Quantum Information and Computation, Vol. 2, No. 6 (2002) 443–486
© Rinton Press

COMPUTATIONAL MODEL UNDERLYING THE ONE-WAY QUANTUM COMPUTER

ROBERT RAUSSENDORF *
Institut für theoretische Physik, Universät München, 80333 München, Germany

HANS J. BRIEGEL†
Institut für theoretische Physik, Universät München, 80333 München, Germany

Received March 14, 2002
Revised August 6, 2002

In this paper we present the computational model underlying the one-way quantum computer which we introduced recently [Phys. Rev. Lett. **86**, 5188 (2001)]. The one-way quantum computer has the property that any quantum logic network can be simulated on it. Conversely, not all ways of quantum information processing that are possible with the one-way quantum computer can be understood properly in network model terms. We show that the logical depth is, for certain algorithms, lower than has so far been known for networks. For example, every quantum circuit in the Clifford group can be performed on the one-way quantum computer in a single step.

Keywords: Models for quantum computation, temporal complexity
Communicated by: R Cleve & G Milburn

1. Introduction

Quantum computation has been formulated within different frameworks such as the quantum Turing machine [1], [2] or the quantum logic network model [3], [4]. In the latter it is particularly easy to establish a connection between physics and the processing of quantum information, since the building blocks of the quantum logic network –the quantum gates– are unitary transformations generated by suitably tailored Hamiltonians. More recently, the capability of projective (von Neumann) measurements to drive a quantum computation has been investigated [5]–[10].

In [9] we have shown that universal quantum computation can be entirely built on one-qubit measurements on a certain class of highly entangled multi-qubit states, the cluster states [11]. In this scheme, a cluster state forms a resource for quantum computation and the set of measurements form the program. This scheme we called the "one-way quantum computer" since the entanglement in a cluster state is destroyed by the one-qubit measurements and therefore the cluster state can be used only once. To stress the importance of the cluster state for the scheme, we will use in the following the abbreviation $\mathrm{QC}_{\mathcal{C}}$ for "one-way quantum computer".

*email: raussen@theorie.physik.uni-muenchen.de
†email: briegel@theorie.physik.uni-muenchen.de

The present paper is motivated in part by the following observation: If a quantum logic network is simulated on a $\mathrm{QC}_{\mathcal{C}}$, for many algorithms the number of computational steps scales more favorably with the input size than the number of steps in the original network does. To be specific, circuits which realize transformations in the Clifford group –which is generated by all the CNOT-gates, Hadamard-gates and $\pi/2$-phase shifts– can be performed by a $\mathrm{QC}_{\mathcal{C}}$ in a single time step, i.e. all the measurements to implement such a circuit can be carried out at the same time. Generally, in a simulation of a quantum logic network by a one-way quantum computer, the temporal ordering of the gates of the network is transformed into a spatial pattern of measurement bases for the individual qubits on the resource cluster state. For the temporal ordering of the measurements there is no counterpart in the network model.

The main point of this paper is to show that the $\mathrm{QC}_{\mathcal{C}}$ has an independent structure which, among other things, determines the temporal order of measurements. As we shall show, the $\mathrm{QC}_{\mathcal{C}}$ has no quantum register and does not consist of quantum gates. The quantities that are processed with the $\mathrm{QC}_{\mathcal{C}}$ are the outcomes of one-qubit measurements and thus processing of information exists only at the classical level. The $\mathrm{QC}_{\mathcal{C}}$ is nevertheless quantum mechanical as it uses a highly entangled cluster state as the central physical resource.

It is true that both the quantum logic network computer and the $\mathrm{QC}_{\mathcal{C}}$ can simulate each other efficiently. The fact that each quantum logic network can be simulated on the $\mathrm{QC}_{\mathcal{C}}$ has been shown in [9]. The converse is also true because a resource cluster state of arbitrary size can be created by a quantum logic network of constant logical depth. Furthermore, the subsequent one-qubit measurements are within the set of standard tools employed in the network scheme of computation. In this sense, the $\mathrm{QC}_{\mathcal{C}}$ does not add physical means to quantum computation. However, while the network model comprises the means that are used in a computation on a $\mathrm{QC}_{\mathcal{C}}$, it cannot describe *how* they have to be used. In particular, in the above construction –where a $\mathrm{QC}_{\mathcal{C}}$ simulating a quantum logic network is itself simulated by a more complicated network– the temporal order of the measurements and the rules to adapt the measurement bases are not contained in a network description. But without this additional information the network to simulate the $\mathrm{QC}_{\mathcal{C}}$ would be incomplete and would not work. Furthermore, the most efficient temporal ordering of measurements to simulate quantum gates on the $\mathrm{QC}_{\mathcal{C}}$ is not pre-imposed by the temporal ordering of these gates in the corresponding quantum logic network. For example, in the network model two gates cannot be performed in parallel if they do not commute. In the $\mathrm{QC}_{\mathcal{C}}$-realization they still can if they both belong to the Clifford group. As a consequence, all circuits in the Clifford group can be parallelized to logical depth $D = 1$ on the $\mathrm{QC}_{\mathcal{C}}$, as mentioned before. Similarly, all circuits which consist of CNOT-gates and either x- or z-rotations with variable angle, can be parallelized to logical depth $D = 2$, independent of the number of logical qubits or gates. The best networks that have been found for the described circuits or at least for special cases thereof have logarithmic depth in the number of qubits or gates, respectively [13]. In the light of this, it seems that the question of complexity of a quantum computation should be revisited.

The paper is organized as follows. Section 2 contains a summary of the $\mathrm{QC}_{\mathcal{C}}$ in network terms, as described in [9]. In Section 3 the terminology to describe the computational model for the $\mathrm{QC}_{\mathcal{C}}$ is developed. The central part of this paper is Section 4 where the computational model for the one-way quantum computer is presented. In Section 5, the non-network

character of the $\mathrm{QC}_{\mathcal{C}}$ is illustrated at the hand of the temporal complexity for the above mentioned two special classes of circuits. Sections 6 and 7 are the discussion and conclusion.

2. Network picture of the $\mathbf{QC}_{\mathcal{C}}$

Before starting the description of the objects relevant for the computational model we would like to give a short summary of the one-way quantum computer. In [9] it was proved that the $\mathrm{QC}_{\mathcal{C}}$ is universal by showing that it can simulate any quantum logic network.

For the one-way quantum computer, the entire resource for the quantum computation is provided initially in the form of a specific entangled state –the cluster state [11]– of a large number of qubits. Information is then written onto the cluster, processed, and read out from the cluster by one-particle measurements only. The entangled state of the cluster thereby serves as a universal "substrate" for any quantum computation. Cluster states can be created efficiently in any system with a quantum Ising-type interaction (at very low temperatures) between two-state particles in a lattice configuration. More specifically, to create a cluster state $|\Phi\rangle_{\mathcal{C}}$ the qubits on a cluster $\mathcal{C}$ are at first all prepared in an individual state $|+\rangle = 1/\sqrt{2}(|0\rangle+|1\rangle)$ and then brought into a cluster state by switching on the Ising-type interaction H_{int} for an appropriately chosen finite time span T. The time evolution operator generated by the Ising-type Hamiltonian which takes the initial product state to the cluster state is denoted by S. The unitary transformation S on a two-dimensional array of qubits, as used in this paper, has the form

$$\begin{aligned} S &= e^{-i\pi \sum_{k\in\mathcal{C}} |0\rangle_k\langle 0|\otimes|1\rangle_{r(k)}\langle 1| + |0\rangle_k\langle 0|\otimes|1\rangle_{u(k)}\langle 1|} \\ &= \left(\prod_{k\in\mathcal{C}} e^{-i\pi/2\,\sigma_z^{(k)}} e^{i\pi/4\,\sigma_z^{(r(k))}} e^{i\pi/4\,\sigma_z^{(u(k))}}\right) e^{-i\pi/4 \sum_{k\in\mathcal{C}} \sigma_z^{(k)}\sigma_z^{(r(k))} + \sigma_z^{(k)}\sigma_z^{(u(k))}}, \end{aligned} \tag{1}$$

where $r(k)$ denotes the site of the right neighbor of qubit k, i.e. the site next following k in the x-direction and $u(k)$ the site of the upper neighbor of k, i.e. the site next following k in the y-direction. The interaction part of S is generated by an Ising Hamiltonian. In (1) there appear additional posterior local unitary transformations which have no influence on the entanglement properties of the states generated under S.

The quantum state $|\Phi\rangle_{\mathcal{C}}$, the cluster state of a cluster $\mathcal{C}$ of neighbouring qubits provides in advance all entanglement that is involved in the subsequent quantum computation. It has been shown [11] that the cluster state $|\Phi\rangle_{\mathcal{C}}$ is characterized by a set of eigenvalue equations

$$\sigma_x^{(a)} \bigotimes_{a'\in ngbh(a)} \sigma_z^{(a')} |\Phi\rangle_{\mathcal{C}} = (-1)^{\kappa_a} |\Phi\rangle_{\mathcal{C}}, \tag{2}$$

where $ngbh(a)$ specifies the sites of all qubits that interact with the qubit at site $a \in \mathcal{C}$. The eigenvalues are specified by the distribution of the qubits on the lattice and encoded in $\{\kappa_a \in \{0,1\}, a = 1,..,\|\mathcal{C}\|\}$. They can be altered e.g. by applying phase-flips σ_z before or after the Ising interaction. For the special case of $\mathcal{C} = \mathbf{Z}^2$, that is the case of an infinitely extended cluster, $\kappa_k = 0\ \forall k \in \mathcal{C}$. The equations (2) are central for the described computation scheme. It is important to realize that information processing is possible even though the result of every individual measurement in any direction of the Bloch sphere is completely random. The reason for the randomness of the measurement results is that the reduced density operator for

each qubit in the cluster state is $\frac{1}{2}\mathbf{1}$. While the individual measurement results are irrelevant for the computation, the strict correlations between measurement results inferred from (2) are what makes the processing of quantum information on the $\mathrm{QC}_{\mathcal{C}}$ possible.

Let us for clarity emphasize that in the scheme of the $\mathrm{QC}_{\mathcal{C}}$ we distinguish between physical cluster qubits in $\mathcal{C}$ which are measured in the process of computation, and the logical qubits. The logical qubits constitute the quantum information being processed while the cluster qubits in the initial cluster state form an entanglement resource. Measurements of their individual one-qubit state drive the computation.

To process quantum information with this cluster, it suffices to measure its particles in a certain order and in a certain basis. Quantum information is thereby propagated through the cluster and processed. Measurements of σ_z-observables effectively remove the respective lattice qubits from the cluster. Measurements of σ_x are used for "wires" i.e. to propagate logical quantum bits through the cluster, and for the CNOT-gate between two logical qubits. Observables of the form $\cos(\varphi)\,\sigma_x + \sin(\varphi)\,\sigma_y$ are measured to realize arbitrary rotations of logical qubits. Here, the angle φ specifies the measurement direction. For the one-qubit rotations, the basis in which a certain qubit is measured depends on the results of preceding measurements. This introduces a temporal ordering in which the measurements have to be performed. The processing is finished once all qubits except a last one on each wire have been measured. The remaining unmeasured qubits form the quantum register which is now ready for the readout. At this point, the results of previous measurements determine in which basis these "output" qubits need to be measured for the final readout, or if the readout measurements are in the σ_x-, σ_y- or σ_z-eigenbasis, how the readout measurements have to be interpreted. Without loss of generality, we assume in this paper that the readout measurements are performed in the σ_z-eigenbasis.

For illustration and later reference we review two points of the universality proof for the $\mathrm{QC}_{\mathcal{C}}$. First, the realization of the arbitrary one-qubit rotation and of the CNOT-gate as the elements of the universal set of gates. And second, the effect of the randomness of the individual measurement results and how to account for them.

An arbitrary rotation $U_{Rot} \in SU(2)$ can be achieved in a chain of 5 qubits. Consider a rotation in its Euler representation

$$U_{Rot}[\xi,\eta,\zeta] = U_x[\zeta]U_z[\eta]U_x[\xi], \tag{3}$$

where the rotations about the x- and z-axis are

$$\begin{aligned} U_x[\alpha] &= \exp\left(-i\alpha\frac{\sigma_x}{2}\right) \\ U_z[\alpha] &= \exp\left(-i\alpha\frac{\sigma_z}{2}\right). \end{aligned} \tag{4}$$

Initially, the first qubit is in some state $|\psi_{\mathrm{in}}\rangle$, which is to be rotated, and the other qubits are in $|+\rangle$. After the 5 qubits are entangled by the time evolution operator S generated by the Ising-type Hamiltonian, the state $|\psi_{\mathrm{in}}\rangle$ can be rotated by measuring qubits 1 to 4. At the same time, the state is also swapped to site 5. The qubits $1\ldots4$ are measured in appropriately chosen bases, *viz.*

$$\mathcal{B}_j(\varphi_{j,\mathrm{meas}}) = \left\{ \frac{|0\rangle_j + e^{i\varphi_{j,\mathrm{meas}}}|1\rangle_j}{\sqrt{2}},\ \frac{|0\rangle_j - e^{i\varphi_{j,\mathrm{meas}}}|1\rangle_j}{\sqrt{2}} \right\} \tag{5}$$

whereby the measurement outcomes $s_j \in \{0,1\}$ for $j = 1 \ldots 4$ are obtained. Here, $s_j = 0$ means that qubit j is projected into the first state of $\mathcal{B}_j(\varphi_{j,\mathrm{meas}})$. In (5) the basis states of all possible measurement bases lie on the equator of the Bloch sphere, i.e. on the intersection of the Bloch sphere with the x-y-plane. Therefore, the measurement basis for qubit j can be specified by a single parameter, the measurement angle $\varphi_{j,\mathrm{meas}}$. The measurement direction of qubit j is the vector on the Bloch sphere which corresponds to the first state in the measurement basis $\mathcal{B}_j(\varphi_{j,\mathrm{meas}})$. Thus, the measurement angle $\varphi_{j,\mathrm{meas}}$ is equal to the angle between the measurement direction at qubit j and the positive x-axis. For all of the so far constructed gates, the cluster qubits are either –if they are not required for the realization of the circuit– measured in σ_z, or –if they are required– measured in some measurement direction in the x-y-plane. In summary, the procedure to implement an arbitrary rotation $U_R(\xi,\eta,\zeta)$, specified by its Euler angles ξ,η,ζ, is this:

$$\begin{array}{ll} 1. & \text{measure qubit 1 in } \mathcal{B}_1(0) \\ 2. & \text{measure qubit 2 in } \mathcal{B}_2\left(-\xi\,(-1)^{s_1+\kappa'_{1,I}}\right) \\ 3. & \text{measure qubit 3 in } \mathcal{B}_3\left(-\eta\,(-1)^{s_2+\kappa'_2}\right) \\ 4. & \text{measure qubit 4 in } \mathcal{B}_4\left(-\zeta\,(-1)^{s_1+s_3+\kappa'_{1,I}+\kappa'_3}\right) \end{array} \tag{6}$$

If the 5-qubit cluster state in Fig. 1 is created from a product state of all qubits in $|+\rangle$ via the interaction S of eq. (1), we have the specific values $\kappa'_{1,I} = 0, \kappa'_2 = \kappa'_3 = \kappa'_4 = \kappa'_{5,O} = 1$. Please note that in (6) we used the set $\left\{\kappa'_{1,I}, \kappa'_2, \kappa'_3, \kappa'_4, \kappa'_{5,O}\right\}$ instead of $\left\{\kappa_l, l = 1,..,5\right\}$ to specify a cluster state $|\phi\rangle_5$ on a chain of 5 qubits. We will use primed κ, $\{\kappa'_a\}$ whenever we describe a cluster state $|\phi\rangle_{\mathcal{C}_N}$ on some cluster $\mathcal{C}_N$ which consists only of those cluster qubits that are necessary for the implementation of some circuit. Further, the κ' associated with the input qubit 1 we denote by $\kappa'_{1,I}$ instead of κ'_1, and the κ' associated with the output qubit 5 we denote by $\kappa'_{5,O}$ instead of κ'_5. The reason for this notation will be discussed in Section 4.1. The unprimed κ, $\{\kappa_a\}$, are reserved for the description of the cluster state $|\phi\rangle_{\mathcal{C}}$ on the whole cluster $\mathcal{C}$.

Via the procedure (6) the rotation U'_{Rot} is realized:

$$U'_{Rot}[\xi,\eta,\zeta] = U_{\Sigma,Rot}\, U_{Rot}[\xi,\eta,\zeta]. \tag{7}$$

Therein, the random byproduct operator has the form

$$U_{\Sigma,Rot} = \sigma_x^{s_2+s_4+\kappa'_2+\kappa'_4}\sigma_z^{s_1+s_3+\kappa'_{1,I}+\kappa'_3+\kappa'_{5,O}}. \tag{8}$$

It can be corrected for at the end of the computation, as explained below.

A CNOT-gate can be implemented on a cluster state of 15 qubits, as shown in Fig. 1. All measurements can be performed simultaneously. Depending on the measurement results, the following gate is thereby realized:

$$U'_{CNOT} = U_{\Sigma,CNOT}\, CNOT(c,t). \tag{9}$$

Therein the byproduct operator $U_{\Sigma,CNOT}$ has the form

$$
\begin{aligned}
U_{\Sigma,CNOT} &= {\sigma_x^{(c)}}^{\gamma_x^{(c)}} {\sigma_x^{(t)}}^{\gamma_x^{(t)}} {\sigma_z^{(c)}}^{\gamma_z^{(c)}} {\sigma_z^{(t)}}^{\gamma_z^{(t)}}, \text{ with} \\
\gamma_x^{(c)} &= s_2 + s_3 + s_5 + s_6 + \kappa'_2 + \kappa'_3 + \kappa'_5 + \kappa'_6 \\
\gamma_x^{(t)} &= s_2 + s_3 + s_8 + s_{10} + s_{12} + s_{14} + \kappa'_2 + \kappa'_3 + \kappa'_8 + \kappa'_{10} + \kappa'_{12} + \kappa'_{14} \\
\gamma_z^{(c)} &= s_1 + s_3 + s_4 + s_5 + s_8 + s_9 + s_{11} + \\
&\quad \kappa'_{1,I} + \kappa'_3 + \kappa'_4 + \kappa'_5 + \kappa'_{7,O} + \kappa'_8 + \kappa'_{9,I} + \kappa'_{11} + 1 \\
\gamma_z^{(t)} &= s_9 + s_{11} + s_{13} + \kappa'_{9,I} + \kappa'_{11} + \kappa'_{13} + \kappa'_{15,O}.
\end{aligned}
\tag{10}
$$

Please note the constant offset for $\gamma_z^{(c)}$ and further that for both the general rotation and the CNOT-gate the byproduct operators depend only on the combinations $s_k + \kappa'_k$ if k is not an output qubit of the gate. The latter is a general feature of the relevant qubits, as will be shown in Section 3.4.1 .

The randomness of the measurement results does not jeopardize the function of the circuit. Depending on the measurement results, extra rotations σ_x and σ_z act on the output qubit of every implemented gate, as in (7), for example. By use of the propagation relation for general one-qubit rotations

$$
U_R[\xi,\eta,\zeta]\, \sigma_z^s \sigma_x^{s'} = \sigma_z^s \sigma_x^{s'}\, U_R[(-1)^s \xi, (-1)^{s'} \eta, (-1)^s \zeta], \tag{11}
$$

and the one for the CNOT-gate, as in [12],

$$
\mathrm{CNOT}(c,t)\, {\sigma_z^{(t)}}^{s_t} {\sigma_z^{(c)}}^{s_c} {\sigma_x^{(t)}}^{s'_t} {\sigma_x^{(c)}}^{s'_c} = {\sigma_z^{(t)}}^{s_t} {\sigma_z^{(c)}}^{s_c+s_t} {\sigma_x^{(t)}}^{s'_c+s'_t} {\sigma_x^{(c)}}^{s'_c}\, \mathrm{CNOT}(c,t), \tag{12}
$$

these extra rotations can be propagated through the network to act upon the output state. By this we mean that the extra random rotations need not be corrected for after a gate. Instead, one just needs to keep track of them and delay correction until the end of the computation. Then, the extra rotations can be accounted for by properly interpreting the σ_z-readout measurement results.

The propagation relations (11) for the arbitrary rotation and (12) for the CNOT-gate differ with respect to which of the two unitary transformations –the gate or the byproduct operator U_Σ- is modified on the right hand side of (11) and (12). In the case of the propagation of a byproduct operator through a rotation (11), the gate is changed and the byproduct operator remains unchanged. It passes just through. Conversely, in the case of the propagation of a byproduct operator U_Σ through a CNOT-gate (12), it is the byproduct operator which is modified and the gate remains unchanged.

The measurement bases $\mathcal{B}(\varphi)$ and $\mathcal{B}(-\varphi)$ in (5) coincide for angles $\varphi = 0$ and for $\varphi = \pm\pi/2$. For $\varphi = 0$ the measurement basis $\mathcal{B}(\varphi)$ is the eigenbasis of σ_x, and for $\varphi = \pm\pi/2$ the measurement basis $\mathcal{B}(\varphi)$ is the eigenbasis of σ_y. In these cases, the choice of the measurement basis is not influenced by the results of measurements at other qubits. Therefore, rotations whose Euler angles ξ, η, ζ are in the set $\{0, \pm\pi/2\}$ can be realized simultaneously in the first round of measurements, that is no other cluster qubits need to be measured before. Among these rotations are the Hadamard gate and the $\pi/2$ phase shift. As displayed in Fig. 1, the Hadamard gate and the $\pi/2$-phase shift are both realized by performing a pattern of σ_x- and

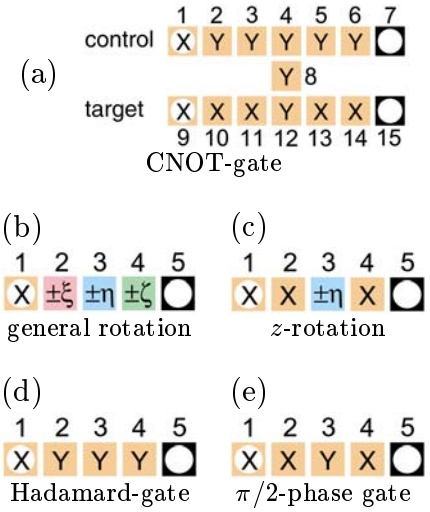

Fig. 1. Realization of elementary quantum gates on the $\mathrm{QC}_{\mathcal{C}}$. Each square represents a lattice qubit. The squares in the extreme left column marked with white circles denote the input qubits, they on the right-most column denote the output qubits. Note that blank squares can represent either σ_z measurements or empty lattice sites.

σ_y-measurements on the cluster $\mathcal{C}$. The byproduct operators which are thereby created are

$$\begin{aligned} U_{\Sigma,H} &= \sigma_x^{s_1+s_3+s_4+\kappa'_{1,I}+\kappa'_3+\kappa'_4}\,\sigma_z^{s_2+s_3+\kappa'_2+\kappa'_3+\kappa'_{5,O}} \\ U_{\Sigma,U_z[\pi/2]} &= \sigma_x^{s_2+s_4+\kappa'_2+\kappa'_4}\,\sigma_z^{s_1+s_2+s_3+\kappa'_{1,I}+\kappa'_2+\kappa'_3+\kappa'_{5,O}+1}. \end{aligned} \tag{13}$$

Owing to the special Euler angles for the Hadamard- and the $\pi/2$-phase gate, the propagation relations for these rotations can also be written in a form resembling the propagation relation (12) for the CNOT-gate

$$\begin{aligned} H\,\sigma_x{}^{s_x}\sigma_z{}^{s_z} &= \sigma_x{}^{s_z}\sigma_z{}^{s_x}\,H, \\ U_z[\pi/2]\,\sigma_x{}^{s_x}\sigma_z{}^{s_z} &= \sigma_x{}^{s_x}\sigma_z{}^{s_x+s_z}\,U_z[\pi/2]. \end{aligned} \tag{14}$$

Under propagation –via the propagation relations (11), (12) and (14)– the byproduct operators resulting from the implementation of the universal gates generate a subgroup of the group

$$\mathcal{U}^{\mathrm{local}} = \left\{1, \sigma_x^{(i)}, \sigma_z^{(i)}, i = 1..n, \text{ and products thereof}\right\} \tag{15}$$

of all possible byproduct operators. $\mathcal{U}^{\mathrm{local}} \subset SU(2)^{\otimes n}$ is a subset of the set of all multi-local unitary operations. Hence, it can be compensated for at the end of the computation by a local change of the measurement bases.

To summarize, any quantum logic network can be simulated on a one-way quantum computer. A set of universal gates can be realized by one-qubit measurements and the gates can be combined to circuits. Due to the randomness of the results of the individual measurements, unwanted byproduct operators are introduced. These byproduct operators can be accounted

for by adapting measurement directions throughout the process of computation. In this way, a subset of qubits on the cluster $\mathcal{C}$ is prepared as the output register. The quantum state on this subset of qubits equals that of the quantum register of the simulated network up to the action of an accumulated byproduct operator. The byproduct operator determines how the measurements on the output register are to be interpreted.

3. Beyond the network picture

In the previous section we have described the $\mathrm{QC}_{\mathcal{C}}$ in a network terminology, which has been useful to prove the universality of the scheme. On the other hand, the cluster qubits do not have to be measured in the order prescribed by the order of the gates in the corresponding network. This observation indicates that the network picture does not describe the $\mathrm{QC}_{\mathcal{C}}$ in every respect.

3.1. *The sets Q_t of simultaneously measurable qubits*

The cluster qubits which we have chosen to take the role of the readout register, for example, are just qubits like any other cluster qubits. It turns out that, in a more efficient way of running the $\mathrm{QC}_{\mathcal{C}}$, the "readout" qubits are not the last ones to be measured but among the first. It is advantageous to forget about the network altogether and to view the $\mathrm{QC}_{\mathcal{C}}$ as a set of one-qubit measurements on a resource quantum state, the cluster state. These measurements have to be performed in a certain order and in a certain basis. The classical information of how to measure subsequent qubits must all be contained in the results of the already performed measurements. Similarly, the final result of the computation must be contained in all the measurement outcomes together.

In the following we will adopt the strategy that every cluster qubit is measured at the earliest possible time. This means that each qubit is measured as soon as the required measurement results from other qubits which determine its measurement basis are known. Let us denote by Q_t the set of qubits which can be measured at the same time in the measurement round t. So, how can the sets Q_t be determined? Q_0 is the set of qubits which are measured in the first round. These are all the qubits whose observables σ_x, σ_y or σ_z are measured. The measurement bases for these qubits do not depend on the results of any previous measurements. To determine the subsequent set Q_1, one looks at which qubits can be measured with the knowledge of the measurement results from the qubits in Q_0. Next, one looks which qubits can be measured with the measurement results from the qubits in Q_0 and Q_1 known. These qubits form the set Q_2. In this manner one proceeds until the whole cluster $\mathcal{C}$ is divided into disjoint subsets Q_t.

As will become clear later, it is useful to introduce the sets $Q^{(t)}$ of yet-to-be measured qubits. More precisely, $Q^{(t)}$ is the set of qubits which remain to be measured after measurement round No. $t-1$,

$$Q^{(t)} = \bigcup_{i=t}^{t_{\max}} Q_i. \tag{16}$$

Mathematically, the sets Q_t are derived from a strict partial ordering in $\mathcal{C}$. The strict partial ordering, in turn, is generated by forward cones which are explained in the next section.

3.2. *The forward- and backward cones*

Be g a gate in the network $\mathcal{N}$ to be simulated and $k \in \mathcal{C}(g)$ a cluster qubit that belongs to the implementation of g. Further, be $\mathcal{O}$, A and Ω three vertical cuts through the network $\mathcal{N}$. A vertical cut is such that it intersects each qubit line in a network only once and that it does not intersect gates. $\mathcal{O}$ intersects $\mathcal{N}$ just after the gate g, i.e. the byproduct operator $U_{\Sigma,g}$ caused by the implementation of g, as given in (8) and (10), is located on $\mathcal{O}$. Note that the byproduct operators generated on $\mathcal{O}$ depend on the measurement results obtained in course of the gate implementation via $(U_k)^{s_k}$. A intersects $\mathcal{N}$ just be fore the input, i.e. an operator propagated to A acts on the input register of $\mathcal{N}$, and Ω intersects $\mathcal{N}$ just before the output such that an operator propagated to Ω acts on the output register of $\mathcal{N}$. For an illustration of the vertical cuts see Fig A.1 in Appendix A. We can now define the forward- and backward cones of the cluster qubits $k \in \mathcal{C}$.

Definition 1 *The forward cone* $\mathrm{fc}(k)$ *of a cluster qubit* $k \in \mathcal{C}$ *is the set of all those cluster qubits* $j \in Q^{(1)}$ *whose measurement basis* $\mathcal{B}(\varphi_{j,\mathrm{meas}})$ *depends on the result* s_k *of the measurement of qubit* k *after the byproduct operator* $(U_k)^{s_k}$ *is propagated from* $\mathcal{O}$ *to* Ω.

Definition 2 *The backward cone* $\mathrm{bc}(k)$ *of a cluster qubit* $k \in \mathcal{C}$ *is the set of all those cluster qubits* $j \in Q^{(1)}$ *whose measurement basis* $\mathcal{B}(\varphi_{j,\mathrm{meas}})$ *depends on the result* s_k *of the measurement of qubit* k *after the byproduct operator* $(U_k)^{s_k}$ *is propagated from* $\mathcal{O}$ *to* A.

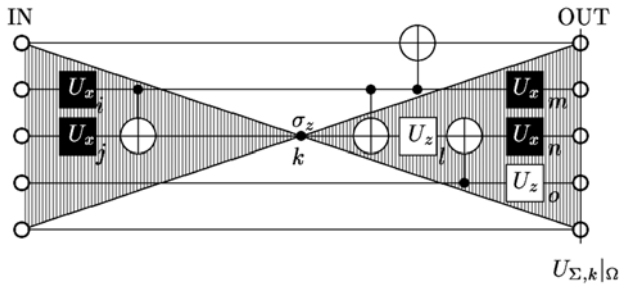

Fig. 2. Forward and backward cones. The measurement of a cluster qubit k for the implementation of the shown quantum logic network may, depending on the measurement outcome, result in a byproduct operator σ_z (the underlying cluster and measurement pattern is not shown). This byproduct operator is propagated forward to act upon the output register as $U_{\Sigma,k}|_\Omega$. In forward propagation, it flips the sign of the measurement angles of the cluster qubits m, n by whose measurement one-qubit rotations are implemented. The cluster qubits m and n are thus in the forward cone of k, $m, n \in \mathrm{fc}(k)$, while l and o are not. Similarly, $i, j \in \mathrm{bc}(k)$.

It will turn out that only the backward cones of the qubits $k \in Q_0$ constitute part of the information specifying an algorithm on the $\mathrm{QC}_\mathcal{C}$, but nevertheless all the backward- and forward cones are important objects in the scheme. Either of the sets, the set of the forward- and that of the backward cones, separately contains the full information of the temporal structure of a computation on the $\mathrm{QC}_\mathcal{C}$.

Let us examine the definitions 1 and 2 for a particular example, the general one-qubit rotation (3) as implemented by the procedure (6) modulo a byproduct operator $U_{\Sigma,Rot}$ as given in (8). The measurement result s_1 of qubit 1 (cf. Fig. 1) modifies the measurement

angle of qubit 2, which is responsible for implementing an x-rotation $U_x(\xi)$, by a factor $(-1)^{s_1}$. Further, it causes a byproduct operator $(\sigma_z)^{s_1}$ at $\mathcal{O}$. If this byproduct operator is propagated forward from $\mathcal{O}$ to Ω it has no effect on qubit 2, because qubit 2 is behind $\mathcal{O}$. The dependence on s_1 of the basis in which qubit 2 has to be measured persists and thus qubit 2 is in the forward cone of qubit 1, $2 \in \mathrm{fc}(1)$. The situation is different if the byproduct operator $(\sigma_z)^{s_1}$ is propagated backwards from $\mathcal{O}$ to A: via the propagation relation (11) the Euler angle ξ is modified by a factor $(-1)^{s_1}$ which has to be accounted for by multiplying the measurement angle $\varphi_{2,\mathrm{meas}}$ by a factor $(-1)^{s_1}$, too. Thus, the factor $(-1)^{s_1}$ modifies the measurement angle $\varphi_{2,\mathrm{meas}}$ twice, once via the procedure (6) and once in backward propagation, and there is no net effect. Qubit 2 is not in the backward cone of qubit 1, $2 \notin \mathrm{bc}(1)$.

What does it mean that a cluster qubit j is in the forward cone of another cluster qubit k, $j \in \mathrm{fc}(k)$? According to the definition, a byproduct operator created via the measurement at cluster qubit k influences the measurement angle $\varphi_{j,\mathrm{meas}}$ at cluster qubit j. To determine the measurement angle at j one must thus wait for the measurement result at k to see what the byproduct operator created randomly by the measurement at k is. If $j \in \mathrm{fc}(k)$, the measurement at qubit j is performed later than that at qubit k. This we denote by $k \prec j$

$$j \in \mathrm{fc}(k) \Rightarrow k \prec j. \tag{17}$$

Please note that the converse of (17) is not true. If $k \prec j$ holds, still $j \in \mathrm{fc}(k)$ may not. This can be easily verified for the example of a general rotation (3). There, according to the procedure for implementing such a rotation described in Section 2, the result of the measurement of qubit 1 enters into in which basis qubit 2 has to be measured. Hence, $2 \in \mathrm{fc}(1)$. By (17), $1 \prec 2$ which means that the measurement at qubit 2 has to wait for the result of the measurement on qubit 1. Similarly, the measurement result on qubit 2 enters in the choice of the measurement basis for the measurement on qubit 3. $3 \in \mathrm{fc}(2)$ and thus $2 \prec 3$. Then $1 \prec 3$ also holds as shown below in (18) , but $3 \notin \mathrm{fc}(1)$, since the measurement result on qubit 1 does not influence the choice of the measurement basis for the measurement on qubit 3.

The relation "$\prec$" is a strict partial ordering. Suppose, that besides $k \prec j$, for another cluster qubit l one had $l \in \mathrm{fc}(j)$ and thus $j \prec l$. This would mean that the measurement at l must wait for the measurement at j, which itself had to wait for the measurement at k. Thus, the measurement at l also had to wait for the measurement at k. Therefore the relation "$\prec$" is transitive,

$$k \prec j \ \wedge \ j \prec l \longrightarrow k \prec l. \tag{18}$$

Further, a measurement to implement a gate cannot and does not need to wait for its own result. Therefore the relation "$\prec$" is anti-reflexive,

$$\neg \exists j \in \mathcal{C} : j \prec j. \tag{19}$$

Let us now cast the procedure to construct the sets of simultaneously measured qubits given above in more precise terms. Be $Q_t \subset \mathcal{C}$ the set of cluster qubits measured in measurement round t, and $Q^{(t)} \subset \mathcal{C}$ the set of qubits which are to be measured in the measurement round t and all subsequent rounds, as defined in (16). Then, Q_0 is the set of qubits which are measured in the first round. These are the qubits of which the observables σ_x, σ_y or σ_z are

measured, so that the measurement bases are not influenced by other measurement results. Further, $Q^{(0)} = \mathcal{C}$. Now, the sequence of sets Q_t can be constructed using the following recursion relation

$$\begin{array}{lcl} Q_t & = & \left\{ q \in Q^{(t)} | \neg \exists p \in Q^{(t)} : p \prec q \right\} \\ Q^{(t+1)} & = & Q^{(t)} \backslash Q_t. \end{array} \tag{20}$$

All those qubits which have no precursors in some remaining set $Q^{(t)}$ and thus do not have to wait for results of measurements of qubits in $Q^{(t)}$ are taken out of this set to form Q_t. The recursion proceeds until $Q^{(t_{\max}+1)} = \emptyset$ for some maximal value $t_{\max}$ of t.

Can it happen that the recursion does not terminate? That were the case if for a number m of qubits $j_1, ..., j_m \in \mathcal{C}$ formed a cycle $j_1 \prec j_2 \prec ... \prec j_m \prec j_1$. Then, none of the qubits $j_1, .., j_n$ could ever taken out of the set. However, by transitivity (18) we then had $j_1 \prec j_1$ which contradicts anti-reflexivity (19). Hence, such a situation cannot occur.

Let us at the end of this section define the forward- and backward cones $\mathrm{fc}(g)$, $\mathrm{bc}(g)$ of the gates g. In eqs. (10) and (13) we have seen that the byproduct operator caused by the implementation of a CNOT- and the $\pi/2$-phase gate contain a constant contribution, $U_0(CNOT) = \sigma_z^{(c)}$ and $U_0(U_z[\pi/2]) = \sigma_z$. This contributions to the respective byproduct operators do not depend on any local variables such as the measurement results and are thus attributed to the gate as a whole. These byproduct operators are of the same form as those depending on the individual measurement results and can influence measurement angles when being propagated forward or backward. Thus we define the forward- and backward cones of gates, in analogy to those of the cluster qubits $k \in \mathcal{C}$, as follows:

The forward cone $\mathrm{fc}(g)$ of a gate $g \in \mathcal{N}$ is the set of all those cluster qubits $j \in Q^{(1)}$ of which the measurement basis $\mathcal{B}(\varphi_{j,\mathrm{meas}})$ is modified if the byproduct operator $U_0(g)$ is propagated forward from $\mathcal{O}$ to Ω.

The backward cone $\mathrm{bc}(g)$ of a gate $g \in \mathcal{N}$ is the set of all those cluster qubits $j \in Q^{(1)}$ of which the measurement basis $\mathcal{B}(\varphi_{j,\mathrm{meas}})$ is modified if the byproduct operator $U_0(g)$ is propagated backward from $\mathcal{O}$ to A.

The forward- and backward cones of gates do not form part of the information representing a quantum algorithm on the $\mathrm{QC}_{\mathcal{C}}$, they will be absorbed into the algorithm angles and the initial value of the information flow vector $\mathbf{I}_{\mathrm{init}}$ introduced below. Their role for the description of a computation on the $\mathrm{QC}_{\mathcal{C}}$ is a technical one.

3.3. *The algorithm- and measurement angles*

There are three different types of angles involved in the described scheme of quantum computation of which the most prominent are the algorithm angles and the measurement angles.

The *algorithm angles* $\left\{ \varphi_{j,\mathrm{algo}}, j \in Q^{(1)} \right\}$ are part of the information that specifies an algorithm on the $\mathrm{QC}_{\mathcal{C}}$. They are derived from the network angles $\left\{ \varphi_{j,\mathrm{qln}}, j \in Q^{(1)} \right\}$, i.e. the Euler angles of the one-qubit rotations in the quantum logic network. Further, the algorithm angles depend on the set $\left\{ \kappa_k, k \in \mathcal{C} \right\}$ characterizing the cluster state $|\phi\rangle_{\mathcal{C}}$ in (2), and on special properties of the measurement pattern. We see that the network angles are absorbed into the algorithm angles. They do not constitute part of the information specifying a $\mathrm{QC}_{\mathcal{C}}$-algorithm.

As described before, the process of computation with the $\mathrm{QC}_{\mathcal{C}}$ comprises several measure-

ment rounds. The first round, in wich the qubits in the set Q_0 are measured, is somewhat different from the following rounds. Therein, all gates of the circuit that belong to the Clifford group are implemented at the same time, no matter where they are located in the corresponding quantum logic network and in which step they would be carried out there. This results in byproduct operators scattered all over the place. These byproduct operators are, according to the scheme described in Section 4.2, propagated backwards. To account for the effect that the byproduct operators have on the algorithm angles, these angles have to updated to the *modified algorithm angles* $\{\varphi'_{j,\mathrm{algo}}, j \in Q^{(1)}\}$. The modified algorithm angles $\varphi'_{j,\mathrm{algo}}$ are calculated from the respective algorithm angles $\varphi_{j,\mathrm{algo}}$ and the results obtained in the first measurement round $\{s_k, k \in Q_0\}$. In the subsequent measurement rounds no further update of the modified algorithm angles occurs. Finally, each qubit $j \in Q_t \subset Q^{(1)}$ is measured in some measurement round t in the basis $\mathcal{B}(\varphi_{j,\mathrm{meas}})$ where $\varphi_{j,\mathrm{meas}}$ denotes the measurement angle of qubit j. The *measurement angle* $\varphi_{j,\mathrm{meas}}$ of a qubit $j \in Q_t$ is calculated from the modified algorithm angle, $\varphi'_{j,\mathrm{algo}}$ and the results $\{s_k, k \in \bigcup_{i=0}^{t-1} Q_i\}$ of the so far obtained measurements.

Before a quantum algorithm is run on the $\mathrm{QC}_{\mathcal{C}}$, the algorithm angles are determined from the cluster and the properties of the algorithm. During runtime of the $\mathrm{QC}_{\mathcal{C}}$, in the first measurement round ($t = 0$), the algorithm angles are replaced by the modified algorithm angles, i.e. only the latter are kept while the former are erased. Then, in the measurement round t a qubit $j \in Q_t$ is measured in the basis determined by the measurement angle $\varphi_{j,\mathrm{meas}}$. After the measurement of qubit j both $\varphi'_{j,\mathrm{algo}}$ and $\varphi_{j,\mathrm{meas}}$ can be erased.

Now there arises the question of how the measurement angles of the actual measurements are calculated from the results of previous measurements. This question will be answered in Section 4.2. The question which interests us most, of course, is: "How can the final result of the computation be determined from all the measurement outcomes?" It will turn out that the answers to both questions are very much related.

3.4. *Quantities for the processing of the measurement results*

3.4.1. *The information vector*

Let us again –as in Section 3.1– make the change of the viewpoint from an explanation of the $\mathrm{QC}_{\mathcal{C}}$ within the network model to a more suitable description, but now focusing on the quantities required to describe the processing of the measurement results.

Suppose in the simulation of a quantum logic network $\mathcal{N}$ on the $\mathrm{QC}_{\mathcal{C}}$ in a network manner –i.e. measuring the "readout" qubits at last– the processing has reached the stage where all but those cluster qubits have been measured which form the output register.

The accumulated byproduct operator U_Σ to act upon the logical output qubits $1, .., n$ is known. It has the form

$$U_\Sigma = \prod_{i=1}^{n} \left(\sigma_x^{(i)}\right)^{x_i} \left(\sigma_z^{(i)}\right)^{z_i}, \tag{21}$$

where $x_i, z_i \in \{0, 1\}$ for $1 \leq i \leq n$. Let us now label the unmeasured qubits on the cluster in the same way as the readout qubits on the quantum logic network are labelled.

The "readout" qubits on the cluster are, at this point, in a state $U_\Sigma|\mathrm{out}\rangle$, where $|\mathrm{out}\rangle$ is

the output state of the corresponding quantum logic network. In the network picture, the computation is completed by measuring each qubit in the σ_z-eigenbasis, thereby obtaining the measurement results $\{s_i'\}$, say. In the $\mathrm{QC}_\mathcal{C}$ scheme, one measures the state $U_\Sigma|\mathrm{out}\rangle$ directly, whereby outcomes $\{s_i\}$ are obtained and the "readout" qubits are projected into the state $|\mathcal{M}\rangle = \prod_{i=1}^n \frac{1+(-1)^{s_i}\sigma_z^{(i)}}{2}\, U_\Sigma|\mathrm{out}\rangle$. Depending on the known byproduct operator U_Σ, the set of measurement results $\{s_i\}$ in general has a different interpretation from what the network readout $\{s_i'\}$ would have. The measurement basis is the same. From (21) one obtains

$$\begin{aligned} |\mathcal{M}\rangle = \prod_{i=1}^n \tfrac{1+(-1)^{s_i}\sigma_z^{(i)}}{2}\, U_\Sigma|\mathrm{out}\rangle &= U_\Sigma \left(U_\Sigma^\dagger \prod_{i=1}^n \tfrac{1+(-1)^{s_i}\sigma_z^{(i)}}{2} U_\Sigma\right)|\mathrm{out}\rangle \\ &= U_\Sigma \prod_{i=1}^n \tfrac{1+(-1)^{s_i+x_i}\sigma_z^{(i)}}{2}|\mathrm{out}\rangle \end{aligned} \tag{22}$$

From (22) we see that a σ_z-measurement on the state $U_\Sigma\,|\mathrm{out}\rangle$ with results $\{s_i\}$ represents the same algorithmic output as a σ_z-measurement of the state $|\mathrm{out}\rangle$ with the results $\{s_i'\}$, where the sets $\{s_i\}$ and $\{s_i'\}$ are related by

$$s_i' \equiv s_i + x_i \bmod 2. \tag{23}$$

The set $\{s_i'\}$ represents the result of the computation. It can be calculated from the results $\{s_i\}$ of the σ_z-measurements on the "readout" cluster qubits, and the values $\{x_i\}$ which can be extracted from the byproduct operator U_Σ. We see that both the contribution of the byproduct operator and the result of the measurement on the "readout" qubits of the cluster enter expression (23) in the same way. Indeed, there is no need to distinguish between these two contributions. On the level of the byproduct operators, the readout measurement result is translated into an additional contribution to the accumulated byproduct operator. Both contributions to the such extended byproduct operator

$$U_{\Sigma R} = U_\Sigma U_R, \text{ with } U_R = \prod_{i=1}^n \left(\sigma_x^{(i)}\right)^{s_i}, \tag{24}$$

stem from random measurement results. It is just that the contributions which constitute U_Σ must be propagated forward from where they originated and the additional contributions from the readout measurements must be propagated one step backwards. Both the forward and the backward propagated contributions to $U_{\Sigma R}$ are propagated to the same location in the network. Forward and backward propagation are closely related. In fact, as the propagation relations (11), (12) and (14) are their own inverse, the rules are the same for both directions. The distinguished role of the readout qubits is only a remnant of the interpretation of the $\mathrm{QC}_\mathcal{C}$ as a quantum logic network. A more adequate description will have the consequence that the cluster qubits on the "output register", for instance, will be measured during the initialization of the $\mathrm{QC}_\mathcal{C}$ such that they are removed from the entangled quantum state even before the main part of the computation starts.

We now define the the *information vector* $\mathbf{I}$, a $2n$-component binary vector which is a function of the quantities $\{x_i, z_i\}$ and the results $\{s_i\}$ of the measurements on the cluster output register. Together, these quantities determine the extended byproduct operator $U_{\Sigma R}$ via (21), and (24).

Definition 3 *The information vector* $\mathbf{I}$ *is given by*

$$\mathbf{I} = \begin{pmatrix} \mathbf{I}_x \\ \\ \mathbf{I}_z \end{pmatrix}, \quad with \ \mathbf{I}_x = \begin{pmatrix} x_1 + s_1 \\ x_2 + s_2 \\ . \\ . \\ . \\ x_n + s_n \end{pmatrix}, \quad \mathbf{I}_z = \begin{pmatrix} z_1 \\ z_2 \\ . \\ . \\ . \\ z_n \end{pmatrix}. \tag{25}$$

As can be seen from (23) and (25), $\mathbf{I}_x$ is a possible result of a readout measurement in a corresponding quantum logic network. $\mathbf{I}_z$ is redundant. However, in Section 4.1 the flow quantity $\mathbf{I}(t)$, the *information flow vector*, will be defined for which $\mathbf{I}(t_{\max}) = \mathbf{I}$, with $t_{\max}$ the index of the final computational step. For $t < t_{\max}$, in $\mathbf{I}(t)$ both the z-part $\mathbf{I}_z(t)$ and the x-part $\mathbf{I}_x(t)$ are required to determine the bases for the one-qubit measurements in Q_{t+1}. As $\mathbf{I}_z(t)$ is of equal importance as $\mathbf{I}_x(t)$ throughout the process of computation we keep $\mathbf{I}_z$ in the definition of $\mathbf{I}$ as well.

The set of possible information vectors $\mathbf{I}$ forms a $2n$ dimensional vector space over F_2, $\mathcal{V}$. Let us consider the group $\mathcal{U}^{\text{local}}$ of all possible extended byproduct operators $U_{\Sigma R}$. If we divide out the normal divisor $\{\pm 1\}$ of $\mathcal{U}^{\text{local}}$, the resulting factor group $\overline{\mathcal{U}} = \mathcal{U}^{\text{local}}/\{\pm 1\}$ is isomorphic to $\mathcal{V}$. From the viewpoint of physics, dividing out the normal divisor $\{\pm 1\}$ means that we ignore a global phase. The isomorphism $\mathcal{I}$ which maps an $\mathbf{I} \in \mathcal{V}$ to the corresponding $U_{\Sigma R} \in \overline{\mathcal{U}}$ is given by

$$\mathcal{I}: \ \mathcal{V} \ni \mathbf{I} \longrightarrow U_{\Sigma R} = \prod_{i=1}^{n} \left(\sigma_x^{(i)}\right)^{[\mathbf{I}_x]_i} \left(\sigma_z^{(i)}\right)^{[\mathbf{I}_z]_i} \in \overline{\mathcal{U}}, \tag{26}$$

where $[\mathbf{I}_x]_i$ and $[\mathbf{I}_z]_i$ are the respective components of $\mathbf{I}_x$ and $\mathbf{I}_z$. The component-wise addition of vectors in $\mathcal{V}$ corresponds, via the isomorphism $\mathcal{I}$, to the multiplication of byproduct operators modulo a phase factor $\{\pm 1\}$. The procedure to implement this product is to first use the operator product, then bring the factors into normal order according to (26) and finally drop the phase. Multiplication of vectors $\mathbf{I} \in \mathcal{V}$ with the scalars 0,1 corresponds to raising the byproduct operators $U_{\Sigma R} \in \overline{\mathcal{U}}$ to the respective powers. One may switch between the two pictures via the isomorphism (26). The algebraic structures involved will be more apparent in the representation using the information vector $\mathbf{I} = \mathbf{I}(\{x_i, z_i, s_i\})$ than in the formulation of the operator $U_{\Sigma R}$.

Now that we have defined the information vector $\mathbf{I}$ in (25) and have seen that the result of the computation can be directly read off from the x-part of $\mathbf{I}$, we would like to find out how $\mathbf{I}$ depends on the measurement outcomes $\{s_k\}$ and the set $\{\kappa_k\}$ of binary numbers that determine the cluster state $|\phi\rangle_{\mathcal{C}}$ in (2). This task is left until Section 4.1. Before we can accomplish it we need some further definitions. It will turn out that the information vector $\mathbf{I}$ can be written as a linear combination of the *byproduct images* which are explained next.

3.4.2. *The byproduct images*

Be Ω the "cut" through a network $\mathcal{N}$ which intersects the qubit lines just before its output. This is the cut at which the extended byproduct operator $U_{\Sigma R}$ is accumulated. Consider a qubit k on the cluster $\mathcal{C}$ which is measured in the course of computation. Depending on the

result of the measurement on qubit k, a byproduct operator $(U_k)^{s_k}$ is introduced in $\mathcal{N}$ at the location of the logical output qubits of the gate for whose implementation the cluster qubit k was measured. This byproduct operator U_k can –by using the propagation relations (11), (12) and (14)– propagated from where it occurred to the cut Ω. There it appears as the forward propagated byproduct operator $U_k|_\Omega$. Now we can define the *byproduct image* $\mathbf{F}_k$ of a cluster qubit $k \in \mathcal{C}$. Each cluster qubit $k \in \mathcal{C}$ has a byproduct image.

Definition 4 *For each cluster qubit $k \in \mathcal{C}$ the byproduct image $\mathbf{F}_k$ is the vector that corresponds via the isomorphism $\mathcal{I}^{-1}$ (26) to the forward propagated byproduct operator $U_k|_\Omega$,*

$$\mathbf{F}_k = \mathcal{I}^{-1}\left(U_k|_\Omega\right). \tag{27}$$

In the definition (27) of the byproduct image $\mathbf{F}_k$ it is mentioned only implicitly that the image is evaluated on the cut Ω. Later in the discussion it will become apparent that we could evaluate the byproduct image on every vertical cut $\mathcal{O}$. Sometimes, if we compare to other vertical cuts, we will explicitly write $\mathbf{F}_k|_\Omega$ for $\mathbf{F}_k$.

The set of byproduct images $\{\mathbf{F}_k, k \in \mathcal{C}\}$ is an important quantity for the scheme. It represents part of the information which is needed to run a quantum algorithm with the $\mathrm{QC}_\mathcal{C}$.

In eq. (10) the byproduct operator for the CNOT-gate as realized according to Fig. 1 is given. This byproduct operator contains a constant contribution $U_0(CNOT) = \sigma_z^{(c)}$. As U_0 does not depend on any local variables, neither on $\{s_k\}$ nor on $\{\kappa_k\}$, it makes no sense to attribute it to any of the cluster qubits that were measured to realize the gate. Instead, it is attributed to the part of the measurement pattern that implements the gate as a whole, or –for simplicity– to the gate itself. For any gate g, $U_0(g)$ can be propagated forward to the cut Ω to act upon the "readout" qubits. There it appears as the forward propagated byproduct operator $U_0(g)|_\Omega$. In analogy to the byproduct images of the cluster qubits, we can now define the byproduct images of the gates g of the quantum logic network that is simulated on the $\mathrm{QC}_\mathcal{C}$. For any such gate g the byproduct image $\mathbf{F}_g$ is the vector that corresponds to $U_0(g)|_\Omega$ via

$$\mathbf{F}_g = \mathcal{I}^{-1}(U_0(g)|_\Omega). \tag{28}$$

Please note that in contrast to the byproduct images $\mathbf{F}_k$ of cluster qubits $k \in \mathcal{C}$ the byproduct images $\mathbf{F}_g$ of gates do not form a separate part of the information specifying a quantum algorithm on the $\mathrm{QC}_\mathcal{C}$. They will be absorbed into the initialization value $\mathbf{I}_{\mathrm{init}}$ of the information flow vector defined in Section 4.1 and they are thus only a convenient tool in the derivation of the computational model.

Via $\mathcal{I}^{-1}$ we map the multiplication of byproduct operators, i.e. their accumulation, onto addition modulo 2 on the level of the vectors in $\mathcal{V}$. Now there arises the question whether other operations on the byproduct operators could be expressed in terms of the corresponding vectors, too. Specifically, one may ask how the byproduct operator propagation looks like on the level of the $\mathbf{I} \in \mathcal{V}$.

3.4.3. *The propagation matrices*

The answer to this question is that on the level of the vector quantities in $\mathcal{V}$ propagation is described by multiplication with certain $2n \times 2n$-matrices C. Consider two cuts $\mathcal{O}_1$ and $\mathcal{O}_2$

through a network which intersect each qubit line only once. Further, be the two cuts such that they do not intersect each other and that $\mathcal{O}_1$ is earlier than $\mathcal{O}_2$. The part of the quantum logic network between $\mathcal{O}_1$ and $\mathcal{O}_2$ is denoted by $\mathcal{N}_{\mathcal{O}_1 \to \mathcal{O}_2}$. Be $\mathbf{I}_k|_{\mathcal{O}_1}$ and $\mathbf{I}_k|_{\mathcal{O}_2}$ the vectors describing a byproduct operator resulting from the measurement of qubit k, propagated to the cuts $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively. Then we have

$$\mathbf{I}_k|_{\mathcal{O}_2} = C(\mathcal{N}_{\mathcal{O}_1 \to \mathcal{O}_2})\, \mathbf{I}_k|_{\mathcal{O}_1}. \tag{29}$$

To any quantum logic network $\mathcal{N}$ a matrix $C_{\mathcal{N}}$ can be assigned. For a network $\mathcal{N}_2 \circ \mathcal{N}_1$ composed of two subnetworks $\mathcal{N}_1$ and $\mathcal{N}_2$ (of which $\mathcal{N}_1$ is carried out first) the propagation matrix is equal to the product of the propagation matrices of the subnetworks

$$C(\mathcal{N}_2 \circ \mathcal{N}_1) = C(\mathcal{N}_2) C(\mathcal{N}_1). \tag{30}$$

Because of property (30) we only need to find the propagation matrices for the general one-qubit rotations, the CNOT-, the Hadamard- and the $\pi/2$-phase gate. The one-qubit rotations and the CNOT-gate alone form a universal set of gates. The reason why we also include the Hadamard- and the $\pi/2$-phase gate is that here they are treated differently from the general rotations, as can be seen from the propagation relations (11) and (14). By propagation through a Hadamard- or $\pi/2$-phase gate, the gate is left unchanged while the byproduct operator changes; whereas for the propagation through a general rotation, the rotation changes and the byproduct operator stays the same. Thus, for finding the byproduct images the general rotations in $\mathcal{N}$ can be replaced by the identity. Only the CNOT-, Hadamard and $\pi/2$-phase gates have an effect. The special treatment of the Hadamard and the $\pi/2$-phase gate is advantageous with respect to the temporal complexity of a computation, because if one uses the propagation relation (14) the implementation of the Hadamard- and the $\pi/2$-phase gate does not need to wait for results of any previous measurements. To sum up, to each possible $\mathcal{N}$ belongs a unitary operation $U(\mathcal{N})$ in the Clifford group and a corresponding matrix $C(\mathcal{N})$, such that

$$\mathcal{I}\left(C(\mathcal{N})\mathbf{I}\right) = U(\mathcal{N})\, \mathcal{I}\left(\mathbf{I}\right)\, U(\mathcal{N})^{\dagger}, \quad \forall \mathbf{I} \in \mathcal{V}. \tag{31}$$

Let us now give the propagation matrices for propagation through CNOT-, Hadamard and $\pi/2$-phase gates. The propagation matrices C are conveniently written in block form

$$C = \left(\begin{array}{c|c} C_{xx} & C_{zx} \\ \hline C_{xz} & C_{zz} \end{array}\right), \tag{32}$$

where C_{xx}, C_{zx}, C_{xz} and C_{zz} are $n \times n$ matrices with binary-valued entries.

For the Hadamard gate $H^{(i)}$ on the logical qubit i one finds

$$\begin{array}{rcl} \left[C_{xx}(H^{(i)})\right]_{kl} = \left[C_{zz}(H^{(i)})\right]_{kl} & = & \delta_{kl} + \delta_{ki}\delta_{il}, \\ \left[C_{zx}(H^{(i)})\right]_{kl} = \left[C_{xz}(H^{(i)})\right]_{kl} & = & \delta_{ki}\delta_{il}, \end{array} \tag{33}$$

where e.g. $\left[C_{xx}(H^{(i)})\right]_{kl}$ denotes the entry of row k and column l in C_{xx}. Note that the qubit index i is not summed over in (33) and that the addition is modulo 2.

For the $\pi/2$-phase gate $U_z^{(i)}(\pi/2)$ on the logical qubit i one finds

$$\begin{aligned}\left[C_{xx}(U_z^{(i)}(\pi/2))\right]_{kl} &= \delta_{kl},\\ \left[C_{zz}(U_z^{(i)}(\pi/2))\right]_{kl} &= \delta_{kl},\\ \left[C_{xz}(U_z^{(i)}(\pi/2))\right]_{kl} &= \delta_{ki}\delta_{il},\\ \left[C_{zx}(U_z^{(i)}(\pi/2))\right]_{kl} &= 0.\end{aligned} \tag{34}$$

For the CNOT-gate on control qubit c and target qubit t one finds the propagation matrix $C(\mathrm{CNOT}(c,t))$ with

$$\begin{aligned}[C_{xx}(\mathrm{CNOT}(c,t))]_{kl} &= \delta_{kl} + \delta_{kt}\delta_{cl},\\ [C_{zz}(\mathrm{CNOT}(c,t))]_{kl} &= \delta_{kl} + \delta_{kc}\delta_{tl},\\ C_{zx}(\mathrm{CNOT}(c,t)) &= 0,\\ C_{xz}(\mathrm{CNOT}(c,t)) &= 0.\end{aligned} \tag{35}$$

We will make use of the propagation matrices in the discussion of temporal complexity of algorithms on the $\mathrm{QC}_{\mathcal{C}}$ in Section 5.2.

For the action of the propagation matrices C on the vectors $\mathbf{I} \in \mathcal{V}$ there exist conserved quantities. One of them, $\mathbf{I}_{x,1}^T\mathbf{I}_{z,2} + \mathbf{I}_{z,1}^T\mathbf{I}_{x,2}$, is discussed in the next section.

3.4.4. *Conservation of the symplectic scalar product*

The symplectic scalar product

$$(\mathbf{I}_1, \mathbf{I}_2)_S = \mathbf{I}_{x,1}^T\mathbf{I}_{z,2} + \mathbf{I}_{z,1}^T\mathbf{I}_{x,2} \bmod 2 \tag{36}$$

is conserved. For any $\mathbf{I}_1, \mathbf{I}_2 \in \mathcal{V}$ and C the identity

$$(\mathbf{I}_1, \mathbf{I}_2)_S = (C\mathbf{I}_1, C\mathbf{I}_2)_S \tag{37}$$

holds. Let us briefly explain why the symplectic scalar product (36) is conserved. First, note that the symplectic scalar product tells whether two operators $\mathcal{I}(\mathbf{I}_1), \mathcal{I}(\mathbf{I}_2)$ in the Pauli group commute or anti-commute,

$$\mathcal{I}(\mathbf{I}_1)\mathcal{I}(\mathbf{I}_2) = (-1)^{(\mathbf{I}_1,\mathbf{I}_2)_S}\mathcal{I}(\mathbf{I}_2)\mathcal{I}(\mathbf{I}_1). \tag{38}$$

Relation (38) is the only place in this paper where we pay attention to the sign factor of a byproduct operator. There, the product, e.g. $\mathcal{I}(\mathbf{I}_1)\mathcal{I}(\mathbf{I}_2)$, denotes the usual operator product. However, everywhere else in this paper a product $\mathcal{I}(\mathbf{I}_1)\mathcal{I}(\mathbf{I}_2)$ denotes operator multiplication modulo a global phase factor ± 1, i.e. the product is normal ordered as in (26) and the phase factor is dropped.

Using relation (38), the invariance (37) of the scalar product (36) is easily demonstrated. Consider the quantity $\mathcal{I}(C\mathbf{I}_1)\mathcal{I}(C\mathbf{I}_2)$ with $\mathcal{I}(C\mathbf{I}_1) = U\,\mathcal{I}(\mathbf{I}_1)\,U^\dagger$ and $\mathcal{I}(C\mathbf{I}_2) = U\,\mathcal{I}(\mathbf{I}_2)\,U^\dagger$ as

in (31). Then, we find

$$\begin{aligned}\mathcal{I}(C\mathbf{I}_1)\mathcal{I}(C\mathbf{I}_2) &= U\,\mathcal{I}(\mathbf{I}_1)\,U^\dagger\,U\,\mathcal{I}(\mathbf{I}_2)\,U^\dagger \\ &= U\,\mathcal{I}(\mathbf{I}_1)\,\mathcal{I}(\mathbf{I}_2)\,U^\dagger \\ &= (-1)^{(\mathbf{I}_1,\mathbf{I}_2)_S}U\,\mathcal{I}(\mathbf{I}_2)\,\mathcal{I}(\mathbf{I}_1)\,U^\dagger \\ &= (-1)^{(\mathbf{I}_1,\mathbf{I}_2)_S}U\,\mathcal{I}(\mathbf{I}_2)\,U^\dagger\,U\,\mathcal{I}(\mathbf{I}_1)\,U^\dagger \\ &= (-1)^{(\mathbf{I}_1,\mathbf{I}_2)_S}\mathcal{I}(C\mathbf{I}_2)\mathcal{I}(C\mathbf{I}_1),\end{aligned} \tag{39}$$

where the third line holds by (38). On the other hand, as we can see from (38) directly that

$$\mathcal{I}(C\mathbf{I}_1)\mathcal{I}(C\mathbf{I}_2) = (-1)^{(C\mathbf{I}_1,C\mathbf{I}_2)_S}\mathcal{I}(C\mathbf{I}_2)\mathcal{I}(C\mathbf{I}_1). \tag{40}$$

From (39) and (40) together it follows that $(\mathbf{I}_1,\mathbf{I}_2)_S = (C\mathbf{I}_1, C\mathbf{I}_2)_S$, as stated in (37).

The symplectic scalar product (36) will prove useful in determining the measurement angles from previously obtained measurement results.

3.4.5. *The cone test*

The cone test is used to find out whether two measurements, which are part of some gates of a circuit, influence each other, i.e. whether one of the measurements has to wait for the result of the other. The cone test does not reveal which of the two measurements has to be performed first.

Let j,k be some cluster qubits $k \in \mathcal{C}$ and $j \in Q^{(1)}$. Qubit j is not measured in the first measurement round and thus the observable measured at qubit j is a nontrivial linear combination of σ_x and σ_y, hence j can be in the forward and backward cones of some other cluster qubits. We would like to find out whether j is in the forward or backward cone of k. For this question the cone test provides a necessary and sufficient criterion. It reads

$$\forall\, k \in \mathcal{C}, j \in Q^{(1)} :\ \ j \in \mathrm{fc}(k) \vee j \in \mathrm{bc}(k) \Longleftrightarrow (\mathbf{F}_j, \mathbf{F}_k)_S = 1. \tag{41}$$

To check whether a qubit lies in some other qubits backward or forward cone we only need the two byproduct images and can use the symplectic scalar product.

We further observe that

$$\forall\, j,k \in Q^{(1)} :\ \ j \in \mathrm{fc}(k) \Longleftrightarrow k \in \mathrm{bc}(j). \tag{42}$$

If we confine k to $k \in Q^{(1)} \subset \mathcal{C}$ we can insert (42) into (41) such that the expression on the l.h.s. of (41) becomes symmetric with respect to j and k. This fits in well since the r.h.s of (41) is also symmetric.

Similar to (41) we can give a criterion for whether or not a qubit $j \in Q^{(1)}$ is in the forward- or backward cone $\mathrm{fc}(g)$, $\mathrm{bc}(g)$ of some gate g. It reads

$$\forall g \in \mathcal{N},\ j \in Q^{(1)} :\ j \in \mathrm{fc}(g) \vee j \in \mathrm{bc}(g) \Longleftrightarrow (\mathbf{F}_j, \mathbf{F}_g)_S = 1. \tag{43}$$

The proofs of the cone tests (41), (43) and the forward-backward cone relation (42) are given in Appendix A.

3.5. *To what a quantum logic network condenses*

Simulating a quantum logic network on a $\mathrm{QC}_{\mathcal{C}}$ is a two-stage process. Before the genuine computation, we feed a classical computer with the network to be simulated. It returns the quantities needed to run the respective algorithm on the $\mathrm{QC}_{\mathcal{C}}$. These quantities are the sets Q_t of simultaneously measurable qubits, the measurement bases of the qubits $k \in Q_0$, the algorithm angles $\varphi_{l,\mathrm{algo}}$ for $l \in \mathcal{C}\backslash Q_0$, the backward cones $\mathrm{bc}(k)$ of the qubits $k \in Q_0$, the byproduct images $\mathbf{F}_j$ for $j \in \mathcal{C}$ and the initialization value $\mathbf{I}_{\mathrm{init}}$ of the information flow vector $\mathbf{I}(t)$. Together these quantities represent the program for the $\mathrm{QC}_{\mathcal{C}}$.

In [9] we wrote that the set of one-qubit measurements on a cluster state represents the program. Now we can be more specific about the measurement pattern representing the program for the $\mathrm{QC}_{\mathcal{C}}$. The measurement pattern has both a temporal and a spatial structure. The temporal structure is given by the sets Q_t of simultaneously measured qubits. The spatial structure consists of the bases (σ_x-, σ_y- or σ_z-) of the measurements in the first round and of the measurement angles in the subsequent rounds. The measurement angles can be determined only run-time, since they involve the random outcomes of previous measurements.

4. Computational model for the $\mathbf{QC}_{\mathcal{C}}$

In the preceding two sections we have established the notions of the sets of simultaneously measurable qubits, backward cones, byproduct images, measurement angles and the information vector. In this section, the computational model underlying the $\mathrm{QC}_{\mathcal{C}}$ is described in these terms. First, we would like to give a summary of the characteristic features of the model:

- The $\mathrm{QC}_{\mathcal{C}}$ has no quantum input and no quantum output.
- For any given quantum algorithm, the cluster $\mathcal{C}$ is divided into disjoint subsets $Q_t \subset \mathcal{C}$ of qubits, $t = 0, 1, \ldots, t_{\max}$, where $Q_p \cap Q_q = \emptyset$ for $p \neq q$ and $\bigcup_{t=0}^{t_{\max}} Q_t = \mathcal{C}$. These subsets are measured one after the other in the order given by the index t. In measurement round t the set Q_t of qubits is measured.
- The classical information gained by the measurements is processed within a flow scheme. The flow quantity is a classical $2n$-component binary vector $\mathbf{I}(t)$, where n is the number of logical qubits of a corresponding quantum logic network and t the number of the measurement round.
- This vector $\mathbf{I}(t)$, the *information flow vector*, is updated after every measurement round. That is, after the one-qubit measurements of all qubits of a set Q_t have been performed simultaneously, $\mathbf{I}(t-1)$ is updated to $\mathbf{I}(t)$ through the results of these measurements. In turn, $\mathbf{I}(t)$ determines which one-qubit observables are to be measured of the qubits of the set Q_{t+1}.
- The result of the computation is given by the information flow vector $\mathbf{I}(t_{\max})$ after the last measurement round. From this quantity the readout measurement result of the quantum register in the corresponding quantum logic network can be read off directly without further processing.

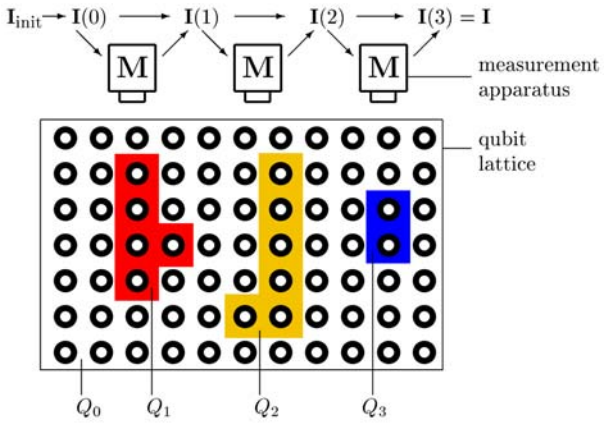

Fig. 3. General scheme of the quantum computer via one-qubit measurements. The sets Q_t of lattice qubits are measured one after the other. The results of earlier measurements determine the measurement bases of later ones. All classical information from the measurement results needed to steer the $\mathrm{QC}_{\mathcal{C}}$ is contained in the information flow vector $\mathbf{I}(t)$. After the last measurement round $t_{\max}$, $\mathbf{I}(t_{\max})$ contains the result of the computation.

We should make a comment on the first point. The $\mathrm{QC}_{\mathcal{C}}$ has no quantum output. Of course, the final result of any computation –including quantum computations– is a classical number, but for the quantum logic network the state of the output register before the readout measurements plays a distinguished role. For the $\mathrm{QC}_{\mathcal{C}}$ this is not the case, there are just cluster qubits measured in a certain order and basis. If, to perform a particular algorithm on the $\mathrm{QC}_{\mathcal{C}}$, a quantum logic network is implemented on a cluster state there is a subset of cluster qubits which play the role of the output register. These qubits are, however, not the final qubits to be measured, but among the first (!).

The $\mathrm{QC}_{\mathcal{C}}$ has no quantum input. This means that the quantum input state is *known* and can thus be created from some standard quantum state, e.g. $|00...0\rangle$, by a circuit preceding the main part of the computation. Shor's algorithm where one starts with an input state $\bigotimes_{i=1}^{n} 1/\sqrt{2}(|0\rangle_i + |1\rangle_i)$ is an example for such a situation. Other scenarios are conceivable, e.g. where an unknown quantum input is processed and the classical result of the computation is retransmitted to the sender of the input state; or the unmeasured network output register state is retransmitted. These scenarios would lead only to slight modifications in the computational model. They are, however, not in the focus of this paper. The reader who is interested in how to read in and process an unknown quantum state with the $\mathrm{QC}_{\mathcal{C}}$ is referred to [9].

4.1. *Obtaining the computational result from the measurement outcomes*

Now that we have defined the information vector $\mathbf{I}$ in (25) and have seen that the result of the computation can be directly read off from the x-part of $\mathbf{I}$, we will explain how $\mathbf{I}$ depends on the measurement outcomes $\{s_k\}$ and the set $\{\kappa_k\}$ of binary numbers that determine the cluster state $|\phi\rangle_{\mathcal{C}}$ in (2). For this purpose, we will express $U_{\Sigma R}$ in terms of $\{s_k\}$ and $\{\kappa_k\}$, and use the isomorphism (26) to obtain $\mathbf{I}$. As in (24), $U_{\Sigma R}$ can be decomposed into two parts, the

accumulated byproduct operator U_Σ and the contribution from the "readout" measurements, U_R. We will proceed in three steps. First, we will consider U_Σ for the case where only the cluster qubits for the "readout" and those of which observables $\cos\varphi\,\sigma_x + \sin\varphi\,\sigma_y$ are measured are present. Second, we will extend the obtained result to the case where both the relevant and the redundant qubits are present, i.e. where a universal cluster $\mathcal{C}$ is used for computation. Third, we will include the contribution U_R from the "readout" qubits.

To derive $\mathbf{I}$ as a function of $\{s_k\}$ and $\{\kappa_k\}$, we need to define the following sets. $\mathcal{C}$ is a universal cluster. Let $O \subset \mathcal{C}$ be the subset of the cluster which, in the simulation of a quantum logic network on the $\mathrm{QC}_\mathcal{C}$, consists of the readout qubits. Let $\mathcal{C}_N \subset \mathcal{C}$ denote the cluster that contains only the relevant cluster qubits, i.e. those which are measured in a direction in the equator of the Bloch sphere, and the "readout" qubits. Be $Q_{0,z} \subset \mathcal{C}$ the set of qubits of which the operator σ_z is measured. Among these sets, the following relations hold:

$$\begin{aligned} \mathcal{C}_N \cup Q_{0,z} &= \mathcal{C} \\ \mathcal{C}_N \cap Q_{0,z} &= O. \end{aligned} \tag{44}$$

Further, let us denote a standard form of the cluster on which the gate g can be implemented as $\mathcal{C}(g)$. The cluster $\mathcal{C}(g)$ shall consist only of essential qubits, i.e. those which are measured in a direction in the equator of the Bloch sphere (x-y-plane). The set $\mathcal{C}_N$ of qubits is the union of all clusters $\mathcal{C}(g)$

$$\mathcal{C}_N = \bigcup_{g \in \mathcal{N}} \mathcal{C}(g). \tag{45}$$

Each $\mathcal{C}(g)$ is subdivided into an input zone $\mathcal{C}_I(g)$, an output zone $\mathcal{C}_O(g)$ and the body of the gate $\mathcal{C}_M(g)$; $\mathcal{C}_I(g) \cap \mathcal{C}_O(g) = \emptyset, \mathcal{C}_M(g) = \mathcal{C}(g) \setminus (\mathcal{C}_I(g) \cup \mathcal{C}_O(g))$. In the two cases described in Section 2, the general rotation and the CNOT-gate, these sets are $\mathcal{C}(U_{Rot}(\xi,\eta,\zeta)) = \{1,2,3,4,5\}$, $\mathcal{C}_I(U_{Rot}(\xi,\eta,\zeta)) = \{1\}$, $\mathcal{C}_M(U_{Rot}(\xi,\eta,\zeta)) = \{2,3,4\}$, $\mathcal{C}_O(U_{Rot}(\xi,\eta,\zeta)) = \{5\}$ and $\mathcal{C}(CNOT) = \{1,..,15\}$, $\mathcal{C}_I(CNOT) = \{1,9\}$, $\mathcal{C}_M(CNOT) = \{2,..,6,8,10,..,14\}$, $\mathcal{C}_O(CNOT) = \{7,15\}$, with the labeling of qubit sites according to Fig. 1.

Let us now explain in greater detail the notation $\kappa_{k,I}, \kappa_{l,O}$ which made use of in equations (6), (8) and (10) in Section 2. Some of the sets $\mathcal{C}(g)$ have an overlap. If a qubit k in the output zone $\mathcal{C}_O(g)$, $k \in \mathcal{C}_O(g)$ is not a readout qubit, it is also in the input zone of some gate $\tilde{g}$ succeeding g, i.e. $k \in \mathcal{C}_I(\tilde{g})$. The procedures to implement the gates $g, \tilde{g}$ on clusters $\mathcal{C}(g), \mathcal{C}(\tilde{g})$ depend on the eigenvalues of the states $|\phi\rangle_{\mathcal{C}(g)}, |\phi\rangle_{\mathcal{C}(\tilde{g})}$ in the respective eigenvalue equations of form (2). Namely, the procedures depend on the set $\{\kappa'_a, a \in \mathcal{C}(g)\}$ or $\{\kappa'_a, a \in \mathcal{C}(\tilde{g})\}$, respectively. In the described case, we have $k \in \mathcal{C}(g)$, $k \in \mathcal{C}(\tilde{g})$ and $k \in \mathcal{C}_N$. Thus, there are three eigenvalue equations associated with k, one for the state $|\phi\rangle_{\mathcal{C}(g)}$, one for $|\phi\rangle_{\mathcal{C}(\tilde{g})}$ and one for $|\phi\rangle_{\mathcal{C}_N}$. In all the three cases the symbols $K^{(k)}$ denote different operators, because the sets $ngbh(k)$ are different. For $\mathcal{C}(g)$, $k \in \mathcal{C}_O(g)$ has a left but no right neighbor. For $\mathcal{C}(\tilde{g})$, $k \in \mathcal{C}_I(\tilde{g})$ has a right but no left neighbor. For $\mathcal{C}_N$, k has both a left and right neighbor. We specify the eigenvalue in the relevant equation (2) for the state $|\phi\rangle_{\mathcal{C}(g)}$, where k is an output qubit, by $\kappa'_{k,O}$, i.e. $K^{(k)}|\phi\rangle_{\mathcal{C}(g)} = (-1)^{\kappa'_{k,O}}|\phi\rangle_{\mathcal{C}(g)}$. Similarly, we specify the eigenvalue in the equation for $|\phi\rangle_{\mathcal{C}(\tilde{g})}$, where k is an input qubit, by $\kappa'_{k,I}$ and in the equation for the state $|\phi\rangle_{\mathcal{C}_N}$

by κ'_k. The $\kappa'_{k,I}$, $\kappa'_{k,O}$ and κ'_k are generally different, but related in the following way

$$\begin{array}{lrcl} \forall k \mid \exists\, g, \tilde{g} \in \mathcal{N}, \text{ such that } k \in \mathcal{C}_O(g) \wedge k \in \mathcal{C}_I(\tilde{g}): & \kappa'_k & = & \kappa'_{k,I} + \kappa'_{k,O}, \\ \forall k \mid \exists g \in \mathcal{N}, \text{ s.th. } k \in \mathcal{C}_O(g) \ \wedge \ \neg\exists \tilde{g} \in \mathcal{N}, \text{ s.th. } k \in \mathcal{C}_I(\tilde{g}): & \kappa'_k & = & \kappa'_{k,O} \\ \forall k \mid \neg\exists g \in \mathcal{N}, \text{ s.th. } k \in \mathcal{C}_O(g) \ \wedge \ \exists \tilde{g} \in \mathcal{N}, \text{ s.th. } k \in \mathcal{C}_I(\tilde{g}): & \kappa'_k & = & \kappa'_{k,I}. \end{array} \tag{46}$$

The latter case of (46) applies to cluster qubits which belong to the cluster equivalent of the input- or output quantum register. The first line of (46) is proved in Appendix B, the second and third line are straightforward.

For all qubits k in the gate bodies, i.e. for which $\exists g \in \mathcal{N}$ such that $k \in \mathcal{C}_M(g) \subset \mathcal{C}_N$, the κ'_k that defines $|\phi\rangle_{\mathcal{C}(g)}$ and the κ'_k that defines $|\phi\rangle_{\mathcal{C}_N}$ are equivalent,

$$\forall k \,|\, \exists g \in \mathcal{N}, \text{ such that } k \in \mathcal{C}_M(g): \ \kappa'_{k\in\mathcal{C}(g)} = \kappa'_{k\in\mathcal{C}_N} \tag{47}$$

and are therefore denoted by the same symbol. Provided with these definitions and relations, we can now start to express $\mathbf{I}$ in terms of $\{s_k, k \in \mathcal{C}\}$ and $\{\kappa_k, k \in \mathcal{C}\}$.

Let us –in the first step– discuss the accumulated byproduct operator U_Σ for a computation on the special cluster $\mathcal{C}_N$. To U_Σ contribute all the byproduct operators $U_\Sigma(g)$ that are created in the implementation of the gates g. For all necessary cases, the general rotations (8), the CNOT-gate (10) and the special rotations Hadamard-gate and $\pi/2$-phase-gate (13), the byproduct operators $U_\Sigma(g)$ can be written in the form

$$U_\Sigma(g) = \left(\prod_{k\in\mathcal{C}_I(g)} (U_k)^{s_k+\kappa'_{k,I}}\right)\left(\prod_{k\in\mathcal{C}_M(g)} (U_k)^{s_k+\kappa'_k}\right)\left(\prod_{k\in\mathcal{C}_O(g)} (U_k)^{\kappa'_{k,O}}\right) U_0(g), \tag{48}$$

where we have attributed the contributions to U_Σ which depend on $\kappa'_{k,I}$, κ'_k or $\kappa'_{k,O}$ to the qubit k. $U_0(g)$ is constant in the measurement outcomes $\{s_k, k \in \mathcal{C}_N\backslash O\}$ and $\{\kappa'_k, k \in \mathcal{C}_N\backslash O\}$ and we therefore attribute it to the gate g as a whole rather than to a particular cluster qubit. For all rotations we have $U_0(g) = \mathbf{1}$, but for the CNOT-gate –if realized as depicted in Fig. 1– the contribution is nontrivial as can be read off from (10), $U_0(CNOT) = \sigma_z^{(c)}$.

To determine the effect of $U_\Sigma(g)$ on U_Σ we propagate, by use of the propagation relations (11), (12) and (14), the byproduct operators $U_\Sigma(g)$ forward to the cut Ω which intersects the corresponding network $\mathcal{N}$ just before the output. The forward propagated byproduct operator that results from the byproduct operator $U_\Sigma(g)$ we denote by $U_\Sigma(g)|_\Omega$. In the same way, the forward propagated byproduct operator originating from U_k, the byproduct operator generated via the measurement of qubit k, is denoted by $U_k|_\Omega$ for all $k \in \mathcal{C}_N\backslash O$. Finally, the forward propagated byproduct operator originating from $U_0(g)$, the byproduct operator attributed to the gate g as a whole, is denoted by $U_0(g)|_\Omega$. To give an explicit expression, be $\mathcal{O}$ the vertical cut through a network $\mathcal{N}$ at the output of a gate g and $U(\mathcal{N}_{\mathcal{O}\to\Omega})$ the unitary operation in the Clifford group which corresponds to the part of the network $\mathcal{N}$ with all the one-qubit rotations except for the Hadamard- and $\pi/2$-phase gates replaced by the identity, as explained in Section 3.4.3. Then, the forward propagated byproduct operators are given by

$$\begin{array}{rcl} U_\Sigma(g)|_\Omega & = & U(\mathcal{N}_{\mathcal{O}\to\Omega})\, U_\Sigma(g)\, U(\mathcal{N}_{\mathcal{O}\to\Omega})^\dagger \\ U_k|_\Omega & = & U(\mathcal{N}_{\mathcal{O}\to\Omega})\, U_k\, U(\mathcal{N}_{\mathcal{O}\to\Omega})^\dagger \\ U_0(g)|_\Omega & = & U(\mathcal{N}_{\mathcal{O}\to\Omega})\, U_0(g)\, U(\mathcal{N}_{\mathcal{O}\to\Omega})^\dagger \end{array} \tag{49}$$

The contribution $U_\Sigma(g)|_\Omega$ from the gate g to U_Σ is

$$U_\Sigma(g)|_\Omega = \left(\prod_{k\in\mathcal{C}_I(g)} U_k|_\Omega{}^{s_k+\kappa'_{k,I}}\right)\left(\prod_{k\in\mathcal{C}_M(g)} U_k|_\Omega{}^{s_k+\kappa'_k}\right)\left(\prod_{k\in\mathcal{C}_O(g)} U_k|_\Omega{}^{\kappa'_{k,O}}\right) U_0(g)|_\Omega. \quad (50)$$

The total byproduct operator U_Σ is the product of all forward propagated byproduct operators $U_\Sigma(g)|_\Omega$, $U_\Sigma = \prod_{g\in\mathcal{N}} U_\Sigma(g)|_\Omega$, and thus given by

$$U_\Sigma = \prod_{g\in\mathcal{N}}\left(U_0(g)|_\Omega \prod_{k\in\mathcal{C}_I(g)} U_k|_\Omega{}^{s_k+\kappa'_{k,I}} \prod_{k\in\mathcal{C}_M(g)} U_k|_\Omega{}^{s_k+\kappa'_k} \prod_{k\in\mathcal{C}_O(g)} U_k|_\Omega{}^{\kappa'_{k,O}}\right). \quad (51)$$

As can be seen from eq. (51), for $k\in\mathcal{C}_M(g),\ g\in\mathcal{N}$ the accumulated byproduct operator U_Σ depends only on the combinations $s_k+\kappa'_k$. This is, in fact, true for all $k\in\mathcal{C}$ as will be shown below. That this property of U_Σ in its general form is not directly visible in eq. (51) is a disadvantage of this equation. A further disadvantage of (51) is that it depends on the $\kappa'_{k,I},\ k\in\mathcal{C}_I(g)$, $\kappa'_l,\ l\in\mathcal{C}_M(g)$ and $\kappa'_{m,O},\ m\in\mathcal{C}_O(g)$, but not on the $\kappa_k,\ k\in\mathcal{C}$ which define the cluster state $|\phi\rangle_\mathcal{C}$.

To find a more convenient expression for U_Σ we first observe that by use of eq. (51) U_Σ can be written in the form

$$U_\Sigma = \left(\prod_{k\in\mathcal{C}_N\backslash O} U_k|_\Omega{}^{s_k}\right)\cdot \mathrm{const}\left(\left\{\kappa'_k,\ k\in\mathcal{C}_N\right\}\right). \quad (52)$$

This is possible because in eq. (51) for $k\in\mathcal{C}_O(g)$ the factors contributing to U_Σ do not depend on $\{s_k\}$. Thus, if one only picks the s_k-dependent contributions the product index variable k runs over $\bigcup_{g\in\mathcal{N}}(\mathcal{C}_I(g)\cup\mathcal{C}_M(g)) = \mathcal{C}_N\backslash O$. Now, U_Σ depends for all $k\in\mathcal{C}_N\backslash O$ only on the combinations $s_k+\kappa'_k$ which can be seen as follows: Let us consider two cluster states $|\phi\rangle_{\mathcal{C}_N}$ and $|\tilde{\phi}\rangle_{\mathcal{C}_N}$ on the cluster $\mathcal{C}_N$ which are related via $|\tilde{\phi}\rangle_{\mathcal{C}_N} = \sigma_z^{(k)}|\phi\rangle_{\mathcal{C}_N}$, such that the respective eigenvalues are related by $\kappa'_k(|\tilde{\phi}\rangle_{\mathcal{C}_N}) = \kappa'_k(|\phi\rangle_{\mathcal{C}_N}) + 1 \bmod 2$, and $\kappa'_l(|\tilde{\phi}\rangle_{\mathcal{C}_N}) = \kappa'_l(|\phi\rangle_{\mathcal{C}_N})$ for all $l\neq k$. Suppose one would use the cluster state $|\phi\rangle_{\mathcal{C}_N}$ for a computation. For each choice of $k\in\mathcal{C}_N\backslash O$ qubit k is measured in some direction $\vec{r}$ on the equator of the Bloch sphere, i.e. the operator $\vec{r}\cdot\vec{\sigma}^{(k)}$ with $\vec{r}\cdot\vec{e}_z = 0$ is measured, and the measurement result is s_k. For the resulting state one finds

$$\frac{1+(-1)^{s_k}\vec{r}\cdot\vec{\sigma}^{(k)}}{2}|\phi\rangle_{\mathcal{C}_N} = \sigma_z^{(k)}\frac{1+(-1)^{s_k+1}\vec{r}\cdot\vec{\sigma}^{(k)}}{2}|\tilde{\phi}\rangle_{\mathcal{C}_N}. \quad (53)$$

The state on the r.h.s. of eq. (53) is modulo the posterior $\sigma_z^{(k)}$ equal to the state into which one had projected if the cluster state $|\tilde{\phi}\rangle_{\mathcal{C}_N}$ instead of $|\phi\rangle_{\mathcal{C}_N}$ was used for computation and the result $s_k+1 \bmod 2$ was obtained. Since we are only interested in the measurement results but not the resulting quantum state, the posterior $\sigma_z^{(k)}$ is irrelevant for the computation. Using a cluster state $|\phi\rangle_{\mathcal{C}_N}$, characterized by κ'_k, and obtaining the measurement result s_k is equivalent to using a cluster state $|\tilde{\phi}\rangle_{\mathcal{C}_N}$, characterized by $\kappa'_k+1 \bmod 2$, and obtaining the measurement result $s_k+1 \bmod 2$. Thus, U_Σ can depend only on $s_k+\kappa'_k \bmod 2$ for all

$k \in \mathcal{C}_N \backslash O$, i.e.

$$U_\Sigma = \left(\prod_{k \in \mathcal{C}_N \backslash O} {U_k}^{s_k + \kappa'_k} \right) \cdot \text{const.} \tag{54}$$

Now, the constant operator can be identified via comparison of eqs. (51) and (54). For this purpose, we choose $\left\{ \kappa'_{k,I} = 0, \kappa'_l = 0, \kappa'_{m,O} = 0, \ \forall g \in \mathcal{N}, k \in \mathcal{C}_I(g), l \in \mathcal{C}_M(g), m \in \mathcal{C}_O(g) \right\}$ and insert it into eq. (51). This choice implies via eq. (46) that $\left\{ \kappa'_k = 0, \ \forall k \in \mathcal{C}_N \right\}$ which we insert into eq. (54). In this way we find const $= \prod_{g \in \mathcal{N}} U_0(g)|_\Omega$ and thus

$$U_\Sigma = \prod_{k \in \mathcal{C}_N \backslash O} {U_k}^{s_k + \kappa'_k} \cdot \prod_{g \in \mathcal{N}} U_0(g)|_\Omega. \tag{55}$$

Let us now –in the second step– include the effect of the qubits $l \in Q_{0,z} \backslash O$ on U_Σ. These qubits are among the redundant qubits which are measured in the first measurement round. Redundant here always means redundant with respect to a given quantum logic network to be simulated. If one starts with a universal cluster state $|\phi\rangle_\mathcal{C}$ on a cluster $\mathcal{C}$ and projects out the qubits $l \in Q_{0,z} \backslash O$ the resulting state on the unmeasured qubits $k \in \mathcal{C}_N$ is again a cluster state $|\phi\rangle_{\mathcal{C}_N}$. The eigenvalues that specify the state $|\phi\rangle_{\mathcal{C}_N}$ in an equation analogous to eq. (2) depend on the results of the σ_z-measurements on the qubits $l \in Q_{0,z} \backslash O$. For the set $\left\{ \kappa'_k, \ k \in \mathcal{C}_N \right\}$ that specifies $|\phi\rangle_{\mathcal{C}_N}$ one finds

$$\forall k \in \mathcal{C}_N : \ \kappa'_k = \kappa_k + \sum_{j \in nbgh(k) \cap Q_{0,z} \backslash O} s_j, \tag{56}$$

as can be easily derived from eq. (2). In eq. (56) the κ_k are those which specify the universal cluster state $|\phi\rangle_\mathcal{C}$ in (2). If one now inserts eq. (56) into (55) one obtains

$$U_\Sigma = \left(\prod_{k \in \mathcal{C}_N \backslash O} {U_k|_\Omega}^{s_k + \kappa_k} \right) \left(\prod_{k \in \mathcal{C}_N \backslash O} \prod_{j| \ \substack{j \in nbgh(k) \wedge \\ j \in Q_{0,z} \backslash O}} {U_k|_\Omega}^{s_j} \right) \left(\prod_{g \in \mathcal{N}} U_0(g)|_\Omega \right). \tag{57}$$

Therein and in the following it should be understood that a product of operators is set equal to the unity operator if the index variable runs over an empty set. The second factor in eq. (57) can now be rewritten in the following way

$$\begin{aligned} \prod_{k \in \mathcal{C}_N \backslash O} \prod_{j| \ \substack{j \in nbgh(k) \wedge \\ j \in Q_{0,z} \backslash O}} {U_k|_\Omega}^{s_j} &= \prod_{j \in \mathcal{C}_N \backslash O} \prod_{k| \ \substack{j \in nbgh(k) \wedge \\ k \in Q_{0,z} \backslash O}} {U_j|_\Omega}^{s_k} \\ &= \prod_{(j,k)| \ \substack{j \in \mathcal{C}_N \backslash O \wedge \\ j \in nbgh(k) \wedge \\ k \in Q_{0,z} \backslash O}} {U_j|_\Omega}^{s_k} \\ &= \prod_{k \in Q_{0,z} \backslash O} \prod_{j| \ \substack{j \in nbgh(k) \wedge \\ j \in \mathcal{C}_N \backslash O}} {U_j|_\Omega}^{s_k}. \end{aligned} \tag{58}$$

In the first line of (58) the labels j and k were interchanged and the relation $j \in nbgh(k) \iff k \in nbgh(j)$ was used. In the second and third line the order of the products over k and j was interchanged.

We now define the forward propagated byproduct operators $U_k|_\Omega$ for qubits $k \in Q_{0,z}\backslash O = \mathcal{C}\backslash\mathcal{C}_N$ as

$$U_k|_\Omega = \prod_{\substack{j|\ j \in nbgh(k) \wedge \\ j \in \mathcal{C}_N\backslash O}} U_j|_\Omega\,, \qquad \forall k \in Q_{0,z}\backslash O. \tag{59}$$

In this way, we have traced back the forward propagated byproduct operators for qubits $k \in Q_{0,z}\backslash O$ to those for qubits $j \in \mathcal{C}_N\backslash O$ which are already known. On the level of the corresponding byproduct images we find via the isomorphism (26)

$$\mathbf{F}_k = \sum_{\substack{j|\ j \in nbgh(k) \wedge \\ j \in \mathcal{C}_N\backslash O}} \mathbf{F}_j\,, \qquad \forall k \in Q_{0,z}\backslash O. \tag{60}$$

If we insert the definition (59) into (58) we obtain

$$\prod_{k\in\mathcal{C}_N\backslash O} \prod_{\substack{j|\ j \in nbgh(k) \wedge \\ j \in Q_{0,z}\backslash O}} {U_k|_\Omega}^{s_j} = \prod_{k\in Q_{0,z}\backslash O} {U_k|_\Omega}^{s_k}. \tag{61}$$

Thus, with $Q_{0,z}\backslash O = \mathcal{C}\backslash\mathcal{C}_N$ from eq. (44) and substituting (61) into (57), one finds

$$U_\Sigma = \left(\prod_{k\in\mathcal{C}\backslash O} {U_k|_\Omega}^{s_k}\right)\left(\prod_{k\in\mathcal{C}\backslash Q_{0,z}} {U_k|_\Omega}^{\kappa_k}\right)\left(\prod_{g\in\mathcal{N}} U_0(g)|_\Omega\right). \tag{62}$$

Finally –in the third step– we investigate the contribution from U_R to $U_{\Sigma R}$ which comes from the set O of "output" qubits. With eq. (24) we have $U_R = \prod_{k=1}^{n} \left(\sigma_x^{(k)}\right)^{s_k}$ where the n cluster qubits in the set O shall be labeled in the same way as the n logical qubits. Then, we can define the propagated byproduct operators $U_k|_\Omega$ for $k \in O$ as

$$U_k|_\Omega = \sigma_x^{(k)}\,, \qquad \forall k \in O, \tag{63}$$

and the corresponding byproduct images via (27)

$$\mathbf{F}_k = (0_1, 0_2, .., 0_{k-1}, 1_k, 0_{k+1}, .., 0_n; 0_{n+1}, .., 0_{2n})^T\,, \qquad \forall k \in O, \tag{64}$$

where e.g. 1_k denotes a 1 at the kth position of $\mathbf{F}_k$, $k \in 0$. Combining eqs. (62) and (63) in (24), the extended byproduct operator $U_{\Sigma R} = U_\Sigma U_R$ becomes

$$U_{\Sigma R} = \left(\prod_{k\in\mathcal{C}} {U_k|_\Omega}^{s_k}\right)\left(\prod_{k\in\mathcal{C}\backslash Q_{0,z}} {U_k|_\Omega}^{\kappa_k}\right)\left(\prod_{g\in\mathcal{N}} U_0(g)|_\Omega\right). \tag{65}$$

Via the isomorphism (26) and using the definitions of the byproduct images (27) and (28) one can now express the information vector $\mathbf{I} = \mathcal{I}^{-1}(U_{\Sigma R})$ as a function of the measurement results $\{s_k,\ k \in \mathcal{C}\}$ and the $\{\kappa_k,\ k \in \mathcal{C}\}$ defining the cluster state $|\phi\rangle_\mathcal{C}$ on the cluster $\mathcal{C}$,

$$\mathbf{I} = \sum_{k\in\mathcal{C}} s_k\mathbf{F}_k + \sum_{k\in\mathcal{C}\backslash Q_{0,z}} \kappa_k\mathbf{F}_k + \sum_{g\in\mathcal{N}} \mathbf{F}_g \tag{66}$$

To derive the expression (66) for the information vector has been the primary purpose of this section.

With the expression (66) at hand we are finally able to define the quantity which carries the algorithmic information during the computational process and which has already been mentioned on earlier occasions in this paper, the *information flow vector* $\mathbf{I}(t)$.

Definition 5 *The information flow vector* $\mathbf{I}(t)$ *is given by*

$$\mathbf{I}(t) = \sum_{k \in \bigcup_{i=0}^{t} Q_i} s_k \, \mathbf{F}_k + \sum_{k \in \mathcal{C} \setminus Q_{0,z}} \kappa_k \mathbf{F}_k + \sum_{g \in \mathcal{N}} \mathbf{F}_g \tag{67}$$

The quantity $\mathbf{I}(t)$ is similar to $\mathbf{I}$ as given in (66), but to $\mathbf{I}(t)$ only contribute the byproduct images of qubits from a subset $\bigcup_{i=1}^{t} Q_i$ of $\mathcal{C}$. The information flow vector $\mathbf{I}(t_{\max})$ after the final measurement round $t_{\max}$ equals the information vector $\mathbf{I}$,

$$\mathbf{I} = \mathbf{I}(t_{\max}). \tag{68}$$

As will be shown later, during all steps of the computation, except for after the final one, the information flow vector determines the measurement bases for the cluster qubits that are to be measured in the next round. After the final round it contains the result of the computation. Thus, it has a meaning in every step of the computation. No further information obtained from the measurements is needed. In this sense, the information flow vector can be regarded as the carrier of the *algorithmic information* on the $\mathrm{QC}_{\mathcal{C}}$.[‡]

The information flow vector has a constant part which does not depend on the measurement results $\{s_k\}$. This part alone forms its initialization value $\mathbf{I}_{\mathrm{init}}$,

$$\mathbf{I}_{\mathrm{init}} = \sum_{k \in \mathcal{C} \setminus Q_{0,z}} \kappa_k \mathbf{F}_k + \sum_{g \in \mathcal{N}} \mathbf{F}_g, \tag{69}$$

such that $\mathbf{I}(t)$ becomes

$$\mathbf{I}(t) = \mathbf{I}_{\mathrm{init}} + \sum_{k \in \bigcup_{i=0}^{t} Q_i} s_k \, \mathbf{F}_k. \tag{70}$$

From eq. (69) we see that the byproduct images of the gates $\mathbf{F}_g$ do not form an independent part of the information specifying a quantum algorithm on the $\mathrm{QC}_{\mathcal{C}}$. Instead, they are absorbed into the initialization value $\mathbf{I}_{\mathrm{init}}$ of $\mathbf{I}(t)$.

The measurement bases in which the results s_k are obtained –referred to implicitly in (66) and (67)– are not fixed a priori, but must be determined during the computation. They will be calculated using the byproduct images $\{\mathbf{F}_k, k \in \mathcal{C}\}$ and $\mathbf{I}(t)$, as explained in Sections 4.2 and 4.3. Besides the byproduct images, the algorithm angles $\varphi_{j,\mathrm{algo}}$, $j \in Q^{(1)}$ are also needed to determine the appropriate measurement bases. They are related to the network angles $\varphi_{j,\mathrm{qln}}$, $j \in Q^{(1)}$ that specify the one-qubit rotations in the corresponding quantum logic network via

$$\varphi_{j,\mathrm{algo}} = (-1)^{\eta_j} \, \varphi_{j,\mathrm{qln}}, \quad j \in Q^{(1)}, \tag{71}$$

[‡] The way we use the term "algorithmic information" has nothing to do with the –in general non-computable– algorithmic information content of an object as it is defined in Kolmogorov complexity theory [14].

where η_j is given by

$$\eta_j = \sum_{k|\ \substack{k \in \mathcal{C}\backslash Q_{0,z}\,, \\ j \in \mathrm{bc}(k)}} \kappa_k + \sum_{g|\ \substack{g \in \mathcal{N}\,, \\ j \in \mathrm{bc}(g)}} 1. \tag{72}$$

The pair of equations (71), (72) is, for the moment, just a definition of the algorithm angles. It will become apparent in Sections 4.2 and 4.3 that this definition is indeed useful.

4.2. *Description of the model*

As already listed in Section 3.5, a quantum algorithm on the $\mathrm{QC}_{\mathcal{C}}$ is specified by the sets Q_t of simultaneously measured qubits, the backward cones $\mathrm{bc}(k)$ of the qubits $k \in Q_0$, the measurement bases of the qubits $k \in Q_0$, the byproduct images $\mathbf{F}_j$ for $j \in \mathcal{C}$, the algorithm angles $\varphi_{l,\mathrm{algo}}$ for $l \in Q^{(1)}$ and the initialization value $\mathbf{I}_{\mathrm{init}}$ of the information flow vector $\mathbf{I}(t)$. If an algorithm is not given in this form but rather as a quantum logic network composed of CNOT-gates and one-qubit rotations, the above quantities can be derived from the network as explained in the previous sections.

Let us summarize this step of classical pre-processing. First, the measurement pattern is obtained –if one has no better idea– by patching together the measurement patterns for the individual gates displayed in Fig. 1. This gives the measurement directions for the qubits $k \in Q_0$. The network angles $\varphi_{j,\mathrm{qln}}$ for the qubits $j \in Q^{(1)}$ are taken from the quantum logic network to be simulated. To determine the sets $\{Q_t, t = 0..t_{\max}\}$, we need the forward cones. The forward cones $\mathrm{fc}(k)$ for all qubits $k \in \mathcal{C}$ can be obtained using the expressions (8), (10) for the byproduct operators and the propagation relations (11), (12) and (14). From the forward cones we derive a strict partial ordering "$\prec$" (17) among the cluster qubits, and from the strict partial ordering we derive the sets $Q_t \subset \mathcal{C}$ via (20). The byproduct images $\mathbf{F}_k$ for the qubits $k \in \mathcal{C}\backslash Q_{0,z}$ are obtained from their definition (27) once the corresponding forward propagated byproduct operators are obtained from (49). The byproduct images of the qubits $k \in Q_{0,z}\backslash O$ are traced back to those in the set $\mathcal{C}\backslash Q_{0,z}$ via eq. (60). The byproduct images of the remaining qubits in $\mathcal{C}$, $k \in O$, are given by (64). To determine the algorithm angles we need the backward cones $\mathrm{bc}(k)$ for the qubits $k \in Q_0$ and the backward cones of gates $\mathrm{bc}(g)$. Then, the algorithm angles are given by (71), (72). Finally, for the initialization value $\mathbf{I}_{\mathrm{init}}$ of the information flow vector we need the byproduct images $\mathbf{F}_g$ of the gates g which we obtain from eq. (28). $\mathbf{I}_{\mathrm{init}}$ is set via (69). All the pre-processing required to extract the listed quantities from a quantum logic network can be performed efficiently on a classical computer, see Appendix C.

The scheme of quantum computation on the $\mathrm{QC}_{\mathcal{C}}$ comprises several measurement rounds in which the following steps have to be performed:

1. First measurement round.

   (a) Measure all qubits $k \in Q_0$. Obtain measurement results $\{s_k | k \in Q_0\}$.

   (b) Modify the angles $\varphi_{j,\mathrm{algo}}$ for the continuous gates

$$\varphi_{j,\mathrm{algo}} \longrightarrow \varphi'_{j,\mathrm{algo}} = \varphi_{j,\mathrm{algo}}\,(-1)^{\eta'_j}, \tag{73}$$

with

$$\eta'_j = \sum_{k \in Q_0 \,|\, j \in \mathrm{bc}(k)} s_k \tag{74}$$

for all $j \in Q^{(1)}$.

(c) Update the information flow vector from $\mathbf{I}_{\mathrm{init}}$ to $\mathbf{I}(0)$

$$\mathbf{I}(0) = \mathbf{I}_{\mathrm{init}} + \sum_{k \in Q_0} s_k \mathbf{F}_k. \tag{75}$$

2. Subsequent measurement rounds.

Perform the following three steps (2a) - (2c) for all qubit sets $Q_t \subset \mathcal{C} \backslash Q_0$ in ascending order, beginning with Q_1. In the measurement round t,

(a) Determine the measurement bases for $j \in Q_t$ according to

$$\varphi_{j,\mathrm{meas}} = \varphi'_{j,\mathrm{algo}} \, (-1)^{(\mathbf{I}(t-1), \mathbf{F}_j)_S} \tag{76}$$

(b) Perform the measurements on the qubits $j \in Q_t$. Thereby obtain the measurement results $\{s_j \in \{0,1\} \mid j \in Q_t\}$.

(c) Update the information flow vector $\mathbf{I}$

$$\mathbf{I}(t) = \mathbf{I}(t-1) + \sum_{j \in Q_t} s_j \, \mathbf{F}_j. \tag{77}$$

The information flow vector $\mathbf{I}(t_{\max})$ after the final measurement round $t_{\max}$ equals the information vector $\mathbf{I}$, as can be seen from (68). At the end of the computation, from $\mathbf{I}$ we can directly read off the result $\mathbf{I}_x$ of the computation. $\mathbf{I}_x$ is identical to the readout of the corresponding quantum logic network.

Remark 1. Note that in the first measurement round the byproduct operators created by the measurements on qubits in Q_0 have been propagated *backwards* to set the angles $\{\varphi'_{j,\mathrm{algo}}\}$. There is also a scheme in which the byproduct operators caused by the measurements in the initialization round are propagated forward to set the modified algorithm angles $\{\varphi'_{j\,\mathrm{algo}}\}$. In that scheme, the update of the information flow vector $\mathbf{I}(t)$ and the rule to determine the measurement angles $\varphi_{j,\mathrm{meas}}$ are the same as in the described scheme, given by (76) and (77). What is different is the initialization and the appearance of a step of post-processing. In the modified scheme, in eqs. (72) and (74) the backward cones $\mathrm{bc}(k)$ are replaced by the respective forward cones $\mathrm{fc}(k)$ and $\mathbf{I}_{\mathrm{init}}$ is set to zero. The quantity which was $\mathbf{I}_{\mathrm{init}}$ in (69) is computed as well but now stored as an auxiliary quantity $\Delta\mathbf{I}$ until the end of the computation. After the last measurement round $t_{\max}$, the information vector $\mathbf{I}$ then is obtained by the relation $\mathbf{I} = \mathbf{I}(t_{\max}) + \Delta\mathbf{I}$, which requires the extra post-processing step and extra memory during the computation. We have chosen to present the scheme with backward propagation of byproduct operators in order to avoid this superfluous post-processing. This way, the quantity $\mathbf{I}(t)$ which steers the computational process directly displays the result of the computation after the final update to $\mathbf{I}(t_{\max})$.

Remark 2. This comment concerns the choice $\mathcal{O} = \Omega$ of the cut on which the byproduct images $\mathbf{F}_k$ and $\mathbf{F}_g$ are evaluated. In the visualization of the $\mathrm{QC}_\mathcal{C}$ as an implementation of a quantum logic network the cut Ω plays a distinguished role. The byproduct operators accumulated at Ω determine how the "readout" measurements have to be interpreted. In the computational model underlying the $\mathrm{QC}_\mathcal{C}$, however, the former readout qubits are just qubits to be measured like any other cluster qubits. Here, the cut Ω is not distinguished. Due to the invariance (37) of the symplectic scalar product (36) the byproduct images $\mathbf{F}_k$, which enter the expression (76) for the $\varphi_{k,\mathrm{meas}}$ directly and via (69) and (77), can be evaluated with respect to *any* vertical cut $\mathcal{O}$ through the corresponding quantum logic network. The information vector $\mathbf{I}$ which displays the result of the computation in its x-part $\mathbf{I}_x$ would then be related to the information flow vector after the final measurement round $\mathbf{I}(t_{\max})$ via $\mathbf{I} = C(\mathcal{N}_{\mathcal{O}\to\Omega})\,\mathbf{I}(t_{\max})$. Thus, the particular vertical cut $\mathcal{O} = \Omega$ was chosen just to avoid an additional step of post-processing. The dependence on the cut $\mathcal{O}$ would vanish altogether if one would write the n output bits of the quantum computation in the form $[I_x]_i = (\mathbf{I}|_\mathcal{O}, f_i|_\mathcal{O})_S$ for suitably chosen $\{f_i \in \mathcal{V}, i = 1,..,n\}$, e.g. for the case $\mathcal{O} = \Omega$, $f_1 = (0,..,0;1,0,..,0)^T$, $f_2 = (0,..,0;0,1,0,..,0)^T$, and the other f_i, $i \leq n$ accordingly.

4.3. *Proof of the model*

In this section it is shown that if we run the $\mathrm{QC}_\mathcal{C}$ according to the scheme described in Section 4.2, we obtain the same result as in the corresponding quantum logic network. This requires to prove that (a) one does indeed choose all the measurement angles correctly and (b) obtains at the end of the computation the result $\mathbf{I}_x$, the x-part of the information vector $\mathbf{I}$ as given in (66).

To show point (b), we use (69), (75) and (77) and obtain for the information vector

$$\mathbf{I} = \sum_{k\in\mathcal{C}\setminus Q_{0,z}} \kappa_k \mathbf{F}_k + \sum_{g\in\mathcal{N}} \mathbf{F}_g + \sum_{k\in Q_0} s_k \mathbf{F}_k + \sum_{k\in\bigcup_{i=1}^{t_{\max}} Q_i} s_k \mathbf{F}_k$$

which coincides with (66). This ensures that we obtain the right vector $\mathbf{I}$ at the end of the computation, provided the measurement bases were chosen appropriately, as required for (a). This is checked below.

First we observe that the measurement angle $\varphi_{j,\mathrm{meas}}$ and the network angle $\varphi_{j,\mathrm{qln}}$ are for all $j \in Q^{(1)}$ related in the following way

$$\varphi_{j,\mathrm{meas}} = (-1)^{\vartheta_j}\,\varphi_{j,\mathrm{qln}}\,, \tag{78}$$

with

$$\vartheta_j = \sum_{k\in\mathcal{C}|j\in\mathrm{fc}(k)} s_k + \sum_{k\in\mathcal{C}\setminus Q_{0,z}|j\in\mathrm{fc}(k)} \kappa_k + \sum_{g\in\mathcal{N}|j\in\mathrm{fc}(g)} 1 \mod 2. \tag{79}$$

Why does the pair of equations (78), (79) hold? As can be seen from the propagation relation for rotations (11), the network and the measurement angle of a qubit $j \in Q^{(1)}$ can differ only by a sign factor ± 1 and can therefore always be related as in (78). The first and the third sum in (79) follow from the definition of the forward cones of the cluster qubits and of gates in Section 3.2 . The measurement angle at j acquires a factor $(-1)^{s_k}$ if $j \in \mathrm{fc}(k)$ and a factor of (-1) for each gate g with $j \in \mathrm{fc}(g)$. The κ-dependent part can be derived in the same way

as it was derived for U_Σ through equations (52) - (62). Note that ϑ_j depends, similar to the information vector $\mathbf{I}$, only on the combination $s_k + \kappa_k$ for $k \in \mathcal{C}\backslash Q_{0,z}$ and on s_k for $k \in Q_{0,z}$.

Now, we rewrite the quantity ϑ_j in the following way

$$\begin{aligned}
\vartheta_j &= \sum_{k\in\mathcal{C}|j\in\mathrm{fc}(k)} s_k + \sum_{k\in\mathcal{C}\backslash Q_{0,z}|j\in\mathrm{fc}(k)} \kappa_k + \sum_{g\in\mathcal{N}|j\in\mathrm{fc}(g)} 1 \mod 2 \\
&= \sum_{k\in\mathcal{C}|j\in\mathrm{fc}(k)} s_k + \sum_{k\in\mathcal{C}\backslash Q_{0,z}|j\in\mathrm{fc}(k)} \kappa_k + \sum_{g\in\mathcal{N}|j\in\mathrm{fc}(g)} 1 + \\
&\quad +2\left(\sum_{k\in Q_0|j\in\mathrm{bc}(k)} s_k + \sum_{k\in\mathcal{C}\backslash Q_{0,z}|j\in\mathrm{bc}(k)} \kappa_k + \sum_{g\in\mathcal{N}|j\in\mathrm{bc}(g)} 1\right) \mod 2 \\
&= \underbrace{\sum_{k\in Q_0|j\in\mathrm{fc}(k)\vee j\in\mathrm{bc}(k)} s_k}_{S_1} + \underbrace{\sum_{k\in Q^{(1)}|j\in\mathrm{fc}(k)} s_k}_{S_2} + \underbrace{\sum_{k\in Q_0|j\in\mathrm{bc}(k)} s_k}_{S_3} + \\
&\quad + \underbrace{\sum_{k\in\mathcal{C}\backslash Q_{0,z}|j\in\mathrm{fc}(k)\vee j\in\mathrm{bc}(k)} \kappa_k}_{S_4} + \underbrace{\sum_{k\in\mathcal{C}\backslash Q_{0,z}|j\in\mathrm{bc}(k)} \kappa_k}_{S_5} + \\
&\quad + \underbrace{\sum_{g\in\mathcal{N}|j\in\mathrm{fc}(g)\vee j\in\mathrm{bc}(g)} 1}_{S_6} + \underbrace{\sum_{g\in\mathcal{N}|j\in\mathrm{bc}(g)} 1}_{S_7} \mod 2.
\end{aligned} \tag{80}$$

We now discuss the seven terms $S_1, .., S_7$. All sums are evaluated modulo 2.

Term S_1 of (80):

$$S_1 = \sum_{k\in Q_0|j\in\mathrm{fc}(k)\vee j\in\mathrm{bc}(k)} s_k = \sum_{k\in Q_0} s_k\,(\mathbf{F}_k, \mathbf{F}_j)_S, \tag{81}$$

where the last identity holds by the cone test (41).

Term S_2 of (80):
Let be $j \in Q_t$ and $k \in Q_i$. Qubit j can only then be in the forward cone of k, $j \in \mathrm{fc}(k)$, if $i < t$. Hence

$$\begin{aligned}
S_2 &= \sum_{k\in Q^{(1)}\,|\,j\in\mathrm{fc}(k)} s_k \\
&= \sum_{k\in\bigcup_{i=1}^{t-1} Q_i|j\in\mathrm{fc}(k)} s_k \\
&= \sum_{k\in\bigcup_{i=1}^{t-1} Q_i} s_k\,(\mathbf{F}_k, \mathbf{F}_j)_S.
\end{aligned} \tag{82}$$

In (82) the last line again follows by using the cone test (41).

Term S_3 of (80):

$$S_3 = \sum_{k \in Q_0 | j \in \mathrm{bc(k)}} s_k = \eta_j'. \tag{83}$$

This equity follows by the definition of η_j' in (74). Thus, the term S_3 is the contribution to ϑ_j coming from the first measurement round where the algorithm angles $\{\varphi_{j,\mathrm{algo}}\}$ are changed to the modified algorithm angles $\{\varphi'_{j,\mathrm{algo}}\}$.

Term S_4 of (80):

$$S_4 = \sum_{k \in \mathcal{C} \setminus Q_{0,z} | j \in \mathrm{fc}(k) \vee j \in \mathrm{bc}(k)} \kappa_k = \sum_{k \in \mathcal{C} \setminus Q_{0,z}} \kappa_k \left(\mathbf{F}_k, \mathbf{F}_j\right)_S , \tag{84}$$

which follows by the cone test (41).

Terms $S_5 + S_7$ of (80):

$$S_5 + S_7 = \sum_{k \in \mathcal{C} \setminus Q_{0,z} | j \in \mathrm{bc}(k)} \kappa_k \; + \sum_{g \in \mathcal{N} | j \in \mathrm{bc}(g)} 1 = \eta_j , \tag{85}$$

via the definition (72) of the η_j.

Finally, term S_6 of (80):

$$S_6 = \sum_{g \in \mathcal{N} | j \in \mathrm{fc}(g) \vee j \in \mathrm{bc}(g)} 1 = \sum_{g \in \mathcal{N}} \left(\mathbf{F}_g, \mathbf{F}_j\right)_S , \tag{86}$$

which follows by the cone criterion (43) for gates.

Now we combine these seven terms $S_1, .., S_7$. By (80) - (86) we obtain

$$\begin{aligned} \vartheta_j &= \eta_j + \eta_j' + \sum_{g \in \mathcal{N}} \left(\mathbf{F}_g, \mathbf{F}_j\right)_S \; + \sum_{k \in \mathcal{C} \setminus Q_0} \kappa_k \left(\mathbf{F}_k, \mathbf{F}_j\right)_S \; + \sum_{k \in \bigcup_{i=0}^{t-1} Q_i} s_k \left(\mathbf{F}_k, \mathbf{F}_j\right)_S \\ &= \eta_j + \eta_j' + \left(\mathbf{I}(t-1), \mathbf{F}_j\right)_S . \end{aligned} \tag{87}$$

The last line follows from the definition (67) of the information flow vector. If we consider the relations (71), (73) and (78) between the angles $\varphi_{j,\mathrm{algo}}$, $\varphi'_{j,\mathrm{algo}}$ and $\varphi_{j,\mathrm{meas}}$, we find

$$\varphi_{j,\mathrm{meas}} = (-1)^{\vartheta_j - \eta_j - \eta_j'} \varphi'_{j,\mathrm{algo}}. \tag{88}$$

Now we insert (87) into (88) and obtain

$$\varphi_{j,\mathrm{meas}} \;=\; \varphi'_{j,\mathrm{algo}} \, (-1)^{\left(\mathbf{I}(t-1), \mathbf{F}_j\right)_S},$$

which proofs that the assignment of the measurement angles (76) is correct, and thereby concludes the proof of the computational model described in Section 4.2.

5. Logical depth and temporal complexity

The logical depth has, to our knowledge, only been defined in the context of quantum logic networks, but it can straightforwardly be generalized to the $\mathrm{QC}_{\mathcal{C}}$. In networks one groups

gates which can be performed in parallel to layers. The logical depth of a quantum logic network then is the minimum number of its layers. Similarly in case of the $\mathrm{QC}_{\mathcal{C}}$, one can group the cluster qubits which can be measured simultaneously to sets Q_t. There, the logical depth of the $\mathrm{QC}_{\mathcal{C}}$-realization of an algorithm is the minimal number of such sets.

Since the one-qubit measurements on the cluster state mutually commute, one may be led to think that they can always be performed all in parallel. They could, but then the measurements would in general not drive a deterministic computation.

In the following, we will denote the logical depth in the context of the $\mathrm{QC}_{\mathcal{C}}$ by D and the logical depth of a quantum logic network by $D_{\mathcal{N}}$.

5.1. $D = 1$ *for circuits in the Clifford group*

The Clifford group of gates is generated by the CNOT-gates, the Hadamard-gates and the $\pi/2$-phase shifts. In this section it is proved that the logical depth of such circuits is $D = 1$ on the $\mathrm{QC}_{\mathcal{C}}$, independent of the number of logical qubits n. For a subgroup of the Clifford group, the group generated by the CNOT- and Hadamard gates alone we can compare the result to the best known upper bound for quantum logic networks, where the logical depth $D_{\mathcal{N}}$ scales like $O(\log n)$ [13].

The elementary gates we use are the Hadamard gate $H = 1/\sqrt{2}(\sigma_x + \sigma_z)$, the $\pi/2$-phase gate $U_z(\pi/2) = \exp(-i\pi/4\,\sigma_z)$, and the CNOT-gate between neighbouring qubits, whose realization on the $\mathrm{QC}_{\mathcal{C}}$ is depicted in Fig. 1. Out of the latter we construct the CNOT-gate between arbitrary qubits via the swap-gate composed of three CNOT-gates. Hence, out of the elements displayed in Fig. 1 any circuit in the Clifford group can be composed. At this point we must emphasize that in a practical realization of a $\mathrm{QC}_{\mathcal{C}}$ we would not perform a general CNOT in the described manner using the swap gate. There is a more efficient realization for the general CNOT, whose spatial resources scale more favourably. This gate will be displayed elsewhere. The purpose here is to keep the argument compact rather than the gates.

It is possible to measure all qubits at once. This works since, as shown in Fig. 1, all cluster qubits necessary for the realization of the CNOT-, Hadamard- and $\pi/2$-phase gates are measured either in the eigenbasis of σ_x or of σ_y. The redundant qubits are measured in σ_z, as explained in [9]. Thus, none of the qubits is measured in a basis whose proper adjustment requires classical information from measurement results at other qubit sites. This concludes the proof of $D = 1$ for circuits in the Clifford group. In a computation, of course all the measurement results obtained must be interpreted. Therefore, there exists a contribution to the computation time from classical post-processing. The connection between logical depth and computation time is discussed in Section 5.3.

5.2. $D = 2$ *for circuits of CNOT-gates and a* $U(1)^{\otimes n}$*-subgroup of rotations*

In this section we prove that the logical depth D of a circuit composed of either CNOT-gates and rotations about the x-axis or of CNOT-gates and rotations about the z-axis is $D = 2$. This set of circuits contains all circuits of diagonal 2-qubit gates as a special case. For circuits of diagonal 2-qubit gates we can compare our result $D = 2$ to the best known result [13] for quantum logic networks where the logical depth scales logarithmically in the number of gates.

Here we give the proof for circuits of CNOT-gates and rotations about the z-axis $U_z(\alpha) =$

$e^{-i\alpha\frac{\sigma_z}{2}}$. The elementary gates used are (a) the rotations about the z-axis $U_z(\alpha) = e^{-i\alpha\frac{\sigma_z}{2}}$, and (b) the CNOT-gate between neighbouring logical qubits. The realization of the rotation U_z is depicted in Fig. 1. Of the CNOT-gate between neighbouring qubits we construct the swap-gate between neighbouring qubits and by that the general CNOT-gate, as in Section 5.1. The strategy to implement the circuit is then: (1) to measure all those qubits on $\mathcal{C}$ which are to be measured in the eigenbases of σ_x, σ_z or σ_y; and (2) to measure the remaining qubits, i.e. the ones which are measured in a direction in the $x-y$-plane.

The result that the measurements in step (1) can be performed in one step has already been shown in Section 5.1. It remains to be shown that the measurements in the tilted measurement directions of step (2) can also be performed in parallel. Let j and l be two cluster qubits which are measured in a tilted basis in step 2 in order to implement the rotations. Using (20) one finds

$$D > 2 \Longrightarrow \exists\, j, l \in Q^{(1)} :\ l \prec j \ \ (\text{that is, } Q^{(2)} \neq \emptyset). \tag{89}$$

Further holds

$$l \prec j \Longrightarrow \exists\, k \in Q^{(1)} :\ j \in \mathrm{fc}(k), \tag{90}$$

because the strict partial ordering "$\prec$" is generated by the forward cones, i.e. $l \prec j \Longleftrightarrow$ either $j \in \mathrm{fc}(l)$, or $\exists (k_1, \ldots k_r) :\ k_1 \in \mathrm{fc}(l) \ \wedge\ \left\{k_s \in \mathrm{fc}(k_{s-1}) | \, 2 \leq s \leq r\right\} \ \wedge\ j \in \mathrm{fc}(k_r)$.

Moreover, from the criterion (41) one derives

$$j \in \mathrm{fc}(k) \Longrightarrow (\mathbf{F}_j, \mathbf{F}_k)_S = 1. \tag{91}$$

Now, by putting the implications (89), (90) and (91) together we obtain

$$D > 2 \Longrightarrow \exists\, j, k \in Q^{(1)} :\ (\mathbf{F}_j, \mathbf{F}_k)_S = 1, \tag{92}$$

which we negate to obtain

$$\forall j, k \in Q^{(1)} : (\mathbf{F}_j, \mathbf{F}_k)_S = 0 \Longrightarrow D \leq 2. \tag{93}$$

Next it is proved that $(\mathbf{F}_j, \mathbf{F}_k)_S = 0$ does indeed hold for all $j, k \in Q^{(1)}$.

A measurement of a qubit at site k , which is part of the implementation of a rotation about the z-axis (central qubit 3 in Fig. 1c), generates a byproduct operator $(U_k)^{s_k} = (\sigma_z)^{s_k}$. This can be seen from equations (3), (7) and (8). Note that in Fig. 1c, qubits 1,2,4 are measured in the σ_x-eigenbasis, they belong to the set Q_0. Now let be i the number of the *logical* qubit on which the rotation $U_z(\varphi_k)$ is performed by the measurement of cluster qubit k. Further, let $\mathcal{O}$ be a vertical cut through the network simulated by the $\mathrm{QC}_{\mathcal{C}}$. $\mathcal{O}$ intersects each qubit line only once. In particular, it shall intersect the qubit line i just at the output side of the rotation $U_z(\varphi_k)$. Thus, the image $\mathbf{F}_k|_{\mathcal{O}}$ of U_k on the cut $\mathcal{O}$ is

$$\mathbf{F}_k|_{\mathcal{O}} = \begin{pmatrix} 0 \\ \\ \mathbf{F}_{kz}|_{\mathcal{O}} \end{pmatrix}, \quad \text{with } F_{kz,l} = \delta_{il}. \tag{94}$$

What we see from (94) is that $\mathbf{F}_{kx}|_{\mathcal{O}} = 0$. Be $\mathcal{N}_{\mathcal{O}\to\Omega}$ the part of the network $\mathcal{N}$ which is located between the two cuts $\mathcal{O}$ and Ω. The byproduct image $\mathbf{F}_k$ corresponding to U_k is then

given by

$$\mathbf{F}_k \equiv \mathbf{F}_k|_\Omega = C(\mathcal{N}_{\mathcal{O}\to\Omega})\,\mathbf{F}_k|_{\mathcal{O}}. \tag{95}$$

The only gates that contribute to $C(\mathcal{N}_{\mathcal{O}\to\Omega})$ are the CNOT gates, as described in section 3.4.3. The propagation matrices for CNOT gates (35) have block-diagonal form. Hence, using (30) the propagation matrix for the network $\mathcal{N}_{\mathcal{O}\to\Omega}$ has block-diagonal form

$$C(\mathcal{N}_{\mathcal{O}\to\Omega}) = \left(\begin{array}{c|c} C_{xx}(\mathcal{N}_{\mathcal{O}\to\Omega}) & 0 \\ \hline 0 & C_{zz}(\mathcal{N}_{\mathcal{O}\to\Omega}) \end{array}\right). \tag{96}$$

From (94), (95) and (96) it follows that the x-part of the byproduct image vector $\mathbf{F}_k$ vanishes for all k

$$[\mathbf{F}_x]_k = 0 \;\; \forall k \in Q^{(1)}. \tag{97}$$

Hence by the definition of the symplectic scalar product (36), we obtain $(\mathbf{F}_j, \mathbf{F}_k)_S = 0$ for all $j, k \in Q^{(1)}$. This proves via (93) $D \leq 2$. The measurements to implement the one-qubit rotations can thus all be performed at the same time. In (93) the case $D = 1$ can be easily be excluded for all interesting cases such that only $D = 2$ remains. This concludes the proof of $D = 2$ for circuits of CNOT-gates and rotations of the form $e^{-i\varphi\frac{\sigma_z}{2}}$. The proof for circuits of CNOT-gates and rotations $e^{-i\varphi\frac{\sigma_x}{2}}$ runs analogously.

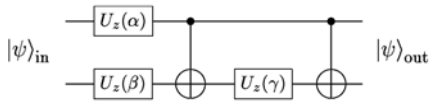

Fig. 4. Network for a diagonal gate composed of z-rotations and CNOT-gate.

Now let us discuss the special case of circuits composed of diagonal two-qubit gates. A diagonal gate G_d in the computational basis is of the form

$$G_d = \begin{pmatrix} e^{i\varphi_1} & & & \\ & e^{i\varphi_2} & & \\ & & e^{i\varphi_3} & \\ & & & 1 \end{pmatrix}, \tag{98}$$

modulo a possible global phase which is not relevant.

The network of rotations about the z-axis and of a CNOT-gate shown in Fig. 4 realizes a general diagonal two-qubit gate. In order to obtain the angles φ_1, φ_2 and φ_3 specifying the diagonal gate G_d in (98), one chooses the following angles for the three z-rotations in this

network

$$\begin{aligned} \alpha &= \frac{1}{2}(-\varphi_1 - \varphi_2 + \varphi_3), \\ \beta &= \frac{1}{2}(-\varphi_1 + \varphi_2 - \varphi_3), \\ \gamma &= \frac{1}{2}(-\varphi_1 + \varphi_2 + \varphi_3). \end{aligned} \tag{99}$$

Thus, a circuit of diagonal two-qubit gates can also be regarded as a circuit of z-rotations and CNOT-gates. Therefore we find $D = 2$ for circuits of diagonal two-qubit gates on the $\mathrm{QC}_{\mathcal{C}}$. This result can be compared to the best known upper bound [13] for quantum logic networks where the logical depth is of the order $\mathcal{O}(\log n_G)$ with n_G the number of two-qubit gates.

5.3. *The logical depth D is a good measure for temporal complexity*

In this section, we will express the computation time as a function of the logical depth. The computational model described in section 4.2 consists of an alternating series of measurement rounds and classical processing of the thereby obtained measurement results. The classical processing contributes to the duration of the computation and will therefore enter into the relation between the computation time and the logical depth. For the computation time, this results in a correction logarithmic in the number n of logical qubits involved, and thus the computation time is no longer the logical depth times a constant. For all practical purposes, however, this logarithmic correction is small compared to the time required for the genuine quantum part of the computation, consisting of the measurements.

Let Δ_Q be the time required to perform the simultaneous measurements in one measurement round and Δ_{cl} the time required for the elementary steps of classical processing: say, addition modulo 2 or multiplication of two bits. The time $T_{cl}(t)$ required for classical processing after each measurement round has two contributions. First, the time $T_{cl,\mathbf{I}}(t)$ to update the information flow vector $\mathbf{I}(t)$ and second, the time $T_{cl,\pm}(t)$ to determine the signs of the measurement angles of all measurements in the next round. The total computation time T_{comp} is given by

$$T_{\mathrm{comp}} = D\Delta_Q + \sum_{t=0}^{D-1} T_{cl,\mathbf{I}}(t) + T_{cl,\pm}(t) \tag{100}$$

The update of the information vector $\mathbf{I}(t)$ according to (77) can be done for all $2n$ components in parallel. The update $\mathbf{I}(t-1) \longrightarrow \mathbf{I}(t)$ following measurement round t requires the time that it takes to add up $\|Q_t\|$ bits modulo 2. As Fig. 5 illustrates, $T_{cl,\mathbf{I}}(t)$ is logarithmic in $\|Q_t\|$. The number of qubits in the set Q_t is bounded from above by $\|\mathcal{C}\|$ since $Q_t \subset \mathcal{C}$. Here, $\mathcal{C}$ is any cluster sufficiently large to carry the network to be simulated. Thus

$$T_{cl,\mathbf{I}}(t) \leq \Delta_{cl} \log \|\mathcal{C}\|. \tag{101}$$

To determine the proper measurement angle $\varphi_{k,\mathrm{meas}}$ for the measurement on qubit $k \in Q_{t+1}$ in the next measurement round requires, according to (76), the evaluation of the symplectic scalar product $(\mathbf{I}(t), F_k)_S$. This requires 1 step for multiplication and $\log 2n$ steps for addition modulo 2. Thus,

$$T_{cl,\pm} = \Delta_{cl}\,(\log n + 2)\,. \tag{102}$$

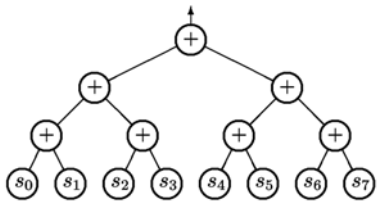

Fig. 5. Tree for mod 2-adding m binary numbers in $\approx \log(m)$ steps.

Combining (100), (101) and (102), the total computation time T_{comp} is bounded from above by

$$T_{\text{comp}} \leq D\,\Delta_Q \left(1 + \frac{\Delta_{cl}}{\Delta_Q}\left[\log \|\mathcal{C}\| + \log n + 2\right]\right). \tag{103}$$

We see that, although the computation time T_{comp} is linear in the logical depth D, it contains contributions logarithmic in the number n of logical qubits and in the cluster size $\|\mathcal{C}\|$. These logarithmic contributions are, however, suppressed by the ratio between the characteristic time for classical processing and the characteristic time for the von-Neumann measurements, Δ_{cl}/Δ_Q. This ratio can, in practice, be very small. Therefore, the logarithmic corrections become important only in the limit of large clusters and large n. As will be argued below, even in the regimes where a quantum computer is believed to become useful, say $n \approx 10^5$, the logarithmic corrections have only a minor influence on the total computation time.

We now eliminate the dependence of the total computation time on the cluster size $\|\mathcal{C}\|$. For this we assume that on the $\text{QC}_{\mathcal{C}}$ we simulate a quantum logic network with the network logical depth $D_{\mathcal{N}}$. Now, we give an upper bound on $\|\mathcal{C}\|$ as a function of n and $D_{\mathcal{N}}$. As displayed in Fig. 1, a single CNOT gate has height 3 and width 6 on the cluster $\mathcal{C}$. Here we do not count the output qubits of the gates since they also form the input qubits of the gates in the next slice. As in Fig. 1, the rotation has height 1 and width 4, if the output qubit is again not counted for the width. The wires for the logical qubits on the cluster can be arranged with distance 2. Each set of parallelized gates will at most require a slice of width 6 on the cluster. The circuit as a whole requires an additional slice of width 1 for the output. A swap-gate that is composed of three CNOT-gates, requires an array of 3×18 qubits on the cluster. If a general CNOT-gate on the cluster were composed of a next-neighbour CNOT-gate and swap-gates (in practice it is not), then it would require at most an array of [height] $\times$ [width] $=[2n-1] \times [18(n-2)+6]$ qubits. Hence, a CNOT gate would, to leading order, consume at most $36n^2$ cluster qubits. Each rotation would require at most –in the worst case where on the network it could not be performed in parallel with other gates– a slice of width 4 on the cluster, so it consumes, to leading order, at most $8n$ cluster qubits. The total number of gates in the network is bounded from above by $n\,D_{\mathcal{N}}$. The simulation of each gate costs at most $\max(36n^2, 8n) = 36n^2$ cluster qubits. Hence, the size of the required cluster is bounded by

$$\|\mathcal{C}\| \leq 36n^3 D_{\mathcal{N}}. \tag{104}$$

If we now use the assumption about a good quantum algorithm that the logical depth scales polynomially in the number of qubits n,

$$D_{\mathcal{N}} = c\, n^p, \tag{105}$$

and insert (104) and (105) into (103), we obtain

$$T_{\text{comp}} \leq D\, \Delta_Q \left(\left[1 + \frac{\Delta_{cl}}{\Delta_Q}(4 + \log 9c) \right] + \frac{\Delta_{cl}}{\Delta_Q}(p+4) \log n \right). \tag{106}$$

From a practical point of view, we find that the logarithmic corrections –even for numbers n of logical qubits in the range of 10^5– play a minor role since they are suppressed by the ratio Δ_{cl}/Δ_Q. We could plug in some typical numbers, say $\Delta_Q = 1\ \mu$s, $\Delta_{cl} = 1$ ns, $p = 3$ and $n = 10^5$, to obtain $\Delta_{cl}/\Delta_Q\,(p+4)\log n \approx 0.12$ (or ≈ 0.24 for $n = 10^{10}$).

The spatial overhead $\|\mathcal{C}\|$ is polynomial in the number n of logical qubits. But, if one adopts this more practical viewpoint one may not be satisfied by the mere result that the spatial overhead scales polynomially, but might want to know what the scaling power actually is. Above we found that $\|\mathcal{C}\|$ scales with the $(p+3)$th power of n. However, in the above argument, we focused on the computation time where the precise value of exponent for the spatial scaling did not play an important role, and thus have been extremely wasteful with spatial resources. A more careful discussion yields a more favourable scaling of the spatial overhead.

From a strict scaling point of view, we find in (106) that the computation time is no longer equal to the logical depth D times a constant, but there are $\log n$ -corrections due to the classical processing. This is, as the above numbers illustrate, of little relevance for practical purposes. The classical processing can be parallelized to such a degree that it increases the total computation time only marginally.

6. Discussion

The discussion about the logical depth of certain algorithms with the $\text{QC}_{\mathcal{C}}$ in Section 5 showed that there are ways of information processing possible with the $\text{QC}_{\mathcal{C}}$ which cannot be explained adequately in network model terms. This made a computational model appropriately describing the $\text{QC}_{\mathcal{C}}$ desirable. The computational model underlying the $\text{QC}_{\mathcal{C}}$ that we found does not seem to have much in common with the network model. It is based on objects of a different sort which require an interpretation. In this section, we attempt to clarify the role of the binary valued information flow vector $\mathbf{I}(t)$ and that of the stepwise measured quantum state.

What is the role of the information flow vector $\mathbf{I}(t)$? In every computational step except the final one the information flow vector $\mathbf{I}(t)$ is completely random. So one might ask whether it contains information at all. It does, since in every step except the last one it tells what has to be done next. After the final computational step at time $t_{\max}$ the quantity $\mathbf{I}(t_{\max})$ contains the result of the computation. Thus, the quantity $\mathbf{I}(t)$ has a meaning in every computational step. In this sense, it represents the algorithmic information in the described scheme of quantum computation.

What is the role of the stepwise measured quantum state? To see that explicitly, let us consider the scenario where a quantum computation is halted in the middle and continued

at a later time by another person who only knows which steps of the computation are left to perform but does not know what has been done so far. In analogy to a teleportation protocol where both the result from the Bell measurement and the quantum state at the receivers side are required to reconstruct the initial state, the halted computation can be successfully completed only if both pieces –the intermediate information flow vector $\mathbf{I}(t_i)$ and the half-measured quantum state– are stored until the computation proceeds. Thus, the quantum state cannot be neglected just because it does not appear in the formal description of the computational model. The quantum correlations in the stepwise measured state are what basically enables the described way of quantum information processing. However, the role of this state is a passive one. It serves as a resource that is used up during the course of computation.

For the description of the $\mathrm{QC}_{\mathcal{C}}$ there is no necessity to refer to the "qubit" as a basic notion of quantum information theory. As described above, the stepwise modified quantum state attains the role of a consumable resource but not that of the carrier of algorithmic information. It is thus sufficiently described in terms of standard quantum mechanics, namely as a quantum state of entangled two-level systems. The processed algorithmic information is classical.

Let us at the end of this discussion come back to the role which the randomness of the individual measurement results plays for the $\mathrm{QC}_{\mathcal{C}}$. It may surprise that a set of classical binary numbers represents the algorithmic information in a scheme of quantum computation. In the network model the quantum state (of the quantum register) is usually considered to represent the processed information. For the $\mathrm{QC}_{\mathcal{C}}$, the situation is different. There, the randomness of the individual measurement results makes it necessary to store classical steering information. The need to process this information has called for a novel information carrying quantity. What, in a network-like description of the $\mathrm{QC}_{\mathcal{C}}$, has been regarded as a mere byproduct turns out to be the central quantity of information processing with the $\mathrm{QC}_{\mathcal{C}}$.

7. Conclusion

We have described the computational model underlying the one-way quantum computer, which is very different from the quantum logic network model. The formal description of the $\mathrm{QC}_{\mathcal{C}}$ is based on primitive quantities of which the most important are the sets $Q_t \subset \mathcal{C}$ of cluster qubits defining the temporal ordering of measurements on the cluster state, and the binary valued information flow vector $\mathbf{I}(t)$ which is the carrier of the algorithmic information.

The $\mathrm{QC}_{\mathcal{C}}$ has no quantum input, no quantum output and no quantum register, and the unitary gates from some universal set are not the elementary building blocks of $\mathrm{QC}_{\mathcal{C}}$-quantum algorithms. The quantities that are processed with the $\mathrm{QC}_{\mathcal{C}}$ are the outcomes of one-qubit measurements and thus processing of information exists only at the classical level. The $\mathrm{QC}_{\mathcal{C}}$ is nevertheless quantum mechanical as it uses a highly entangled cluster state as the central physical resource.

As a practical implication of the new model, the logical depth of certain algorithms on the $\mathrm{QC}_{\mathcal{C}}$ is lower than for their corresponding network realizations. As shown for circuits of diagonal gates or of CNOT-, Hadamard- and $\pi/2$-phase gates, the logical depths for the $\mathrm{QC}_{\mathcal{C}}$ are constant in the number of gates and logical qubits, namely one and two. The best bounds for networks that have been known so far scale logarithmically. It therefore seems

that, at least for the $\mathrm{QC}_{\mathcal{C}}$, the question of temporal complexity must be revisited. The tools for this discussion together with first results have been provided in this paper.

Acknowledgements

This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) via the Schwerpunktprogramm QIV. We thank D. E. Browne, O. Forster, D. P. DiVincenzo, D. W. Leung, A. Winter and R. Cleve for helpful discussions.

References

1. D. Deutsch, Proc. R. Soc. London Ser. A **400**, 97 (1985).
2. E. Bernstein and U. Vazirani, Proc. of the 25th Annual ACM Symposium on Theory of Computing, 11 (1993).
3. D. Deutsch, Proc. R. Soc. London Ser. A **425**, 73 (1989).
4. A. Yao, Proc. of the 34th Annual IEEE Symposium of Foundations of Computer Science, 352 (1993).
5. R. B. Griffiths and C.-S. Niu, Phys. Rev. Lett. **76**, 3228 (1996).
6. M. A. Nielsen and I. L. Chuang, Phys. Rev. Lett. **79**, 321 (1997).
7. D. Gottesman and I. L. Chuang, Nature (London) **402**, 390 (1999).
8. E. Knill, R. Laflamme and G. J. Milburn, Nature (London) **409**, 46 (2001).
9. R. Raussendorf and H.-J. Briegel, Phys. Rev. Lett. **86**, 5188 (2001).
10. M. A. Nielsen, quant-ph/0108020 (2001); A. Fenner and Y. Zhang, quant-ph/0111077 (2001); D. W. Leung, quant-ph/0111122 (2001).
11. H.-J. Briegel and R. Raussendorf, Phys. Rev. Lett. **86**, 910 (2001).
12. E. Knill, R. Laflamme and W. H. Zurek, quant-ph/9702058 (1997).
13. C. Moore and M. Nilsson, quant-ph/9808027 (1998).
14. M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications.* Springer (1997).

Appendix A. Proof cone test and forward-backward cone relation

Here we prove the cone test (41) and the forward-backward cone relation (42). Considering the cone test, first note that whether a one-qubit rotation at some position in the network is about the z-axis or a about the x-axis can be identified by the potential byproduct operator produced when the rotation is implemented. This can be seen by inspecting (3), (7), (8) and the procedure to implement a general rotation as described in section 2. The x-rotations $U_x(\xi)$ and $U_x(\zeta)$ of $U_R(\xi,\eta,\zeta)$ in (3) are implemented by measurements on the qubits 2 and 4 of a 5-qubit chain. As can be seen from (8), they contribute to the byproduct operator U_Σ of the rotation U_R with ${\sigma_x}^{s_2+s_4}$ where s_2 and s_4 are the results of the measurements on qubits 2 and 4. Further, the rotation about the z-axis, $U_z(\zeta)$, is implemented by measurement of qubit 3. The contribution to the byproduct operator which is thereby generated is, from (8), ${\sigma_z}^{s_3}$. We see that x-rotations only generate byproduct operators σ_x and z-rotations only generate byproduct operators σ_z.

A byproduct operator generated via the measurement on the cluster qubit k must be propagated either forward or backward to possibly reach the rotation on the logical qubit i implemented via the measurement on the cluster qubit j. Let be $\mathcal{O}_K$ and $\mathcal{O}_J$ two cuts through the network which intersect each logical qubit line only once. More specifically, $\mathcal{O}_K$ intersects

the qubit line i just before the rotation implemented by the measurement at cluster qubit k. $\mathcal{O}_J$ intersects the qubit line i just before the rotation implemented by the measurement at cluster qubit j.

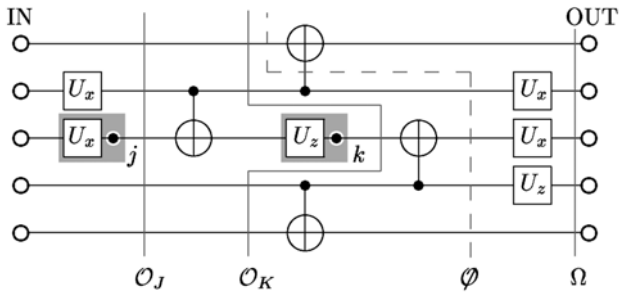

Fig. A.1. Vertical cuts. The vertical cuts intersect each qubit line exactly once but do not intersect gates. Thus, $\mathcal{O}_J$, $\mathcal{O}_J$ and Ω are vertical cuts, but $\not\mathcal{O}$ is not. For two of the rotations in the displayed network, the sub-clusters on which these gates are implemented are symbolically displayed in gray underlay. Via the measurement of the cluster qubits j and k (displayed as black dots with white border), the rotation angles of the respective rotations U_x and U_z are set. For the network shown, $\mathcal{O}_J \leq \mathcal{O}_K$ and also $k \in \mathrm{fc}(j)$.

There are two cases which can occur. Either the cut $\mathcal{O}_K$ is before the cut $\mathcal{O}_J$ in the network $\mathcal{N}$ which we denote by $\mathcal{O}_K \leq \mathcal{O}_J$, or $\mathcal{O}_J$ is before the cut $\mathcal{O}_K$ which we denote by $\mathcal{O}_J \leq \mathcal{O}_K$. It can also be that both is true at the same time but it cannot be that neither of the two relations hold.

Case I: $\mathcal{O}_K \leq \mathcal{O}_J$.
The byproduct operator generated via the measurement at qubit k must be propagated forward to possibly affect the measurement at qubit j. It is not possible that the result of the measurement on qubit j has an effect on the measurement basis chosen at k.

Let us introduce a further cut $\mathcal{O}_{J'}$ which is the same as $\mathcal{O}_J$, except for that it intersects the line of the logical qubit i in the network $\mathcal{N}$ just after the rotation implemented via the measurement on the cluster qubit j. The potential byproduct operator which is generated via the measurement on cluster qubit k and then propagated forward to the cuts $\mathcal{O}_J$ and $\mathcal{O}_{J'}$, is denoted by $U_k|_{\mathcal{O}_J}$ and $U_k|_{\mathcal{O}_{J'}}$, respectively (the byproduct operators which are actually generated are $\left(U_k|_{\mathcal{O}_J}\right)^{s_k}$ and $\left(U_k|_{\mathcal{O}_{J'}}\right)^{s_k}$). Further, we denote the restriction of the byproduct operators $U_k|_{\mathcal{O}_J}$ and $U_k|_{\mathcal{O}_{J'}}$ to the logical qubit i by $\left[U_k|_{\mathcal{O}_J}\right]_i$ and $\left[U_k|_{\mathcal{O}_{J'}}\right]_i$. The two cuts differ only on the logical qubit i, and there only by the side of the respective cut on which the rotation is located. Therefore, using (11), it follows that $U_k|_{\mathcal{O}_J} = U_k|_{\mathcal{O}_{J'}}$. Hence also

$$\left[U_k|_{\mathcal{O}_J}\right]_i = \left[U_k|_{\mathcal{O}_{J'}}\right]_i . \tag{A.1}$$

If the rotation implemented via the measurement on cluster qubit j is about the x-axis, then the measurement on qubit j has to wait for the measurement on cluster qubit k iff $\left[U_k|_{\mathcal{O}_J}\right]_i$ contains a contribution σ_z. The measurement on j itself produces a potential byproduct operator $\left[U_j|_{\mathcal{O}_{J'}}\right]_i = \sigma_x$. Similarly, if the rotation implemented via the measurement on j is about the z-axis then the measurement on j has to wait for the measurement on k iff $\left[U_k|_{\mathcal{O}_J}\right]_i$

contains a contribution σ_x. The measurement on j itself produces a potential byproduct operator $\left[U_j|_{\mathcal{O}_{J'}}\right]_i = \sigma_z$.

Because of (A.1) U_k can as well be evaluated at the cut $\mathcal{O}_{J'}$ instead of $\mathcal{O}_J$. The byproduct operator on the intersection of qubit line i and cut $\mathcal{O}_{J'}$ resulting from the measurement on qubit j can be written in the form

$$\left[U_j|_{\mathcal{O}_{J'}}\right]_i = \left(\sigma_x^{(i)}\right)^{x_{j,i}} \left(\sigma_z^{(i)}\right)^{z_{j,i}} \text{ with } \begin{pmatrix} x_{j,i} \\ z_{j,i} \end{pmatrix} = \begin{cases} \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \text{for } z\text{-rotations} \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \text{for } x\text{-rotations} \end{cases} . \tag{A.2}$$

The byproduct operator on the intersection of qubit line i and cut $\mathcal{O}_{J'}$ resulting from the measurement on qubit k reads

$$\left[U_k|_{\mathcal{O}_{J'}}\right]_i = \left(\sigma_x^{(i)}\right)^{x_{k,i}} \left(\sigma_z^{(i)}\right)^{z_{k,i}} . \tag{A.3}$$

One can now easily check for both the cases of an x- and a z-rotation implemented by the measurement on qubit j that the measurement of qubit j must wait for the result of the measurement of qubit k iff

$$x_{j,i} z_{k,i} + z_{j,i} x_{k,i} = 1 \bmod 2. \tag{A.4}$$

Now we note that the correspondence between $\begin{pmatrix} x_{j,i} \\ z_{j,i} \end{pmatrix}$ and $\left[U_j|_{\mathcal{O}_{J'}}\right]_i$; and between $\begin{pmatrix} x_{k,i} \\ z_{k,i} \end{pmatrix}$ and $\left[U_k|_{\mathcal{O}_{J'}}\right]_i$ is via the restriction of the isomorphism (26) on qubit i. Thus, $x_{j,i}, z_{j,i}$ are just the i-components of $\mathbf{I}_x|_{\mathcal{O}_{J'}}$ and $\mathbf{I}_z|_{\mathcal{O}_{J'}}$, respectively. Equivalent relations hold for $x_{k,i}, z_{k,i}$. One finds

$$\begin{array}{lll} x_{j,i} = \left[I_{x,j}|_{\mathcal{O}_{J'}}\right]_i & , & z_{j,i} = \left[I_{z,j}|_{\mathcal{O}_{J'}}\right]_i \\ x_{k,i} = \left[I_{x,k}|_{\mathcal{O}_{J'}}\right]_i & , & z_{k,i} = \left[I_{z,k}|_{\mathcal{O}_{J'}}\right]_i \end{array} . \tag{A.5}$$

Further we observe that

$$\left[I_{x,j}|_{\mathcal{O}_{J'}}\right]_l = 0, \quad \left[I_{z,j}|_{\mathcal{O}_{J'}}\right]_l = 0 \text{ for all } l \neq i, \tag{A.6}$$

since the byproduct operator introduced by the implementation of the rotation acts, at the cut $\mathcal{O}_{J'}$, non-trivially only on the logical qubit i. Thus we can write

$$\begin{aligned} x_{j,i} z_{k,i} + z_{j,i} x_{k,i} &= \sum_{l=1}^{n} x_{j,l} z_{k,l} + z_{j,l} x_{k,l} \\ &= \left(\mathbf{I}_j|_{\mathcal{O}_{J'}}, \mathbf{I}_k|_{\mathcal{O}_{J'}}\right)_S \\ &= \left(\mathbf{I}_j|_{\Omega}, \mathbf{I}_k|_{\Omega}\right)_S \\ &= \left(\mathbf{F}_j, \mathbf{F}_k\right)_S, \end{aligned} \tag{A.7}$$

where the second line holds by the definition (36) and the third by (29) and the conservation (37) of the symplectic scalar product. Inserting (A.7) into (A.4) yields

$$\mathcal{O}_K \leq \mathcal{O}_J : \quad j \in \mathrm{fc}(k) \Longleftrightarrow \left(\mathbf{F}_j, \mathbf{F}_k\right)_S = 1. \tag{A.8}$$

For $\mathcal{O}_K \leq \mathcal{O}_J$, $j \in \mathrm{bc}(k)$ cannot occur, hence with (A.8),

$$\mathcal{O}_K \leq \mathcal{O}_J : \quad j \in \mathrm{fc}(k) \vee j \in \mathrm{bc}(k) \Longleftrightarrow \left(\mathbf{F}_j, \mathbf{F}_k\right)_S = 1. \tag{A.9}$$

Case II: $\mathcal{O}_J \leq \mathcal{O}_K$.
First we observe that j can only be in the backward cone of k, but not in the forward cone. Thus, the byproduct operator generated via the measurement on k must be propagated backwards in the network to reach the gate for whose implementation qubit j is to be measured. The reasoning is completely analogous to case I, up to the fact that the potential byproduct operator generated via the measurement of cluster qubit k is in this case propagated backwards onto the cut $\mathcal{O}_{J'}$. Qubit j is in the backward cone of qubit k iff the quantity $\left(\mathbf{I}_j|_{\mathcal{O}_{J'}}, \mathbf{I}_k|_{\mathcal{O}_{J'}}\right)_S$ is equal to 1. Again, by conservation (37) of the symplectic scalar product follows

$$\mathcal{O}_J \leq \mathcal{O}_K : \quad j \in \mathrm{bc}(k) \Longleftrightarrow (\mathbf{F}_j, \mathbf{F}_k)_S = 1. \tag{A.10}$$

For $\mathcal{O}_J \leq \mathcal{O}_K$, $j \in \mathrm{fc}(k)$ cannot occur, and therefore with (A.10),

$$\mathcal{O}_J \leq \mathcal{O}_K : \quad j \in \mathrm{fc}(k) \vee j \in \mathrm{bc}(k) \Longleftrightarrow (\mathbf{F}_j, \mathbf{F}_k)_S = 1. \tag{A.11}$$

Now we combine the two cases and with (A.9) and (A.11) we obtain

$$k \in \mathcal{C}, j \in Q^{(1)} : \quad j \in \mathrm{fc}(k) \ \vee \ j \in \mathrm{bc}(k) \Longleftrightarrow (\mathbf{F}_j, \mathbf{F}_k)_S = 1,$$

which proves the cone test (41).

The proof of the cone test for gates (43) goes along the same lines, only the byproduct operator $(U_k)^{s_k}$ generated via the measurement at cluster qubit $k \in \mathcal{C}$ has to be replaced with the byproduct operator $U_0(g)$ of the gate g.

Finally, the proof the forward-backward cone relation shall be outlined. Suppose that $j \in \mathrm{fc}(k)$. With the same methods as used in the proof of (41) one can derive that

$$\begin{array}{rcl} j \in \mathrm{fc}(k) & \Longleftrightarrow & (\mathbf{I}_k|_{\mathcal{O}_J}, \mathbf{I}_j|_{\mathcal{O}_J})_S = 1, \\ k \in \mathrm{bc}(j) & \Longleftrightarrow & (\mathbf{I}_k|_{\mathcal{O}_K}, \mathbf{I}_j|_{\mathcal{O}_K})_S = 1. \end{array} \tag{A.12}$$

Then, with (A.12) and the invariance (37) of the symplectic scalar product

$$j \in \mathrm{fc}(k) \quad \Longleftrightarrow \quad k \in \mathrm{bc}(j),$$

which proves (42).

Appendix B. Relation among $\kappa'_{k,I}$, $\kappa'_{k,O}$ and κ'_k

Let us briefly explain why the first line of eq. (46) holds. Be the left neighbor of the qubit $k \in \mathcal{C}_O(g) \cap \mathcal{C}_I(\tilde{g})$ in question denoted by $k-1$ and the right neighbor by $k+1$. The interaction S_N takes a product state $|P\rangle_{\mathcal{C}_N} = \bigotimes_{i \in \mathcal{C}_N} |+\rangle_i$ to a cluster state $|\phi\rangle_{\mathcal{C}_N}$. Since all pair-interactions in the Hamiltonian generating S_N commute, S_N can be written in the form $S_N = S_{k-1,k}\, S_{k,k+1}\, S'$, where $S_{k-1,k}$ describes the pair-interaction between $k-1$ and k, $S_{k,k+1}$ the pair-interaction between k and $k+1$, and S' all the remaining pair-interactions. One can easily check that

$$\begin{array}{rcl} S_N\, \sigma_x^{(k)}\, S_N^\dagger & = & (-1)^\alpha \sigma_z^{(k-1)} \sigma_x^{(k)} \sigma_z^{(k+1)} \\ S_{k-1,k}\, \sigma_x^{(k)}\, S_{k-1,k}^\dagger & = & (-1)^\beta \sigma_z^{(k-1)} \sigma_x^{(k)} \\ S_{k,k+1}\, \sigma_x^{(k)}\, S_{k,k+1}^\dagger & = & (-1)^\gamma \sigma_x^{(k)} \sigma_z^{(k+1)} \\ S_{k-1,k}\, \sigma_z^{(k+1)}\, S_{k-1,k}^\dagger & = & \sigma_z^{(k+1)}, \end{array} \tag{B.1}$$

with $\alpha, \beta, \gamma \in \{0, 1\}$. Since the state $|P\rangle_{\mathcal{C}_N}$ obeys the eigenvalue equation $|P\rangle_{\mathcal{C}_N} = \sigma_x^{(k)} |P\rangle_{\mathcal{C}_N}$ the state $|\phi\rangle_{\mathcal{C}_N}$ obeys $|\phi\rangle_{\mathcal{C}_N} = S_N \, \sigma_x^{(k)} \, S_N^\dagger \, |\phi\rangle_{\mathcal{C}_N}$. Thus, via eqs. (2) and (B.1) we can identify $\alpha = \kappa'_k$. Further holds

$$S_N \, \sigma_x^{(k)} \, S_N^\dagger = S_{k-1,k} S_{k,k+1} \, \sigma_x^{(k)} \, S_{k+1,k}^\dagger S_{k,k-1}^\dagger = (-1)^{\beta+\gamma} \sigma_z^{(k-1)} \sigma_x^{(k)} \sigma_z^{(k+1)}, \tag{B.2}$$

such that $\alpha = \kappa'_k = \beta + \gamma$. For the cluster state $|\phi\rangle_{\mathcal{C}(g)}$ on the cluster $\mathcal{C}(g)$ where only the qubits $k-1$ and k are present, but not qubit $k+1$, there holds $|\phi\rangle_{\mathcal{C}(g)} = S_{k-1,k} \sigma_x^{(k)} \, S_{k-1,k}^\dagger \, |\phi\rangle_{\mathcal{C}(g)}$. Via eqs. (2) and (B.1) we can thus identify $\beta = \kappa'_{k,O}$. In the same way, if we look at the cluster state $|\phi\rangle_{\mathcal{C}(g')}$ on the cluster $\mathcal{C}(g')$ we can identify $\gamma = \kappa'_{k,I}$. Combining the relations for α, β, γ, we finally obtain $\kappa'_k = \kappa'_{k,I} + \kappa'_{k,O}$ as in eq. (46).

Appendix C. Temporal complexity of the classical pre-processing

The time that it takes for a classical computer (i.e. a compiler) to translate an algorithm into a machine-specific set of operations (i.e. the machine code) is usually not regarded as to count for the temporal complexity of that algorithm. For quantum logic networks this viewpoint is certainly justified because there the complexity to compute the circuit layout is well understood and known not to exceed the complexity of the quantum logic network itself.

For the $\mathrm{QC}_{\mathcal{C}}$ however, the situation is less clear. For the time being, we do not know of any other way to obtain the quantities characterizing a $\mathrm{QC}_{\mathcal{C}}$-computation than to derive them from the network formulation of the respective algorithm. Therefore, we must exclude the possibility that for the $\mathrm{QC}_{\mathcal{C}}$ the algorithmic complexity of a quantum computation is shuffled from the genuine quantum part of the computation to the classical pre-processing, and that this classical pre-processing may be exponentially hard. As will be shown below, such a case does not occur. All the classical pre-processing can be done in polynomial time.

To see this, we assume that the quantum algorithm on n logical qubits is given as a sequence of $\|\mathcal{N}\|$ elementary gates. For good quantum algorithms, $\|\mathcal{N}\|$ is polynomial in n, as is $\|\mathcal{C}\|$, the number of physical qubits in the cluster $\mathcal{C}$ required to run the algorithm (see Section 5.3).

The layout of the measurement pattern requires to assign $\|Q_0\|$ measurement bases and $\|\mathcal{C} \backslash Q_0\|$ angles. Creating the pattern is for itself not a problem since it can be obtained by patching together the measurement patterns of the elementary gates which are available in block form. The temporal complexity for this step is thus $O(\|\mathcal{C}\|)$.

To obtain the byproduct images we introduce $\|\mathcal{N}\|$ vertical cuts $\mathcal{O}_i$, $i = 1, .., \|\mathcal{N}\|$ to the network, one after each gate (such that $\mathcal{O}_{\|\mathcal{N}\|} = \Omega$) and compute the $2n \times 2n$-matrices $C(\mathcal{N}_{\mathcal{O}_i \to \Omega})$ for $i = 1, .., \|\mathcal{N}\| - 1$, starting with $i = \|\mathcal{N}\| - 1$. The operational effort for this is of the order $O(n^3 \|\mathcal{N}\|)$. By use of these matrices the byproduct images for cluster qubits $k \in \mathcal{C} \backslash Q_{0,z}$ can now be obtained via (29), which requires $O(n^2)$ elementary operations per byproduct image. The way to obtain the byproduct images $\mathbf{F}_g$ of the gates is the same. For $k \in Q_{0,z} \backslash O$ at most four byproduct images have to be added in (60), which requires $O(n)$ operations. The computation of $\mathbf{F}_k$ for $k \in O$ is trivial. Thus, to compute a byproduct image requires at most $O(n^2)$ operations per cluster qubit or gate such that the complexity to compute all of them is at most $O\left(n^2(\|\mathcal{C}\| + \|\mathcal{N}\|)\right)$.

The backward- and forward cones of the cluster qubits $k \in \mathcal{C}$ are computed using the temporal ordering of gates in a sequence representing the quantum logic network and the cone

test (41). The number of cone tests that have to be performed in each case is $\|\mathcal{N}\|(\|\mathcal{N}\|-1)/2$ where the computational effort for each test scales like $O(n)$. Thus, the complexity of this step is $O(n\|\mathcal{N}\|^2)$.

The forward cones generate the anti-reflexive semi ordering "$\prec$". The semi ordering can be computed from them in $O(\|\mathcal{C}\|^5)$ steps.

For each set Q_t there have to be $\|Q^{(t)}\| \leq \|\mathcal{C}\|$ test of the relation $j \prec k$, $j \in Q^{(t)}$ performed to check whether some qubit $k \in Q^{(t)}$ is in Q_t. Also, $\|Q^{(t)}\|$ qubits have to be checked for each Q_t. At most $\|\mathcal{C}\|$ sets Q_t exist such that the operational effort to obtain the these sets is $O(\|\mathcal{C}\|^3)$.

As far as the stated upper bounds are conclusive, it looks as if the computation of the anti-reflexive semi ordering is the toughest part. However, as elementary a relation between the cluster qubits "$\prec$" is, for the conversion of a quantum logic network into a $\mathrm{QC}_{\mathcal{C}}$-algorithm it needs not be computed. Please note that the semi ordering is finally only needed to compute the sets Q_t via (20). But instead of computing "$\prec$" from the forward cones and the sets Q_t from "$\prec$", the sets Q_t can also be computed from the forward cones directly. For this, please note that $\exists j \in Q_t |\ j \prec k \in Q^{(t)} \Longleftrightarrow \exists j' \in Q_t |\ k \in \mathrm{fc}(j')$. The direction "$\Longleftarrow$" is obvious with $j = j'$. The opposite direction, "$\Longrightarrow$", holds by an argument analogous to the one justifying (90). In fact, statement "$\Longrightarrow$" is the same as (90) with $Q^{(1)}$ replaced by $Q^{(t)}$. Thus, eq. (20) can be replaced by

$$Q_t = \left\{ k \in Q^{(t)} |\ \neg\exists j' \in Q^{(t)} :\ k \in \mathrm{fc}(j') \right\}. \tag{C.1}$$

To set the algorithm angles via (71), (72) requires at most $\|\mathcal{C}\| + \|\mathcal{N}\|$ additions per angle and there are at most $\|\mathcal{C}\|$ such angles. Hence, in total it takes $O\left(\|\mathcal{C}\|(\|\mathcal{C}\| + \|\mathcal{N}\|)\right)$ operations to set them. Finally, to initialize the information flow vector via (69) requires $O\left(n(\|\mathcal{C}\| + \|\mathcal{N}\|)\right)$ operations. Thus we see that all classical processing requires only a polynomial overhead of elementary operations and can therefore be done in polynomial time.