VS}[i, j-1, 1]$ and $\mathrm{VS}[i+2, j-1, 1]$ in parallel (see Figure 4.5 for illustration). We can do this by lowering the horizontal barriers at rows i and $i+2$ (orange in illustration) and elevating the on-site potentials on the diagonal lines $i-j+1$ and $i+2-j+1$ (red in illustration). This will open upwards flows at locations $(i, j-1)$ and $(i+2, j-1)$. However it will also open an upward flow at the location $(i+2, j+1)$. This means, if a qubit is present at that location an unintended shuttling event will happen. To avoid this outcome we must either perform the operations $\mathrm{VS}[i, j-1, 1]$ and $\mathrm{VS}[i+2, j-1, 1]$ in sequence (taking two time-steps) or perform an operation $\mathrm{VS}[i+2, j+1, -1]$ to fix the mistake we made, again taking two time-steps. This is a general problem when considering parallel operations on the QDP.

Secondly, we would like to point out that in realistic setups, we expect a trade-off between parallelism (manifested in algorithmic depth) and operation fidelity (in particular this will be the case in the QDP system). In order to understand this, we have to be aware that most operations consist of applying the correct pulses for the right amount of time. Due to g-factor variations, these durations can slightly vary from dot to dot. In order to perform the perfect gate, for instance, we must be able to terminate one interaction in a parallel operation. This usually entails being able to eliminate a single crossing by resetting one control line prematurely, if the dot at the crossing has a higher g-factor. If this is not possible (maybe because it would cause side effects) a loss in operation fidelity is a consequence of the resulting improperly timed operation. The most robust case is thus to schedule operations line-by-line. By this we mean that we attempt to perform $O(N)$ grid operations in a single time step while using every horizontal, diagonal or vertical line only once per individual

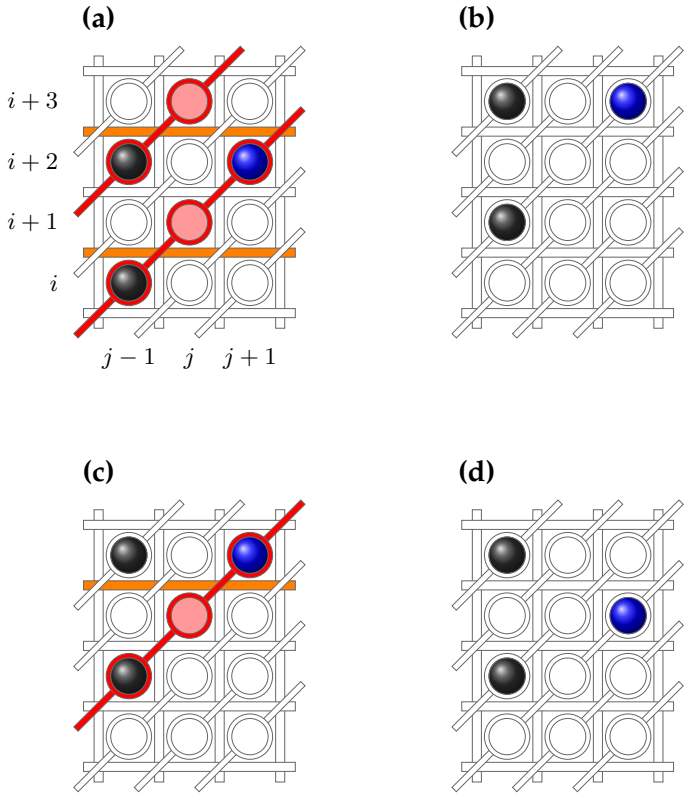


Figure 4.5. Spurious shuttle operations. Here we illustrate an example of unintended side effects that occur due to the limited control. We again denote qubits by colored balls, and color barriers and lines that are operated. Empty dots with changed potentials are colored as well, whereas white dots are unaffected. **(a)** The black qubits are to be shuttled from $(i, j-1)$ to $(i+1, j-1)$ and from $(i+2, j)$ to $(i+3, j)$ respectively without moving the blue qubit. For that purpose, the (orange) barriers between the two dot pairs are lowered, as well as the (red) diagonal lines through $(i, j-1)$ and $(i+2, j)$ are pulsed, such that the dot potentials at those sites are raised. **(b)** The qubit on $(i+3, j+1)$ has unintentionally moved to $(i+2, j+1)$. **(c)** To remedy this situation, we lower the barrier labeled $i+2$ again (orange), and also raise the potential at dot $(i+3, j+1)$ and with it at all other dots that are connected by the pulsed diagonal line (red). In **(d)**, the desired situation is achieved.

grid operation. If we, for instance, schedule several vertical shuttle operations, we may choose to start by lowering the horizontal barrier first and then detune the dot potentials of all qubits adjacent to that barrier, by pulsing the corresponding diagonal lines. To account for the variations, we reset the diagonal lines at slightly different times. Line-by-line operations work with either line types for every two-dot operation (measurement, shuttling and two-qubit gates). Note however that for shuttling operations individual control over one line is sufficient, whereas for measurement and two-qubit gates we would ideally like to be able to control two lines per qubit pair individually, where one line should be the barrier separating the two paired qubits. Results presented in the following take these constraints into account for quantum error correction. The parallel operation nonetheless remains one of the greatest challenges of the crossbar scheme. In this section, we will assume all operations to be perfect (even when performed in parallel) but in Section 4.6 we perform a more detailed analysis of the behavior of the QDP when operational errors are taken into account.

Thirdly, it is important to have access to classical side computations to aid the scheduling of parallel operations without spurious crossings. However, no classical assistance is required for purposes of quantum error correction, such that a discussion of the concrete algorithms is omitted. The interested reader may find an in-depth discussion on the classical side computations within the original work [94] or the crossbar chapter of [95]. As we define parallel versions of the elementary operations in the next step, we would like the reader to bear in mind that these OPCODES work with the classical input, which in our case is however trivial. We begin with discussing parallel shuttle operations.

4.4.1 Parallel shuttle operations

We define parallel versions of the shuttling OPCODES $\mathrm{HS}[i,j,k]$ and $\mathrm{VS}[i,j,k]$ in the following table.

OPCODE	Effect
HS[L]	Perform $\mathrm{HS}[i,j,k]$ for all $(i,j,k) \in \mathrm{L}$
VS[L]	Perform $\mathrm{HS}[i,j,k]$ for all $(i,j,k) \in \mathrm{L}$

This OPCODE takes in a set (denoted as L) of tuples (i,j,k) which de-

note 'locations at which shuttling happens' (i, j) and 'shuttling direction' (k). From these codes it is not immediately clear how many of the shuttling operations can be performed in a single grid operation, i.e. setting the diagonal lines to some configuration and lowering several horizontal or vertical barrier. If multiple grid operations are needed (such as in the example Figure 4.5) we would like this sequence of grid operations to be as short as possible. However, given some initial BOARDSTATE and a parallel shuttling command HS[L] it is not clear what the sequence of parallel shuttling operations actualizing this command is. At the same time, parallelization might not be the ultimate goal, and so other schedules might be implicit in the given OPCODES.

4.4.1.1 Selective parallel single-qubit rotations

In this section we will discuss a particular example that illustrates the use of abstracting away the complexity of parallel shuttling. Imagine a QDP grid initialized in the so called *idle* configuration. This configuration can be seen in Figure 4.6. We will focus on the qubit in the odd columns (i.e. the set $\mathcal{B}$). Imagine a subset S of these qubits to be in the state $|1\rangle$ and the remainder of these qubits to be in the state $|0\rangle$. The qubits on in the set $\mathcal{R}$ can be in some arbitrary (and possibly entangled) multi-qubit state $|\Psi\rangle$. We would like to change the states of the qubits in the set S to $|0\rangle$ without changing the state of any other qubit. Due to the limited single qubit gates (see Section 4.3.3.3) available in the QDP this is a nontrivial problem for some arbitrary set S. However using the power of parallel shuttling we can perform this task as follows. Begin by defining the set of coordinates $\hat{S}$, which hold all qubits in the complement of S in $\mathcal{R}$. Now we begin by performing the parallel shuttling operation

$$\text{HS[L]}, \quad \text{L} = \{(i, j, 1) \ \| \ (i, j) \in \hat{S}\}. \tag{4.4}$$

This operation in effect moves all qubits in $\hat{S}$ out of $\mathcal{R}$ (and into $\mathcal{B}$, note that the dots the qubits are being shuttled in are always empty by the definition of the idle configuration). Now we can use a semi-global single qubit rotation (as discussed in Section 4.3.3.3) to perform X-rotations on all qubits in $\mathcal{R}$, which is at this point all qubits in the set S. These flips change the states of the qubits in S from $|1\rangle$ to $|0\rangle$ without changing the state of any other qubit. Following this we can restore the BOARDSTATE to its original configuration by applying the parallel shuttling command

$$\text{HS[L]}, \quad \text{L} = \{(i, j, -1) \ \| \ (i, j) \in \hat{S}\}. \tag{4.5}$$

OPCODE	Effect
HI[L]	Perform VI$[i, j]$ for $(i, j) \in$ L
VI[L]	Perform HI$[i, j]$ for $(i, j) \in$ L

OPCODE	Effect
VC[L]	Perform VC$[i, j]$ for every (i, j) in L

Now we have applied the required operation. Note that at no point we had to reason about the structure of the set S itself. This complexity was taken care of by the classical subroutines embedded in HS[L]. Next we discuss performing parallel two-qubit gates.

4.4.2 Parallel two-qubit gates

Similar to parallel shuttling it is in general rather involved to perform parallel two-qubit operations in the QDP. We can again define parallel versions of the OPCODES for two-qubit operations and then analyze how to perform them as parallel as possible (again having access to classical side computation).

However, as mentioned before, the parallel operation of two-qubit gates in the QDP will mean taking a hit in operation fidelity vis-à-vis the more controllable line-by-line operation [18]. Since this operation fidelity is typically a much larger error source than the waiting-time-induced decoherence stemming from line-by line operation we will for the remainder of this chapter assume line-by-line operation of the two-qubit gates. This will have an impact when performing quantum error correction on the QDP which we will discuss in more detail in Section 4.6.

For the sake of completeness we also define a parallel version of the CNOT OPCODE. The same considerations of parallel operation hold for the parallel use of CNOT gates as they hold for the CPHASE* and $\sqrt{\text{SWAP}}$ gates. We continue the discussion of parallelism in the QDP by analyzing parallel measurements.

4.4.3 Parallel Measurements

Performing measurements on an arbitrary subset of qubits on the QDP is in general quite involved. Every qubit to be measured requires a ref-

OPCODE	Effect
M[L]	Perform M$[i,j,k]$ for every (i,j,k) in L

erence in a known computational basis state, and an empty dot must be adjacent as a reference for the readout process. The qubits must then be shuttled such that the pairs are horizontally adjacent and located in such a way such that they are in the right columns for the PSB process to take place (revisit Section 4.3.3.2 for more information). On top of the required shuttling the PSB process itself (from a control perspective similar to shuttling) must be performed in a way that depends on the BOARDSTATE and the configuration of the reference qubits. In general this PSB process will be performed line-by-line (for the fidelity reasons mentioned in the beginning of the section) and hence requires a sequence of depth $O(N)$ parallel grid operations (plus the amount of shuttling operations needed to attain the right measurement configuration in the first place). Due to this complexity we will not analyze parallel measurement in detail but rather focus on a particular case relevant to the mapping of the surface code. But first we define a parallel measurement OPCODE M[L] which takes in a list of tuples (i,j,k) denoting locations of qubits to be measured (i,j) and whether the reference qubit is to its left $(k=-1)$ or to its right $(k=1)$.

4.4.3.1 A specific parallel measurement example

Let us consider a specific example of a parallel measurement procedure that will be used in our discussion of error correction. We begin by imagining the BOARDSTATE to be in the *idle* configuration (Figure 4.6 top left). We next perform the shuttle operations needed to change the BOARDSTATE to the *measurement* configuration. This configuration (and how to reach it by shuttling operations from the idle configuration) will be discussed Section 4.4.4 and can be seen in Section 4.6 (c). Next take the qubits to be measured in the parallel measurement operation to be the red qubits in Figure 4.6. The qubits directly to the right or to the left will serve as a reference (blue in Figure 4.6). We will assume that the reference qubits are in the $|0\rangle$ state. If some of them were in the $|1\rangle$ state instead we would perform the procedure given in Section 4.3.3.3 to rotate them to $|0\rangle$ without changing the state of the other qubits on the grid. With that

all the reference qubits are in the set $\mathcal{B}$ and qubits to be read out in the set $\mathcal{R}$, we can perform the PSB process by sending the latter into the dots occupied by the former. Using the shorthand $a \overset{b}{=} c$ to denote $a \bmod b = c$, we employ the commands

$$\text{VS[L]}, \qquad \text{L} = \{(i, j, 1) \parallel i \overset{2}{=} 0,\ j \overset{2}{=} 1,\ i + j \overset{4}{=} 1\} \tag{4.6}$$

to bring the qubits to be measured (red) horizontally adjacent to the reference qubits (blue) and then

$$\text{M[L]}, \qquad \text{L} = \{(i, j, 1) \parallel i \overset{4}{=} 1,\ j \overset{4}{=} 1\} \tag{4.7}$$

and

$$\text{M[L]}, \qquad \text{L} = \{(i, j, -1) \parallel i \overset{4}{=} 3,\ j \overset{4}{=} 3\}. \tag{4.8}$$

All of these operations can be performed in a single time-step, although the line-by-line manner is preferred by reasons laid out earlier. In particular we would like to perform these operations one row at a time since this gives us the ability to control both diagonal and vertical lines individually for each measurement. However, if we first were to align all pairs, a line-by-line measurement is not possible. For instance when performing measurements on the qubits at locations $(1, 1)$ and $(1, 5)$ we must measurement is also invoked on the pair at location $(5, 5)$. To avoid this situation we will align only the qubits in the bottom row, perform the PSB process and readout on that row only and then undo the shuttlings. This we repeat going up in rows until we reach the end of the grid. More formally we perform the following sequence of operations:

For $i \in [0 : N - 2]$

If $i \overset{4}{=} 1$

$$\text{VS[L]}, \qquad \text{L} = \{(i - 1, j, -1) \parallel j \overset{4}{=} 1\}$$

$$\text{M[L]}, \qquad \text{L} = \{(i, j, 1) \parallel j \overset{4}{=} 1\}$$

$$\text{VS[L]}, \qquad \text{L} = \{(i - 1, j, 1) \parallel j \overset{4}{=} 1\}$$

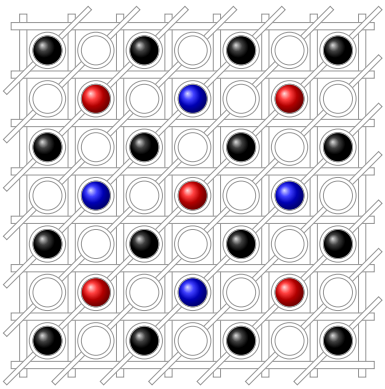
If $i \overset{4}{=} 3$

$$\text{VS[L]}, \qquad \text{L} = \{(i-1, j, -1) \parallel j \overset{4}{=} 3\}$$

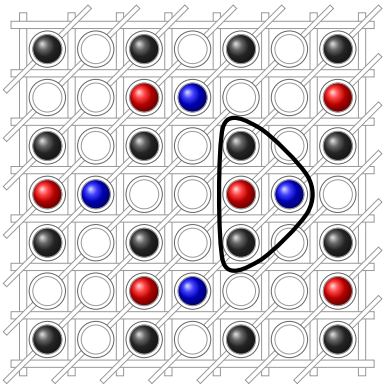
$$\text{M[L]}, \qquad \text{L} = \{(i, j, -1) \parallel j \overset{4}{=} 3\}$$

$$\text{VS[L]}, \qquad \text{L} = \{(i-1, j, 1) \parallel j \overset{4}{=} 3\}\,.$$

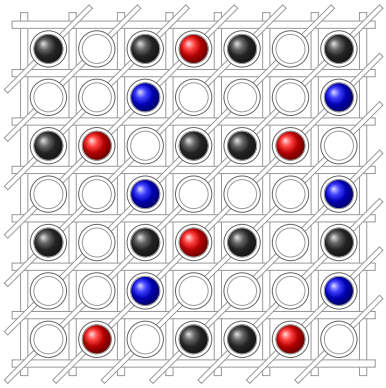
We will use this particular procedure when performing the readout step in a surface code error correction cycle in Section 4.5.3. This concludes our discussion of parallel operations on the QDP. We now move on to highlight some BOARDSTATE configurations that will feature prominently in the surface and color code mappings.



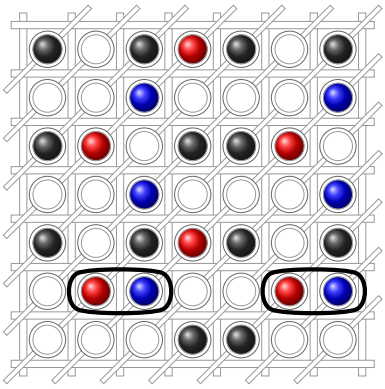
(a) Idle



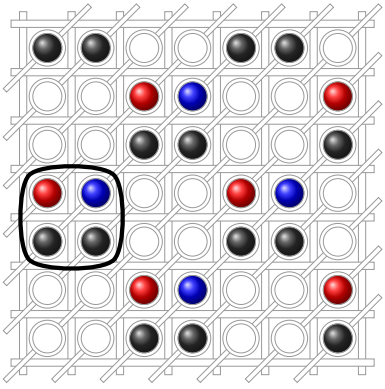
(b) Rightward triangle



(c) Measurement



(d) PSB and readout



(e) Right square

Figure 4.6. Useful BOARDSTATE configurations. We denote memory qubits with dark color, X-measurement qubits by red and Z-measurement qubits by blue. Those will collect the parity of the data qubits in one error correction cycle, and one is the others reference at the PSB measurement. **(a)** The idle configuration is a starting point of all algorithms. All qubits are spread out and well separated. **(b)** The triangle configurations (here we have a rightward triangle, see the frame in the figure) is assumed when the proximity of measurement qubits to data qubits is required. This is the case for the parity measurements in error correction cycles. **(c)** The measurement configuration is formed to bring X- and Z-measurement qubits close to each other, such that a row can be selected in which the measurement is performed. **(d)** Certain measurement qubits are brought to adjacent dots in order to perform the PSB-based measurement and readout in a line-by-line fashion (encircled qubits). Since the rest of the grid is in the measurement configuration, individual control over the barrier lines and one potential is guaranteed without spurious measurements. **(e)** The (right) square configuration is a mid-way point between the idle and (right) triangle configuration. Going through the square configuration keeps the shuttling algorithm manageable, as not more that 2 different heights of the dot potentials are employed. One of the characteristic squares is framed in the figure.

4.4.4 Some useful grid configurations

There are several configurations of the BOARDSTATE that show up frequently enough (for instance in the error correction codes in Section 4.5.3) to merit some special attention. In this section, we list these specific configurations and show how to construct them. Note that this is done using Figure 4.6, in which the red (blue) qubits will later serve as measurement qubits for the $Z-$type ($X-$type) stabilizer tiles of the surface code, while the dark qubits are part of the memory.

4.4.4.1 Idle configuration

The idle configuration is the configuration in which the QDP is initialized. As shown in Figure 4.6, its BOARDSTATE matrix describes a checkerboard pattern. In this configuration no two-qubit gates can be applied between any qubit pair but since it minimizes unwanted crosstalk between qubits [18], it is good practice to bring the system back to this configuration when not performing any operations. For this reason we consider the idle configuration to be the starting point for the construction of all other configurations.

4.4.4.2 Square configuration

As seen in Figure 4.6(e), the square configurations consist of alternating filled and unfilled 2×2 blocks of dots. The so-called right square configuration can be reached from the idle configuration by a shuttling operation $\mathrm{HS}[\mathrm{L}]$ with the set L being

$$\begin{aligned}\mathrm{L} \;&= \{(i,j,1) \parallel i \overset{2}{=} 1,\; j \overset{2}{=} 1,\; i+j \overset{4}{=} 2\} \\ &\cup\; \{(i,j,-1) \parallel i \overset{2}{=} 0,\; j \overset{2}{=} 1,\; i+j \overset{4}{=} 3\}.\end{aligned} \tag{4.9}$$

Note that this operation only takes a single time-step, and the square configuration is shown in Figure 4.6(e). The right square configuration is characterized by Z-measurement qubits being in the left corner of every square. Another flavor of this configuration is the left square configuration, where the Z-measurement qubits are in the upper right corner, and the X-measurement qubits in the left in the left. The left square configuration can be reached from the idle configuration by a shuttling operation $\mathrm{HS}[\mathrm{L}]$ with the set L being

$$\begin{aligned}\mathrm{L} \;&= \{(i,j,1) \parallel i \overset{2}{=} 0,\; j \overset{2}{=} 0,\; i+j \overset{4}{=} 2\} \\ &\cup\; \{(i,j,-1) \parallel i \overset{2}{=} 1,\; j \overset{2}{=} 0,\; i+j \overset{4}{=} 1\}.\end{aligned} \tag{4.10}$$

These configurations are used as an intermediate step for us to reach the triangle configurations.

4.4.4.3 Measurement Configuration

The measurement configuration can be reached from the idle configuration in three time-steps by the following sequence of parallel shuttling operations.

$$\begin{aligned}&\mathrm{HS[A]}, \qquad \mathrm{A} = \{(i,j,-1),\; (i-1,j-1,1) \mid i \overset{4}{=} 1,\; j \overset{4}{=} 2\}, \\ &\mathrm{HS[B]}, \qquad \mathrm{B} = \{(i-1,j-1,1) \parallel i \overset{4}{=} 3,\; j \overset{4}{=} 1\}, \\ &\mathrm{VS[C]}, \qquad \mathrm{C} = \{(i,j,-1) \parallel i \overset{2}{=} 0,\; j \overset{2}{=} 1,\; i+j \overset{4}{=} 1\}.\end{aligned} \tag{4.11}$$

This configuration can be seen in Figure 4.6(d) and it is an intermediate state in the measurement process in which the blue qubits are read out against the red ones. How this measurement protocol works in detail is described in Section 4.4.3.

4.4.4.4 Triangle configurations

In order to collect the parity of memory qubits in the error correction cycles, we need to align the measurement qubits with them, where it hinges on the two-qubit gates whether the alignment is horizontal or vertical. This is reflected in the use of triangle configurations. There are two triangle configurations that can be reached in a single parallel shuttling step from the right square configuration. The first one, seen in Figure 4.6(b), is called the rightward triangle configuration. It can be reached from the square configuration by the grid operation $\mathrm{HS[L]}$ with the set L being

$$\mathrm{L} = \{(i,j,-1) \parallel i \overset{2}{=} 1,\; j \overset{2}{=} 1,\; i+j \overset{4}{=} 3\}, \tag{4.12}$$

which does as much as to shuttle the right memory qubit of every square (framed squares in Figure 4.6(e)) to the empty dot on its right. In this configuration, we are able to perform high-fidelity two-qubit gates between

measurement and memory qubits in every triangle. In order to reach the neighboring pair of memory qubits, we start from the left square configuration horizontally shuttling the left memory qubit out of every square. Operationally, we would perform HS[L] with

$$\mathrm{L} = \{(i, j, 1) \;\|\; i \overset{2}{=} 0,\ j \overset{2}{=} 0,\ i + j \overset{4}{=} 2\}. \tag{4.13}$$

Note again that these parallel shuttling operations can be performed in a single time step. From these configurations the idle configuration can also be reached in a single time step. In the next section, these configurations will feature prominently in the mapping of several quantum error correction codes to the QDP architecture.

4.5 Error correction codes

In this section, we will apply the techniques we developed in the previous sections to map topological quantum error correction codes to the QDP.

4.5.1 Surface code

The planar surface code is well-studied to the point were we have an exact idea of how it should be implemented. In its rotated version, shown in Figure 1.2, one code patch contains $2d^2 - 1$ physical qubits encoding and maintaining a single logical qubit with distance d.Here, d^2 qubits are part of the memory and additional $d^2 - 1$ qubits are used for syndrome measurements. All of them are placed onto rhombus-shaped patch of square lattice, with ears at its boundaries. In the bulk, a checkerboard tiling of stabilizers, in which each plaquette engulf 5 qubits, is found – see Figure 4.7(a). Adjacent plaquettes share 2 memory qubits each, and each tile's central qubit is used for the measurement. In an error correction cycle, it collects the parity of the tile's memory qubits with CNOT-gates [80, 81, 96, 97], see Figure 4.7(b) and (c). Whether the parity is collected in the Hadamard or computational basis, meaning whether the stabilizer on those four qubits is $Z^{\otimes 4}$ or $X^{\otimes 4}$, is dependent on the shade of the plaquette on the checkerboard. In all figures of this thesis depicting surface code, we have chosen to distinguish Z-type stabilizers with a darker shade from white X-type stabilizers. Like in the previous chapter, we assume that gate operations can be performed in parallel as long as they do

not share any resources. This leads the to a constant runtime for all parity collections. The measurement qubits, which now carry all syndrome information, are then read out. After being decoded, possible errors can be rectified and the error correction cycle concluded. Note that it is assumed that readout and correction are assumed to happen in single time steps, such that the entire circle has a runtime of $O(1)$.

Unfortunately, we cannot hope to run surface code cycles in the same manner on the QDP, not even if we neglect the issues of parallel operation and spurious crossings. As it turns out, our idea of how to run the code makes strong assumptions on the capabilities of the device that cannot be matched with the QDP: although CNOT-gates are possible to all adjacent qubits in the QDP, we have already argued to refrain from the use of CPHASE* gates for the sake of fidelity. This renders some of the two-qubit gates in Figure 4.7(b) and (c) nonlocal. Moreover, we require an additional qubit to be present in each stabilizer tile, to serve the measurement qubit as a reference in the syndrome extraction. Also, the readout procedure requires an empty dot along the barrier gate, which raises questions about the packing density of the qubits. To remedy all those issues, we present a revised version of the surface code cycles in Section 4.5.3.

4.5.2 2D color codes

Another important class of planar topological codes are the 2D color codes [83]. These codes are defined on 3-colorable tilings of the Euclidean plane. Two such tilings are featured in the so-called $6.6.6.$ and $4.8.8.$ codes, where hexagonal and square-octagonal shapes occur respectively. Similar to Figure 1.2, we can think of the memory qubits as sitting at the corners of those tiles, but the difference is here that every tile hosts two stabilizers, namely one in which X-operators are applied to their corners and another in which the same operators are replaced by Z. With suitable boundary conditions this construction encodes a single logical qubit with a distance d using an amount of d^2 physical qubits. See Figure 4.8 for examples of the $6.6.6.$ and $4.8.8.$ color codes of distance five, in which tiles and qubits are sketched. Note that, similar to Figure 1.2(b) and (c), these pictures do not include measurement qubits. Planar color codes have lower thresholds than the planar surface code but are more versatile when it comes to fault-tolerant gates, as the support the full Clifford

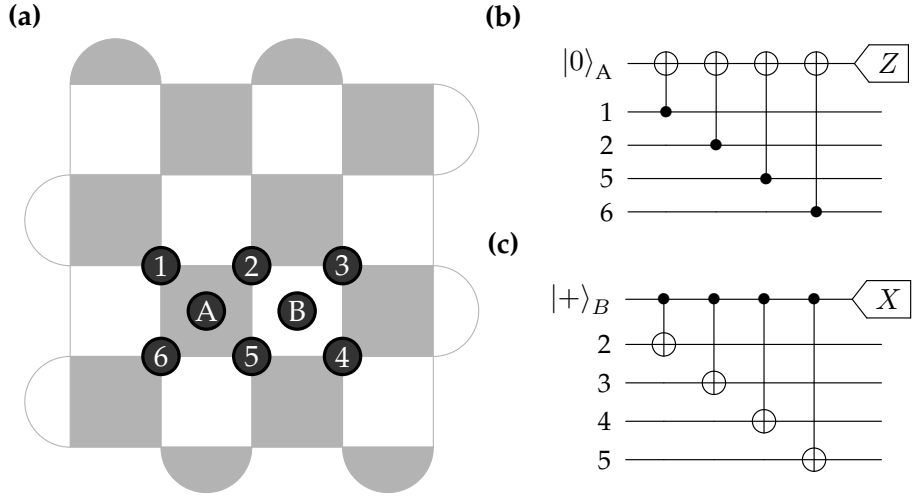


Figure 4.7. Stabilizer measurements in the surface code. **(a)** Distance-five code with some labeled qubits. Here, A and B label measurement qubits, while memory qubits carry numbers. The dark plaquettes indicate that the qubits at its corners are involved in a $X^{\otimes 4}$ stabilizer, where the syndrome is read out on the measurement qubit in its center. White plaquettes indicate regular parity measurements. **(b) & (c)** $Z-$ and X-stabilizer circuits [80, 81, 96, 97], with the qubits from panel (a).

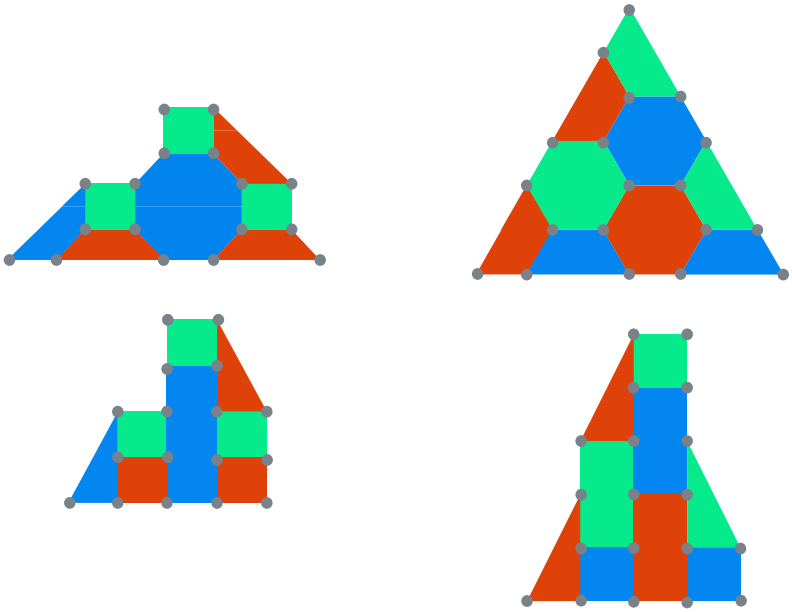


Figure 4.8. Distance 5 examples of the $4.8.8.$ (first from left) and $6.6.6.$ (third from left) color codes [83] and their deformed versions (second from left and fourth from left respectively). The vertices correspond to memory qubits and every colored face corresponds to both an X- and a Z-stabilizer to be measured. These stabilizers can be measured by using weight $4, 6$ and 8 versions of the circuits shown in Figure 4.7. The deformation of the codes does not change the code properties at all. They are a visual guide that facilitates the mapping the crossbar grid in Section 4.5.4.

group as a transversal set. In the next section we will focus on mapping these codes to the QDP using the concepts introduced in Section 4.4.

4.5.3 Surface code mapping

We now describe a protocol that maps the surface code on the architecture described in Section 4.3. The surface code layout has a straightforward mapping that places the memory qubits into even numbered columns, while X- and Z-measurement qubits can be found in the odd ones. This means we have single-qubit control over the set of all memory qubits and the set of all measurement qubits separately. We begin by changing the circuits performing the X- and Z-stabilizer measurements to work with $\sqrt{\text{SWAP}}$ rather than CNOT. We can emulate a

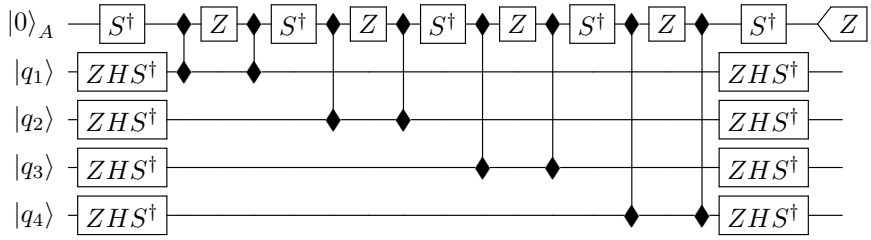


Figure 4.9. Z-stabilizer measurement circuit using the $\sqrt{\text{SWAP}}$ as the main two-qubit gate. The Z- and S-rotations can be performed by the timing procedure described in Section 4.3.3.3.

CNOT gate by using two $\sqrt{\text{SWAP}}$ gates interspersed with a Z-gate on the control plus some single qubit gates. As described in Section 4.3.3.5 the Z- and S-gates on the measurement qubit can performed by waiting, which means they can be performed locally while the single qubit operations on the memory qubits can be performed in parallel using the global unitary rotations described in Section 4.3.3.3. The X- and Z-circuits using $\sqrt{\text{SWAP}}$ are shown in Figure 4.9.

We will split up the quantum error correction cycle by first measuring all X-type stabilizers (the X-cycle) and then all Z-type stabilizers (Z-cycle). This means we can use the idle Z- (X-) measurement qubits as references for the X- (Z-) cycle measurements. For convenience we included a depiction of the surface code Z-cycle unit cell in Figure 4.10(right). Note that all panels in that figure depict the smallest possible building block of a code patch, not the patch itself. The qubit labeled 'A' is going to be measured in the Z-cycle. The numbered qubits are part of the memory and the qubit labeled 'B' is used as a reference for 'A' qubit. It is also the measurement qubit for the X-cycle. We now describe the steps needed to perform the Z-cycle in parallel on the entire surface code sheet. For convenience we ignore the surface code boundary conditions since these can be easily included. The X-cycle is equivalent up to different single qubit gates ($XS^\dagger$ instead of $ZHS^\dagger$ on the memory qubits, $HS^\dagger$ instead of $S^\dagger$ on the measurement qubits) and shifting every operation 2 steps up, e.g. setting $i \mapsto i + 2$ in row indices.

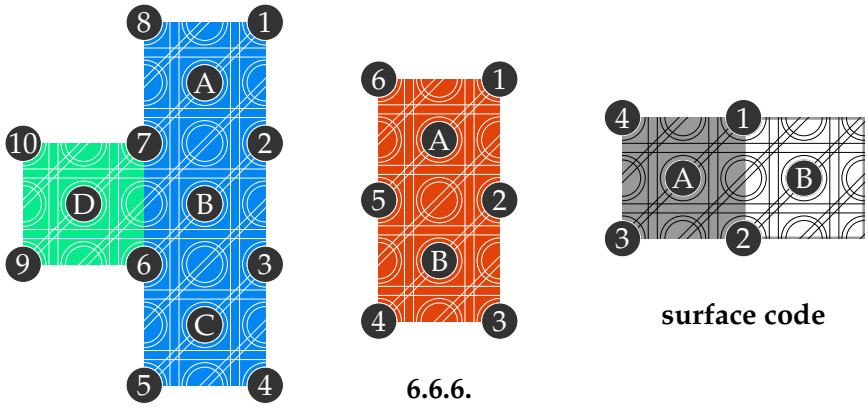


Figure 4.10. Unit cells of topological codes in the QDP. From left to right: deformed 4.8.8. color code, deformed 6.6.6. color code and surface code. Darkened circles correspond to qubits, where qubits used for measurement are labeled with letters, while memory qubits bear numbers. The shaded and colored plaquettes denote stabilizer tiles. Note that the depicted cells do not encode logical qubits, but are the smallest possible building blocks of a code patch. **4.8.8. code**: The qubit labeled 'A' is the is measured for the octagon (now a rectangle) stabilizer, while the qubit labeled 'D' has the same role for the square sub-cell. The qubit labeled 'B' is used to read out the qubit 'D' and the qubit labeled 'C' is used to read out the measurement qubit for the octagon cell directly below the square cell (not pictured). **6.6.6. code**: The qubit labeled 'A' is measurement qubit of that tile while the qubit labeled 'B' is used as a reference to read out the 'A' qubit for the unit cell directly to the bottom left (not pictured). **Surface code**: The qubit labeled 'A' is the measurement qubit of the for the Z-cycle stabilizer using 'B' as a reference. Their roles are reversed in the X-cycle.

The surface code Z-cycle

1. Initialize in the idle configuration.
2. Apply $ZHS^\dagger$ to all qubits in $\mathcal{R}$ (memory) and $S^\dagger$ to qubits in $\mathcal{B}$.
3. Go to right square configuration.
4. Go to rightward triangle configuration.
5. Perform CNOT between qubits A and 1 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \parallel i \overset{2}{=} 1,\ j \overset{2}{=} 0,\ i+j \overset{4}{=} 3\}\,.$$
6. Perform CNOT between qubits A and 2 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \parallel i \overset{2}{=} 0,\ j \overset{2}{=} 0,\ i+j \overset{4}{=} 2\}\,.$$
7. Go to idle configuration.
8. Go to left square configuration.
9. Go to leftward triangle configuration.
10. Perform CNOT between qubits A and 3 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \parallel i \overset{2}{=} 1,\ j \overset{2}{=} 0,\ i+j \overset{4}{=} 1\}\,.$$
11. Perform CNOT between qubits A and 4 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \parallel i \overset{2}{=} 0,\ j \overset{2}{=} 0,\ i+j \overset{4}{=} 0\}\,.$$
12. Go to idle configuration
13. Apply $ZHS^\dagger$ to all qubits in $\mathcal{R}$ and $S^\dagger$ to qubits in $\mathcal{B}$.
14. Apply measurement qubit correction step for qubit B as described in Section 4.4.1.1.
15. Go to measurement configuration.
16. Perform PSB measurement process as described in Section 4.4.3 using qubit B as reference to qubit A.
17. Go to idle configuration.

4.5.4 Color code mapping

The mapping of the color codes is largely analogous to that of the surface code. We begin with the $6.6.6.$ color code as it is easiest to map. First, the tiling on which the color code is must be deformed such that it is more amenable to the square grid structure of the QDP. This is fairly straightforward as can be seen from the $d = 5$ example in Figure 4.8. In the deformed tiling it is clear how to map the code to the QDP. We once again place all memory qubits in the even columns and all measurement qubits in the odd columns. This places the unit 'hexagon' seen in the deformed code into a patch of 3×5 dots on the QDP (see Figure 4.10 (right) for this unit tile). It also puts all memory qubits in $\mathcal{R}$ and 2 extra qubits into $\mathcal{B}$, both of which could be used as measurement qubit in the stabilizer circuit. We will always choose the top qubit ('A') of these two in the hexagon unit cell as the measurement qubit for the error correction cycles. The extra (bottom) qubit ('B') in the unit cell will be used as a reference for the unit hexagon to its direct left. This has the advantage of making the readout process independent of the measurement results of the previous cycles (as was the case in the surface code). Note also that all measurement qubits are positioned along diagonal lines on the QDP grid. This makes the quantum error correction cycle very analogous to the surface code. We once again must split up the X- and Z-cycles (again due to the limited single qubit rotations possible). Below we present the steps needed to perform the Z-cycle (which now measures a weight 6 operator). The X-cycle is identical up to differing single qubit rotations on the memory qubits.

The 6.6.6 color code Z-cycle

1. Apply Steps 1 to 11 in the surface code Z-cycle to perform CNOT gates between qubits A and the memory qubits $1, 2, 5, 6$ in the unit hexagon, ending in the idle configuration.
2. Go to idle configuration but with all even columns up and all odd columns down by performing $\mathrm{VS}[\mathrm{L}]$ with

$$\mathrm{L} = \{(i, j, 1) \parallel i \overset{2}{=} 0,\ j \overset{2}{=} 0\} \cup \{(i, j, -1) \parallel i \overset{2}{=} 1,\ j \overset{2}{=} 1\}\,.$$

3. Go to right square configuration.
4. Go to rightward triangle configuration.
5. Perform CNOT between qubits A and 3 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \parallel i \overset{2}{=} 1,\ j \overset{2}{=} 0,\ i+j \overset{4}{=} 1\}\,.$$
6. Go to idle configuration.
7. Go to left square configuration.
8. Go to leftward triangle configuration.
9. Perform CNOT by performing between qubits A and 4 by VC[L] with
$$\mathrm{L} = \{(i,j) \parallel i \overset{2}{=} 0,\ j \overset{2}{=} 0,\ i+j \overset{4}{=} 2\}\,.$$
10. Go to idle configuration.
11. Invert Step 6 by performing VS[L] with
$$\mathrm{L} = \{(i,j,-1) \parallel i \overset{2}{=} 0,\ j \overset{2}{=} 0\} \cup \{(i,j,1) \parallel i \overset{2}{=} 1,\ j \overset{2}{=} 1\}\,.$$
12. Apply $ZHS^\dagger$ to all qubits in $\mathcal{R}$ (memory) and $S^\dagger$ to qubits in $\mathcal{B}$.
13. Go to measurement configuration.
14. Perform PSB measurement process as described in Section 4.4.3 using qubit B as reference to A in the unit cell to the right.
15. Go to idle configuration.

Next up is the 4.8.8. color code. We deform the tiling on which the code is defined similarly to the 6.6.6. code. The deformed 4.8.8. code lattice can be seen in Figure 4.10 (left). We again place the memory qubits into $\mathcal{R}$ and the measurement qubits into the set $\mathcal{B}$. See Figure 4.10 for a layout of the unit cell of the 4.8.8. code on the QDP. Note that holds two different stabilizers. The square tile has one qubit (qubit 'D' in Figure 4.10) in $\mathcal{B}$, which we will use for the measurement. The deformed octagon tile has three qubits in $\mathcal{B}$. We will use the topmost qubit ('A') as the measurement qubit for the tile while the middle one (qubit 'B') serves

as reference for the square tile measurement directly to its left. The bottommost qubit ('C') will be used to as a reference of the octagon directly below the square tile (not pictured). Because the structure of the $4.8.8.$ code is less amenable to direct mapping the stepping process is a little more involved. We will again only write down the Z-cycle with the X-cycle being the same up to initial and final single-qubit rotations on the memory qubits.

The 4.8.8 color code Z-cycle

1. Initialize in the idle configuration.
2. Apply $ZHS^\dagger$ to all qubits in $\mathcal{R}$ (memory) and $S^\dagger$ to qubits in $\mathcal{B}$.
3. Go to right square configuration.
4. Go to rightward triangle configuration.
5. Perform CNOT between qubits A and 1 and D and 7 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \;\|\; i \overset{2}{=} 1,\ j \overset{2}{=} 0,\ \ i+j \overset{16}{=} 3 \vee 7\}\,.$$
6. Perform CNOT between qubits A and 2 as well as D and 6 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \;\|\; i \overset{2}{=} 0,\ j \overset{2}{=} 0,\ \ i+j \overset{16}{=} 2 \vee 6\}\,.$$
7. Go to left square configuration.
8. Go to left triangle configuration.
9. Perform CNOT between qubits A and 8 and D and 9 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \;\|\; i \overset{2}{=} 1,\ j \overset{2}{=} 0,\ \ i+j \overset{16}{=} 1 \vee 5\}\,.$$
10. Perform CNOT between qubits A and 7 and d and 10 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \;\|\; i \overset{2}{=} 0,\ j \overset{2}{=} 0,\ \ i+j \overset{16}{=} 0 \vee 4\}\,.$$

11. Go to idle configuration.
12. Go to idle configuration but with all even columns up and all odd columns down by performing VS[L] with
$$\mathrm{L} = \{(i,j,1) \parallel i \overset{2}{=} 0,\ j \overset{2}{=} 0\} \cup \{(i,j,-1) \parallel i \overset{2}{=} 1,\ j \overset{2}{=} 1\}\,.$$
13. Go to right square configuration.
14. Go to rightward triangle configuration.
15. Perform CNOT between qubits A and 3 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \parallel i \overset{2}{=} 1,\ j \overset{2}{=} 0,\ i+j \overset{16}{=} 3\}\,.$$
16. Perform CNOT between qubits A and 4 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \parallel i \overset{2}{=} 0,\ j \overset{2}{=} 0,\ i+j \overset{16}{=} 2\}\,.$$
17. Go to idle configuration.
18. Go to left square configuration.
19. Go to leftward triangle configuration.
20. Perform CNOT between qubits A and 6 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \parallel i \overset{2}{=} 1,\ j \overset{2}{=} 0,\ i+j \overset{16}{=} 1\}\,.$$
21. Perform CNOT between qubits A and 5 by performing VC[L] with
$$\mathrm{L} = \{(i,j) \parallel i \overset{2}{=} 0,\ j \overset{2}{=} 0,\ i+j \overset{16}{=} 0\}\,.$$
22. Go to idle configuration.
23. Invert Step 6 by performing VS[L] with
$$\mathrm{L} = \{(i,j,-1) \parallel i \overset{2}{=} 0,\ j \overset{2}{=} 0\} \cup \{(i,j,1) \parallel i \overset{2}{=} 1,\ j \overset{2}{=} 1\}\,.$$
24. Repeat Steps 2 - 23 but shifting $i \mapsto i+2$ and $j \mapsto j+1$.
25. Apply $ZHS^\dagger$ to all qubits in $\mathcal{R}$ and $S^\dagger$ to qubits in $\mathcal{B}$.
26. Go to measurement configuration.
27. Perform PSB measurement process as described in Section 4.4.3 using qubit B (unit cell to the right) as reference for qubit A and using qubit C as reference for qubit D.
28. Go to idle configuration.

4.6 Discussion

In this section, we evaluate the mapping of the error correction codes described above and argue numerically that it is possible to attain the error suppression needed for practical universal quantum computing. We will do this exercise for the planar surface code, as it is the most popular and best understood error correction code. The description given in Section 4.5.3 assumes that all operations can be implemented perfectly in parallel. In practice though, for the reasons outlined in Section 4.4 many operations that can in principle be done in parallel will be done in a line-by-line fashion. Note that for surface code in an array like this, the side lengths of a quadratic grid scale linearly with the code distance as $N = 2d + 1$. This means that the time performing a surface code cycle (and thus the number of errors affecting a logical qubit) rises linearly with the code distance and hence this mapping of the surface code will not exhibit an error correction threshold. As a consequence, the error probability of the encoded qubit (the logical error probability) cannot be made arbitrarily small but rather will exhibit a minimum for some particular code distance after which it will start rising with increasing code distance. Also, the code distance characterizing the minimum will depend nontrivially on the error probability of the code qubits. This is not a very satisfactory situation from a theoretical point of view, but being pragmatic we are not so much interested in asymptotic statements but rather in whether the logical error probability can be made small enough to allow for realistic computation [97]. As a target logical error probability we choose $P_L = 10^{-20}$ as at this point the computation is essentially error free (for comparison, a modern classical processor has an error probability around 10^{-19} [84]). We will use this number as a benchmark to assess if and for what error parameters the surface code mapping in the QDP yields a "practical" logical qubit. In order to assess this we must consider in more detail the sources of error afflicting the surface code operation on the QDP. We will begin by detailing how the surface code is likely to be implemented in practice on the QDP and afterwards consider how this impacts the error behavior of the logical surface code qubit. We will distinguish two classes of error sources: operation induced errors and decoherence induced errors.

4.6.1 Practical implementation of the surface code

Here we present an mapping of the surface code based on the one presented in Section 4.5.3 but differing in the amount of time-steps used to perform certain operations. In particular, we choose to do all shuttle and two-qubit-gate operations in a line-by-line manner. This is a specific choice which we expect will work well but variations of this protocol are certainly possible. As mentioned above, this will mean that the time an error correction cycle takes will scale with the code distance. This means it is important to keep track of the time needed to perform a cycle. We will do this while describing line-by-line operation of the surface code cycle in greater detail below.

In practice, we will perform the protocol in Section 4.5.3 in the following manner. We begin by performing Step 1 and 2 for all qubits. Then we apply Steps $3-7$ but only in rows 0 and 1. Note that after performing these steps on only the first two columns we are back in the *idle* configuration. Now we repeat the previous for rows 2 and 3 and so forth until we reach the end of the grid. Having done these operations we are at the end of Step 7 (go to *idle* configuration) and the grid is the *idle* configuration. We now repeat the same process to perform Step $8-12$ of Section 4.5.3. Next we perform Step 13 which can be done globally. Hereafter we perform step 14 (measurement qubit correction) in standard line-by-line fashion. Note that even in an ideal implementation Step 14 has to be done line-by-line in the worst case. After this we perform Step 15 (go to *measurement* configuration) in a line-by line manner and similarly for Steps 16 (PSB/readout procedure) and 17 (go to *idle* configuration).

Note that in this line-by-line implementation there is a slight asymmetry between the X- and Z-cycles. In Table 4.4, we count the number of time steps that accumulate for every operation type in each program step in Section 4.5.3. We also calculate the number of time steps (per operation type) needed for the full surface code error correction cycle.

4.6.2 Decoherence induced errors

Decoherence induced errors are introduced into the computation by uncontrolled physical processes in the underlying system. The effect of these processes is called decoherence. Decoherence happens even if a

Step	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$\sqrt{\text{SWAP}}$ gates					$2d$	$2d$				$2d$	$2d$						
Z-rotations					$2d$	$2d$				$2d$	d						
Shuttlings			d	d			d	d	d			d			$5d$	$2d$	$3d$
Global rotations		1											1		1		
Measurements																d	

	Average	Total
$\sqrt{\text{SWAP}}$ gates	$8d$	$16d$
Z-rotations	$7d$	$14d$
Shuttlings	$16d$	$32d$
Global rotations	3	6
Measurements	d	$2d$

Table 4.4. Time-step count per operation type and program step for the line-by-line implementation of the surface code cycle described in Section 4.5.3. The number of time-steps is quoted in terms of the code distance d. This table does not specify the exact order in which the operations happen, see Section 4.6.1 for an explanation of the time flow. Note that the table shows the average of the time-step counts for the X- and Z-cycles. The actual count for the individual X- and Z-cycles is slightly different due to the boundary conditions of the surface code. Table cells that are left empty signify zero entries.

qubit is not being operated upon and the amount of decoherence happening during a computation scales with the time that computation takes. Therefore, to account for decoherence induced errors during the error correction cycle we need to compute how long an error correction cycle takes. Generally any operation on the QDP takes a certain amount of time denoted by τ. We distinguish again five different operations: (sw) two-qubit $\sqrt{\text{SWAP}}$ gates, (sh) qubit shuttle operations, (z) single qubit Z-gates by waiting, (gl) global single qubit operations and (m) qubit measurements. The time they take we will denote by $\tau_{sw}, \tau_{sh}, \tau_z, \tau_{gl}$ and τ_m respectively. In Table 4.4 we count the total time taken by the surface code error correction cycle using the mapping described in Sections 4.5.3 and 4.6.1. Table 4.5 summarizes the total number of time-steps for every gate type for a full surface code error correction cycle. Following that table, the total time $\tau_{\text{total}}(d)$ as a function of the code distance d is given by

$$\tau_{\text{total}}(d) = 16d\,\tau_{sw} + 32d\,\tau_{sh} + 14d\,\tau_z + 6\,\tau_{gl} + 2d\,\tau_m. \tag{4.14}$$

This total time can be connected to an error probability by invoking the mean decoherence time of the qubits in the system, the so called T_2 time [88,

Symbol	Operation	Time-steps per cycle
τ_{sw}	$\sqrt{\text{SWAP}}$ gates	$16d$
τ_{sh}	Shuttlings	$32d$
τ_z	Z rotations by waiting	$14d$
τ_{gl}	Global qubit rotations	6
τ_m	Measurements	$2d$

Table 4.5. Time steps required for one error correction cycle of surface code.

98]. We neglect the influence of T_1 in this calculation as it is typically much larger than T_2 in silicon spin qubits [18, 99]). We can find the decoherence induced error probability P_{dec} [88, Page 384] as

$$P_{dec}(d) = \frac{\tau_{\text{total}}(d)}{2T_2}. \tag{4.15}$$

Next we investigate operation induced errors. These will typically be larger than decoherence induced errors but will not scale with the distance of the code.

4.6.3 Operation induced errors

Operation induced errors are caused by imperfect application of quantum operations to the qubit states. According to the five types of operations, we will denote the probability of an error afflicting them by $P_{sw}, P_{sh}, P_z, P_{gl}$ and P_m respectively. In Table 4.6 we list the total number of operations of a given type that memory and measurement qubits participate in over the course of a surface code cycle. In Section 4.8 we give a more detailed per-step overview of the operations performed on memory and measurement qubits. For clarity we have chosen qubit 1 in Figure 4.10 (right) as a representative of the memory qubits and qubit A as a representative of the measurement qubits. Other qubits in the code might have a different ordering of operations but their counts will be the same, except for the qubits located at the boundary of the code patch for which the given counts are an upper bound. For each operation we also calculate the average number of this times it involves memory and measurement qubits. This average number will serve as our measure of operationally induced error.

	Memory qubit			Z-measurement qubit			Average
	Z-cycle	X-cycle	Total	Z-cycle	X-cycle	Total	
$\sqrt{\text{SWAP}}$ gates	4	4	8	8	0	8	8
Z-rotations	0	0	0	7	0	7	3.5
Shuttlings	2	4	6	10	4	14	10
Global rotations	2	2	4	2	3	5	4.5
Measurements	0	0	0	1	1	2	1

Table 4.6. This table lists the total number of operations per qubit type, over the course of a surface code cycle. In Section 4.8 we give a more detailed per-step overview of the operations performed. For clarity we have chosen qubit 1 in Figure 4.10 (right) as a representative of the memory qubits and qubit A as representative of the measurement qubits..

4.6.4 Surface code logical error probability

By tallying up the contributions from operational and decoherence induced errors we can construct a measure for the total error probability per error correction cycle experienced by all physical qubits that make up the code. Note that this a rather crude model that disregards possible influences from inter-qubit correlated errors and time-like correlated errors. Nevertheless it serves as a useful first approximation to the performance of the surface code on the QDP. We define the average per qubit per cycle error probability P_{tot} as

$$P_{\text{tot}}(d) = 8P_{sw} + 3.5P_{sh} + 10P_z + 4.5P_{gl} + P_m + P_{dec}(d). \tag{4.16}$$

Note that this quantity depends linearly on the code distance d. We can plug this total per cycle error probability P_{tot} into an empirical equation for the logical error probability P_L derived in [97]:

$$P_L = 0.03 \left(\frac{P_{tot}(d)}{8P_{th}} \right)^{\frac{d+1}{2}}, \tag{4.17}$$

where P_{th} is the per-step fault-tolerance threshold of the surface code, which we take to be $P_{th} = 0.0057$ following the result in [97]. The factor of 8 is inserted to account for the fact that the empirical relation derived in [97] is between the physical *per-step* error rate and the logical *per cycle* error rate and the protocol analyzed in [97] requires 8 time-steps per surface code error correction cycle. This is an approximation but it will serve our purposes of getting a basic initial estimate of the logical error

Operation	Error probability	Time
Two-qubit $\sqrt{\text{SWAP}}$ gate	$P_{sw} = 10^{-3}$	$\tau_{sw} = 20\text{ns}$
Coherent shuttle	$P_{sh} = 10^{-3}$	$\tau_{sh} = 10\text{ns}$
Z-rotation by waiting	$P_z = 10^{-3}$	$\tau_z = 100\text{ns}$
Global qubit rotation	$P_{gl} = 10^{-3}$	$\tau_{gl} = 1000\text{ns}$
Measurement	$P_m = 10^{-3}$	$\tau_m = 100\text{ns}$

Table 4.7. Error probabilities and times for the five elementary operations of the QDP.

rate. In Table 4.7, we quote error probabilities and operation times that will be plugged into (4.16). These numbers are projections from [18] and references therein. To convert the operation times into decoherence induced error we use the estimated T_2 time of quantum dot spin qubits in ^{28}Si quoted as $T_2 = 10^9\text{ns}$ [18, 99] and (4.15). Plugging these numbers into (4.16) we get the following linear function of the code distance

$$P_{tot} = 2.7 \times 10^{-2} + 2.8d \times 10^{-5} \tag{4.18}$$

which we can plug into the empirical model (4.17). In Figure 4.11 we plot the logical error probability P_L versus code distance. Note that for the experimental numbers provided the practical quantum computing benchmarking $\log(P_L) = -20$ is reached for a code distance of $d = 37$. The maximal code distance for the experimental parameters is $d = 155$ for which the logarithmical logical error probability reaches $\log(P_L) = -41$, after which it starts increasing again. We also plot what would happen if we had the power to operate the QDP (with quoted device parameters) completely in parallel. The physical error rate of the latter scenario is calculated setting $d = 1$ in (4.18). Note that the difference between parallel and crossbar style operation is not that big, the parallel version reaches $P_L = 10^{-20}$ for $d = 31$. This rough model provides some quantitative justification for the implementation of planar error correction codes in the QDP even in the absence of the ability to arbitrarily suppress logical errors. Note also that, due to the long coherence times of the QDP spin qubits [18, 99], the dominant terms in the expression for the total error probability P_{tot} are those associated with operation induced errors. This provides justification for the line-by-line application of two-qubit gates discussed in Section 4.4.2, which takes a longer time to perform but improves gate quality. It also means that long coherence times and/or

fast operation times are likely critical to the success of a crossbar based scheme. This concludes our discussion of the QDP mapping of the surface code. A similar exercise can be done for the $6.6.6.$ and $4.8.8.$ color codes but due to their lower thresholds [100], the results will likely be less positive for current experimental parameters.

4.7 Conclusion

We analyzed the architecture presented in [18], focusing on its crossbar control system. Building on this analysis we presented procedures for mapping the planar surface code and the $6.6.6.$ and $4.8.8.$ color codes. Because the line-by-line operation of the crossbar architecture means the noise in a single error correction cycle scales with the distance, it is not possible to arbitrarily suppress the logical error rate by increasing the code distance. Instead there will be some "optimal" code distance for which the logical error rate is the lowest. Using numbers for [18] and an empirical model taken from [97] we analyzed the logical error behavior of the surface code mapping and found that, for current experimental numbers, it appears plausible to achieve logical error probabilities below $P_{log} = 10^{-20}$, making practical quantum computation possible. However, we strongly stress that this is a rather crude estimate and a more detailed answer would have to take into account the details of the dominant error processes in quantum dot qubits. It must also take into account that while it is possible to achieve certain low noise gates and good coherence times in quantum dots qubits in isolation this does not necessarily mean they will be practically achievable in the current QDP design. A future research direction would be to perform much more detailed simulations of this crossbar system, perhaps with input from future experiments. In such a simulation the effect of correlated errors (which might feasibly appear in a crossbar architecture) could be investigated.

Another possible research direction would be to use the currently developed machinery to map more exotic quantum error correction codes. A first step in this direction would be the implementation of variants of the surface code with more resistance to biased noise [101, 102]. Due to the possibility of qubit shuttling, also codes with long distance stabilizers could in principle be implemented. Codes such as the 3D gauge color codes might be prime candidates for this kind of treatment. How-

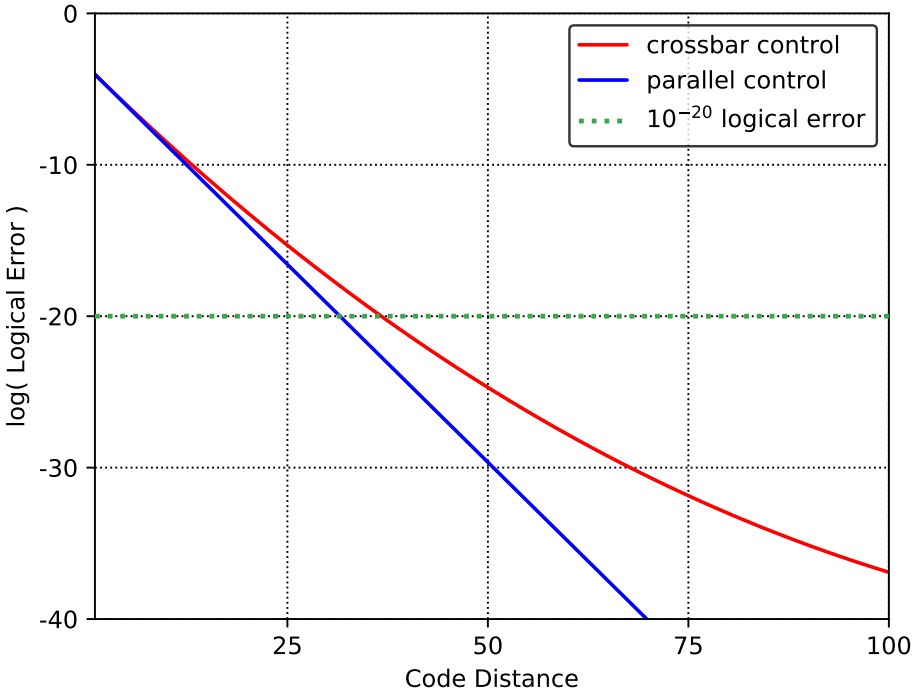


Figure 4.11. Plot of logical error probability versus code distance for the empirical model given in (4.17) with experimental parameters given in Table 4.7. Note that the logical error probability for crossbar operation goes below $P_L = 10^{-20}$ for $d = 37$. This is only slightly slower that parallel operation, which reaches $P_L = 10^{-20}$ for $d = 31$. Due to the scaling of crossbar operation with the code distance the logical error probability bottoms out at some point. This however does not happen until $d = 155$ (not shown) for a logical error rate of $P_L = 10^{-41}$, which is not practically relevant. This rough model gives good indication it is possible to create very low logical error surface code logical qubits in the QDP.

ever, barring some special cases, parallel shuttling is currently being performed in a line-by-line manner. A general classical algorithm for generating optimal (in time) shuttling-steps from an initial to a final BOARDSTATE would vastly simplify the task of mapping more exotic codes and also general quantum circuits. Such an algorithm would probably be useful for any future crossbar quantum architecture.

Lastly, there are important aspects of quantum error correction that are not discussed in this paper. Two of these aspects are the ability to store multiple logical qubits simultaneously and the ability to perform quantum operations on the logical qubits. A popular way of performing these tasks is by encoding multiple logical qubits in a single surface code sheet by introducing topological defects in to the surface code sheet [97]. This process involves not measuring stabilizers at certain points in the sheet, thus creating extra degrees of freedom which can store logical information. The code distance of the code is given by the physical distance (measured in number of physical qubits) between the defects. Operations can then be performed on these logical qubits by moving the defects around each other, a process known as braiding. We think this approach is not natural to the constraints of the crossbar architecture for the following reasons

- Encoding qubits as defects would mean the size of the surface code sheet would scale as the number of encoded qubits. Hence also, in our implementation, the physical error probability per QEC cycle would scale with the number of qubits. This would put an upper limit on the number of qubits that can be implemented.
- Creating and moving defects around requires turning on and off measurements for certain stabilizers in a local manner. This locality runs counter to the design ideas of the crossbar architecture.
- Given that the size of the surface code sheet would scale with the number of logical qubits one would likely face significant issues involving uniformity of control parameters of the entire sheet. This would be a significant issue even if the scaling of the physical error probability can be avoided by clever implementation.

However, we can envision a mode of computation that we speculate is more amenable to this architecture by thinking of an architecture composed of separate modules containing a single logical qubit. We refer

to Figure 7 of [18] for a proposal of implementation. Inside each module our surface code protocol could be run with the ideal code distance given physical error parameters setting the size of these modules. We could then perform logical X- and Z-gates transversally within the modules and we could perform CNOT gates between adjacent modules via lattice surgery. Note that lattice surgery, which involves the turning on and off of stabilizer patches in regular patterns (see [103] for an introduction to lattice surgery), is very amenable to the constraints of the architecture, implying that a high degree of parallelization could be achieved when mapping lattice surgery techniques to the QDP.

4.8 Supplement: surface code operation counts

(The reader may find the corresponding tables on the next pages.)

Steps	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Z-cycle Total
$\sqrt{\text{SWAP}}$ gates					2	2				2	2							8
Z-rotations					2	2				2	1							7
Shuttlings			1				1	1				1		2	1	2	1	10
Global rotations		1											1					2
Measurements																1		1

Table 4.8. Operation counts per step for the Z-measurement during the Z-cycle of the surface code described in Section 4.5.3 and Section 4.6.1. Specifically the Z-measurement qubit is taken to be qubit A in Figure 4.10(right). Table cells that are left empty signify zero entries.

Steps	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	X-cycle Total
$\sqrt{\text{SWAP}}$ gates																		0
Z-rotation																		0
Shuttlings														2	1		1	4
Global rotations		1											1		1			3
Measurements																1		1

Table 4.9. Operation counts per step for the Z-measurement qubit during the X-cycle of the surface code described in Section 4.5.3 and Section 4.6.1. The Z-measurement qubit is taken to be qubit A in Figure 4.10(right). Table cells that are left empty signify zero entries.

Steps	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Z-cycle Total
$\sqrt{\text{SWAP}}$ gates					2						2							4
Z-rotations																		0
Shuttlings			1	1														2
Global rotations		1											1					2
Measurements																		0

Table 4.10. Operation counts per step for a memory qubit during the Z-cycle of the surface code described in Section 4.5.3 and Section 4.6.1. Specifically the memory qubit is taken to be qubit 1 in Figure 4.10(right) but other memory qubits will have the same gate count up to a possible reordering of steps. Table cells that are left empty signify zero entries.

Steps	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	X-cycle Total
$\sqrt{\text{SWAP}}$ gates						2						2						4
Z-rotations																		0
Shuttlings								1	1						1*		1*	4
Global rotations		1											1					2
Measurements																		0

Table 4.11. Operation counts per step for a memory qubit during the X-cycle of the surface code described in Section 4.5.3 and Section 4.6.1. Specifically the memory qubit is taken to be qubit 1 in Figure 4.10(right) but other memory qubits will have the same gate count up to a possible reordering of steps. Table cells that are left empty signify zero entries.
*: Only half of the memory qubits move during this step. In the total operation count this shuttling is counted towards all memory qubits.

4.9 Notations

$a \overset{b}{=} c$	Shorthand for $a \bmod b \ = \ c$.
$[a : b]$	Set of integers from a to b.
(i, j)	Dot locations, in row i and column j.
BOARDSTATE	$(N \times N)$ Matrix mirroring the charge distribution over the grid of dots.
$\mathcal{B}$	Qubits in grid columns of high magnetic field.
CNOT	Controlled-Not gate: $\|0\rangle\langle 0\| \otimes \mathbb{I} + \|1\rangle\langle 1\| \otimes X$.
d	Distance of a quantum code.
CPHASE*	Effective controlled-phase gate, see (4.3).
D$[i][t]$	Set diagonal line i to potential level t, see Table 4.1.
H	Hadamard gate: $(X + Z)/\sqrt{2}$.
H$[i]$	Pulsing horizontal barrier i, see Table 4.1.
HC$[i, j]$	CNOT-gate along horizontal direction, Table 4.3 and Figure 4.4.
HI$[i, j]$	OPCODE for CPHASE* gate between qubits (i, j) and $(i,\ j + 1)$, see Table 4.3.
HS$[i, j, k]$	Horizontal shuttling at (i, j) with flow k, see Table 4.2.
M$[i, j, k]$	Measuring the qubit at (i, j) with the qubit at $(i,\ j+k)$ as a reference, see Table 4.2.
N	The grid inside the processor has the size of $N \times N$ quantum dots, see Figure 4.1(a).
PSB	Pauli spin blockade.
QDP	Abbreviation for *Quantum Dot Processor*, the term we use to describe the proposed quantum device.
$\mathcal{R}$	Qubits in grid columns with low magnetic field.
S	Square-root of a Pauli-Z gate: $\|0\rangle\langle 0\| + i\,\|1\rangle\langle 1\|$.
$\sqrt{\text{SWAP}}$	Square-root of swap gate, (4.2).
V$[i]$	Pulse vertical barrier gate i, see Table 4.1.

$\mathrm{VC}[i,j]$	CNOT-gate in vertical direction, see Table 4.3 and Figure 4.4.
$\mathrm{VI}[i,j]$	OPCODE for $\sqrt{\text{SWAP}}$ gate between qubits (i,j) and $(i+1,\ j)$, see 4.3.
$\mathrm{VS}[i,j,k]$	Vertical shuttling at (i,j) with flow k, see Table 4.2.